Information Filtering In High Velocity Text Streams Using Limited Memory

An Event-Driven Approach to Text Stream Analysis

Inauguraldissertation zur Erlangung der Doktorwürde der Fakultät für Sprach-, Literatur- und Kulturwissenschaften der Universität Regensburg

eingereicht

von

Andreas Bauer

aus

Pilsting

2015

Gutachter: Prof. Dr. phil. Christian Wolff
Zweitgutachter: Apl. Prof. Dr. phil. Thomas Mandl (Hildesheim)

Dedicated to Jonathan, Severin and Kathi.

Acknowledgments

First and foremost, I want to thank my advisor Prof. Dr. Christian Wolff, who offered me the opportunity to write this dissertation, who guided me through to whole process, and who always provided me help and support when needed. He was always willing to review my progresses and to question the results. Throughout the – not always easy – course of this dissertation he provided guidance, advice and valuable feedback and made it thereby possible to successfully finish this dissertation. I also want to thank Prof. Dr. Rainer Hammwöhner, PD. Dr. Jürgen Reischer and the fellows of the PhD seminar for their stimulating and challenging comments, which helped to shape this dissertation. Finally, I want to thank Prof. Dr. Thomas Mandl who took over the part of the second reviewer without hesitating after the sudden decease of Prof. Rainer Hammwöhner.

I would also like to thank Stefan Sagstetter, Holger Striegl, Florian Springer, Roland Erber and Christian Lesny for their reviews, proofreading and their help with the evaluation of the reference model.

Furthermore, I would like to thank my parents, who enabled me to pursue my academic aspirations and provided me the means to successfully finish my education.

Finally, I want to thank my beloved wife Kathi, who always supported me and this project throughout the years. Thanks for helping me through the hard times and backing me up while I was working on this dissertation. And thanks for having our two beautiful sons, Jonathan and Severin, and for sparing the time and taking over the one or two extra diapers, while I was setting on my desk and working on this project.

Abstract

This dissertation is concerned with the processing of high velocity text using event processing means. It comprises a scientific approach for combining the area of information filtering and event processing, in order to analyse fast and voluminous streams of text. In order to be able to process text streams within event driven means, an event reference model was developed that allows for the conversion of unstructured or semi-structured text streams into discrete event types on which event processing engines can operate. Additionally, a set of essential reference processes in the domain of information filtering and text stream analysis were described using event-driven concepts. In a second step, a reference architecture was designed that described essential architectural components required for the design of information filtering and text stream analysis systems in an event-driven manner. Further to this, a set of architectural patterns for building event driven text analysis systems was derived that support the design and implementation of such systems. Subsequently, a prototype was built using the theoretic foundations. This system was initially used to study the effect of sliding window sizes on the properties of dynamic sub-corpora. It could be shown that small sliding window based corpora are similar to larger sliding windows and thus can be used as a resource-saving alternative. Next, a study of several linguistic aspects of text streams was undertaken that showed that event stream summary statistics can provide interesting insights into the characteristics of high velocity text streams. Finally, four essential information filtering and text stream analysis components were studied, *viz.* filter policies, term weighting, thresholds and query expansion. These were studied using three temporal search profile types and were evaluated using standard performance measures. The goal was to study the efficiency of traditional as well as new algorithms within the given context of high velocity text stream data, in order to provide advise which methods work best. The results of this dissertation are intended to provide software architects and developers with valuable information for the design and implementation of event-driven text stream analysis systems.

Zusammenfassung

Diese Dissertation befasst sich mit der Nutzbarmachung von ereignisverarbeitenden Methoden für die Verarbeitung von hochfrequenten Textdatenströmen. Um ereignisverarbeitende Methoden auf Textdatenströme anwenden zu können, muss deren Operationalisierbarkeit in diesem Kontext zunächst hergestellt werden. Hierfür wurde ein Ereignisreferenzmodell entwickelt, das es erlaubt Textdatenströme auf dedizierte Ereignistypen abzubilden und dann mit ereignisverarbeitenden Methoden zu verarbeiten. Dieses Referenzmodell wurde anhand wissenschaftlich akzeptierter Standards entwickelt und mit einem Fragebogen empirisch evaluiert. Zusätzlich wurden verschiedene Referenzprozesse aus dem Bereich Informationsfilterung und Textstromanalyse mit Hilfe ereignisbasierter Konzepte beschrieben. Im nächsten Schritt wurde eine Referenzarchitektur entworfen, die die nötigen Softwarearchitekturkomponenten im Kontext von Informationsfilterung und Textstromanalyse beschreibt und strukturiert. Weiterhin wurde ein Basissatz an Architekturmustern zur Gestaltung von ereignisbasierten Systemen beschrieben. Basierend auf diesen theoretischen Konstrukten wurde ein ereignisbasiertes System zur Informationsfilterung und Textstromanalyse prototypisch umgesetzt. Dieses System wurde dann verwendet um zunächst zu untersuchen, welchen Einfluss die Größe gleitender Fenster auf die Eigenschaften eines dynamischen Subkorpus hat. Es konnte gezeigt werden, dass Korpora basierend auf kleinen, gleitenden Fenster dieselben Eigenschaften aufweisen wie größere und somit als eine ressourcenschonende Alternative verwendet werden können. Danach wurde das System verwendet, um verschiedene linguistische Merkmale der untersuchten Referenzkorpora zu untersuchen. Dabei konnte anschaulich gezeigt werden, dass sich basierend auf den gleitenden Fenster interessante Aspekte von hochfrequenten Textdatenströmen visualiseren lassen. Im Anschluss wurden vier zentrale Komponenten im Bereich der Informationsfilterung untersucht: Filterregeln, Termgewichtung, Schwellwertmethoden und Abfrageerweiterungsmethoden. Diese Methoden wurde hinsichtlich ihrer Effizienz unter Verwendung gängiger Evaluierungsmetriken untersucht. Die Ergebnisse der vorliegenden Dissertation dienen dazu Softwarearchitekten und Entwicklern im Umfeld der Verarbeitung von Textdatenströmen bei der Planung und Implementierung von entsprechenden Systemen zu unterstützen.

Contents

List of Figures

List of Tables

Part I

Introduction and foundations

Chapter 1

Introduction and motivation

1.1 Problem context

Throughout 2014, the internet saw almost 300,000 Tweets being tweeted, 2,5 million pieces of content being shared on Facebook, and more than 200 million emails being sent every minute (Domo, 2014). A plethora of internet data is being generated every moment on the internet as audio or video uploads, blog posts, chat messages, log messages, tweets or emails. And the amount of data is continuing to grow, as more and more people are able to access the internet and produce digital content[1].

Data volume

Applications like *Twitter*, *Wikipedia* or *Facebook*, which evolved around Web 2.0 concepts like blogging, podcasting, wikis or social networking in the mid of the 2000s, create never ceasing streams of photos, videos, audio and text content. But not only social media platforms produce text streams, also log messages of computer systems that are supposed to be analysed by developers or administrators, as well as code commits to a large code base or source hosting platforms like *github*[2] produce many short unstructured relevant text documents. All these sources emit data at an unprecedented volume and pace, i.e. they produce high-velocity text streams, and this poses new challenges towards the consumption and processing of the generated data. The briefness and velocity of the documents within the text stream pose special challenges in terms of efficiency and effectiveness towards text analysis and information retrieval methods like term weightings, query expansion or threshold algorithms.

Efficient processing

Efficient processing of high-volume and high-velocity shot document text

[1] The problem context is exclusively focussed on content that is addressed towards human beings and does not include data that is being exchange between internet capable devices.

[2] `http://github.com`

streams requires concepts and technologies that can be used to extract valuable information from a text stream while, at the same time, keeping up with the speed of the arriving data and while using an optimal amount of system resources. Event processing has attracted interest in this context, as *event processing engines*, which form the central component of an event processing system[3], offer capabilities like in-memory calculations, pattern matching and temporal windows that help to address these challenges.

Event processing evolved around the eponymous concept of *events*. An *event* is a flexible way to look at data, as it allows to split up data into the required granularity and to map them to semantic distinct and concise event types. By deconstructing data into events, data becomes processable by event processing engines and their capabilities for efficient processing can be leveraged. Event processing engines like *Esper* [4] or *Apache Spark Streaming* [5] are highly optimized technologies that allow processing several hundred thousand events per second and are well suited for the analysis of data streams (cf. (EsperTech Inc., 2015) or (H. Kumar, 2015)). Thus, event processing has been successfully applied to areas like finance (Adi et al., 2006), RFID data ((F. Wang et al., 2006), (L. Dong et al., 2006)), logistics (Roth & Donath, 2012), transportation (Buchmann et al., 2011) and telecommunication (Vincent, 2008).

Event driven text stream analysis

But despite their capabilities to process large data volumes, text streams have not yet been considered as a primary area of application for event processing. There are several examples where event processing is applied on text streams (cf. (Grigorik, 2011), (Alegria, 2011), (Apache Spark, 2015b) or (Noll, 2013)), but they were focused on demonstrating that text streams and event processing can be combined in principle. However, they did not provide a further discussion on what methods of text stream analysis and information filtering should look like, how they should be structured and how they should operate in this context. All these examples lack a common understanding of the kind of events that can be derived from a document, and their semantics and properties. In general, a common event reference model and reference architecture for event driven text stream processing that help to plan, design and implement event driven text stream analysis and information filtering systems are still missing. These two aspects are –among others – the research gaps that are addressed in this dissertation.

Information overload

Besides the basic requirement to process text streams, another issue in the context of high-velocity text streams arises: *information overload*. The term

[3] More details on the foundations of event processing can be found in chapter 2

[4] `http://espertech.com`

[5] `https://spark.apache.org/streaming`

was popularized by Alvin Toffler[6] in his book *Future Shock* (Toffler, 1970). There, he argued that

> [we] are accelerating the generalized rate of change in society. We are forcing people to adapt to a new life pace, to confront novel situations and master them in ever shorter intervals. We are forcing them to choose among fast-multiplying options. We are, in other words, forcing them to process information at a far more rapid pace than was necessary in slowly-evolving societies. There can be little doubt that we are subjecting at least some of them to cognitive overstimulation. What consequences this may have for mental health in the techno-societies has yet to be determined. (Toffler, 1970, p. 354)

Information filtering

Information overload was already considered a major challenge back in the 1970s. Nowadays the availability of data has grown in an unprecedented manner and the challenges caused by information overload have exacerbated. Information filtering addresses information overload by removing irrelevant documents from a stream of documents. The process of information filtering, in contrast to information retrieval, is to formulate standing queries that express long-term information needs and which filter potentially relevant information from the stream of incoming data[7]. One of the first to introduce this concept was Peter J. Denning in 1982 (Denning, 1982). Even though the amount of data that was generated then was marginal compared to the current data flow, he already argued that "[i]t is now time to focus more attention on receiving information – the processes of controlling and filtering information that reaches the persons who must use it" (Denning, 1982, p. 163). This means the amount of information must be adjusted to the "personal mental bandwidth" of the user (Denning, 1982, p. 164). And in order to be able to limit the amount of data a user receives, it should be assured that the concepts and methods used for filtering information must be optimized. Therefore, existing information filtering concepts like filter policies, term weighting or thresholds must be re-evaluated in the context of high-velocity text streams, in order to assess the performance these methods yield under the new conditions. Furthermore, methods need to be designed and evaluated that explicitly address characteristics of high volume text streams, e.g. heterogeneity and volatility. These are further aspects addressed in this dissertation.

[6] The term was first mentioned in (Gross, 1964).

[7] For more details cf. section 3.2.1

1.2 Scholarly context: information science

Computer science

Before the research objectives are discussed, the scholarly context of this dissertation is defined. Information science and computer science are related but separate research areas (cf. (Kunz & Rittel, 2000, p. 18)). In order to avoid misunderstandings during the evaluation of this dissertation, this section will elaborate the information scientific viewpoint. This is important, as readers may expect answers to questions that would have to be addressed by a pure computer science thesis. This foremost entails the definition of new algorithms, data structures or programming models to address the problems of processing high-volumes of text stream from a computational point of view. This is not the case. The goal is to leverage methods from computer science, in order to analyse high-velocity text streams.

Information science

(Tucker, 2004, p. 19) defines "[c]omputer science [a]s the study of computational processes and information structures, including their hardware realizations, their linguistic models, and their applications". But this dissertation is an information science dissertation, i.e. it delivers answers to questions regarding how information can be extracted and derived from high-velocity text streams, how the properties of this information behave and how systems can be designed that analyse high-velocity text streams.

Borko's definition

This notion aligns with the definition of information science given by Harold Borko given in (Borko, 1968). It embraces all aspects that are addressed in this dissertation. (Borko, 1968, p. 3) defines *information science* as

> [a] discipline that investigates the properties and behavior[sic] of information, the forces governing the flow of information, and the means of processing information for optimum accessibility and usability. It is concerned with that body of knowledge relating to the origination, collection, organization, storage, retrieval, interpretation, transmission, transformation, and utilization of information. This includes the investigation of information representations in both natural and artificial systems, the use of codes for efficient message transmission, and the study of information processing devices and techniques such as computers and their programming systems.... It has both a pure science component ... and a applied science component, which develops services and products.

Dissertation foci

That is why the approach, deliverables and results of this thesis are governed by the principles stated in Borko's definition. Consequently the following

foci for this dissertation are defined:

- The design and evaluation of an event reference model for event driven text stream analysis and information filtering systems.
- The design and implementation of a reference architecture for event driven text stream analysis and information filtering systems.
- The study of the behaviours and characteristics of text streams properties using the two previous artefacts.
- The empirical evaluation of the information filtering components such as filter policies, term weights, thresholds and relevance feedback within high-velocity text stream.

1.3 Objectives

This section elaborates upon the aforementioned foci and derives the research objectives of this dissertation. (Kelter, n.d.) stated that the main objective of a dissertation is to find a solution to a problem that has not been (satisfactorily) solved yet. This means that besides applying the known principles of the investigated research area, it is mandatory to come up with something genuine and unseen. Thus, the dissertation will address following research objectives:

1. **The application of event processing on text stream analysis**
   Event processing provides means that allow for efficiently operating on data streams[8]. Text streams are a special kind of data streams and therefore it is logical to use event processing for efficient text stream processing. This dissertation presents an approach to combining text streams and event processing in a sound scientific way by proposing reference models and architectures and evaluating these concepts with real-world data.

2. **Event reference model for text stream processing**
   Processing unstructured matter like text requires means that allow for dissecting a text document into its components and mapping them to discrete and well-defined event entities that can be processed by event processing engines. Therefore, an event reference model is introduced that structures concepts from text analysis, corpus analysis, information retrieval and information filtering into an event model, which can be used to construct text stream processing applications.

[8] This entails e.g. temporal windows or standing queries. For details cf. 2

3. **Reference architecture, reference processes and architecture patterns for text stream processing and information filtering applications**
   The processing of high-velocity text streams poses various challenges to a system that is intended to execute such a task. This comprises tasks like text pre-processing, event mapping, term weight calculation, model adaptation, query expansion, etc. In order to facilitate the design and implementation of event driven text stream analysis and information filtering systems, a layered reference architecture is proposed that describes the essential text domain related task and contextualizes them for event driven systems. Furthermore, a set of reference processes and how they can be designed to be implemented within event driven systems is introduced. Finally, a basic set of architectural patterns is introduced that helps to address basic architectural design decisions.

4. **Studying the effects of windows size on text stream properties and automated ways for sliding window size estimation**
   Event processing offers the possibility to operate on data streams using sliding windows. Sliding windows represent computationally efficient mechanisms to process data streams, as they only considered a dynamic subset of the data stream. One of the main goals of this dissertation is to show if sliding windows can efficiently operate on text streams. The central question is if a *dynamic sub-corpus* approach, which only contains a small subset of the whole data stream, will perform sufficiently well in text stream analysis and information filtering tasks with a decent performance. Therefore, structural and statistical properties of sliding windows with varying sizes are studied, in order to derive a reasonable lower bound for the used test corpora. Furthermore, methods are studied to automatically derive suitable window sizes. Finally all the approaches are evaluated.

5. **Automated definition of filter policies**
   Filter policies act as gatekeepers to computationally expensive processing stages, like query expansion or score calculations, of a text stream or information filtering system. This means a filter policy is supposed to remove as many irrelevant documents as possible without removing too many relevant documents. This dissertation compares three types of filter policies: heuristic filter policies, filter policies based on frequent item patterns and classifier based policies. The frequent item based approach represents a new way to automatically define filter policies. All three policies are compared in terms of common performance measures like precision or recall, and recommendations are given on which approach works well in specific scenarios.

6. **Design, comparison and evaluation of term weighting methods**
   One of most important factors in estimating the relevance of a document is the score that can be attributed to the terms contained in a document. There are many different term weighting schemes, but these were in fact conceived for long text documents and not explicitly for the short documents that are the subject of this dissertation. One could assume that term weighting schemes work for short and long documents likewise, but if that is really true, there is not scientific comparison. Additionally, there is no extensive study on how different term weighting schemes perform in a high-velocity text stream scenario.
   Furthermore, new term weighting schemes are studied that explicitly address the nature of high-velocity text streams. All these term weighting schemes are compared and evaluated in terms of their performance regarding precision or recall on two social media text stream corpora. The evaluation is performed on an overall, macro-average level as well as on search profile category level[9].

7. **Extensive study of threshold methods**
   The goal of information filtering is to classify documents into categories of relevance and non-relevance. In order to execute this classification thresholds are required that mark the border of each category. Common approaches in information filtering use distribution-mixture models, where the intersection of the distribution of the relevant and non-relevant class marks the threshold. The shortcoming of this approach is that in reality the scores of document classes do not follow a parametric distribution with fixed parameters.
   Therefore the ability of event processing engines to monitor a multitude of different statistical parameters is exploited to evaluate and compare various threshold strategies. The goal is to show how different threshold strategies can be modelled and how these perform on different topics and search profile categories. These computationally efficient methods are also compared against a variety of machine learning classifiers due to two reasons. First, machine learning has become the leading approach to address data analysis problems and, second, the efficiency and performance of machine learning algorithms, which use features that are based on the proposed sub-corpus approach, compared to the simpler threshold method are relevant to this study.

8. **Design, comparison and evaluation of query expansion methods**

[9]The search profiles (search queries), which were used for the evaluation tasks, were classified into three temporal profile categories according to (R. Jones & Diaz, 2007). More details can be found in section 8.1.2

As previously mentioned; a plethora of data is being generated every instant. Therefore, it can usually be assumed that there are enough relevant documents available regarding a given search profile. But nevertheless, too few relevant documents, or simply the wrong ones, may be encountered. Also, specific domains like law investigation or crime surveillance are not supposed to miss relevant information. Therefore, higher recall values are required. Thus, relevant feedback and query expansion are required to broaden the scope of search profiles in order to provide the required coverage. To address this, several state of the art methods for generating query expansion terms are compared against a new relevance feedback method called *ReNove* that addresses explicitly the requirements of documents in high-velocity text streams.

1.4 Scope and limitations

Textual data only

The focus of this dissertation is on textual data only. Data streams consisting of videos, audio content or photos are not included in this thesis. Furthermore, only textual data that is addressed towards human beings are considered, i.e. data streams that are generated by machines such as messages streams that are exchanged between machines and are only intended to digested by machines, e.g. in the context of the *Internet of Things*, are not considered.

Short documents

Second, the focus is on short documents, i.e. the results of this dissertation are applicable on document types such as news headlines, Tweets, comments, source code commits, email headlines, on short messages like SMS or e.g. *WhatsApp* messages or short blog posts consisting only of a few lines. The only requirement is that the documents are created continuously and thereby form a stream of text. For several reasons, the scope is narrowed to these types of documents. First, multimedia data streams require different approaches to make the content accessible to computers and to analyse it than text. This exceeds the scope of a single dissertation. Second, analysis of multimedia data is rather different to text analysis. Again, this would exceed the scope of a single dissertation. Third, long documents are not emitted at the same speed as short documents because of the time it takes to create them.

Third, this dissertation only considers text streams with a certain event emittance rate. This means the source that is monitored should produce at least several hundred or several thousand events per second.

Finally, this thesis is an information science dissertation, and event process-

ing is used where it supports the investigation of the subject matter *high-velocity text streams*. This means event processing represents a means to achieve the goal of analysing high-volume text streams, i.e. the advance is in terms of text stream analysis and information filtering. It is not focused on enhancing generic event processing aspects such as quality-of-service aspect, event processing query languages, pattern matching algorithms or similar issues.

1.5 Research methodology

Design Science

The research methodology that is used in this dissertation is based on *Design Science*. In their paper "Design Science in Information Systems"; (Hevner et al., 2004) presented a way of "how to conduct, evaluate, and present design-science research" for information systems. They argue that "Design Science is inherently a problem solving process. The fundamental principle of design-science research ... is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact[sic]" (Hevner et al., 2004, p. 82).

Design Science is a pragmatic approach that is focused on real world problems. In the context of information systems it provides a framework that guides the researcher through the process of finding a suitable and satisfying solution to a given problem. Design Science is centred around the idea of "learning through building" (Vaishnavi & Kuechler, 2004). This means you can study a problem best when you build an artefact, a piece of software, and study the problem you want to solve by analysing this problem with the artefact you have designed and built.

Tasks of information science researchers

In this sense Design Science supports the *tasks of information science researchers*- as they are put forward (Kunz & Rittel, 2000). Their seminal paper proposes a range of tasks for information scientists. They claim that objectives for an information scientist are e.g. the research and development as well as the design and operation of information systems, including their constituting parts and processes (Kunz & Rittel, 2000, p. 42). These objectives can be fulfilled by the guidelines provided by Design Science.

A comprehensible overview of the guidelines in Design Science, as well as how these are addressed in this thesis is provided in table 1.1. The guidelines were presented by (Hevner & Chatterjee, 2010, p. 12ff).

Guideline	Description	Addressed by
No. 1: Design as an artefact	Design Science research is supposed to produce viable artefacts in the form of a construct, a model, a method, or an instantiation.	Design of an event reference model, a reference architecture and the relevant implementation.
No. 2: Problem relevance	The objective of Design Science research is to develop technology-based solutions to important and relevant business problems.	High volume and high-velocity text streams are ubiquitous. Event processing offers viable means to solve the filtering and text processing challenges that are posed by such a setting. Previous research in the fields of event processing, text stream processing and information filtering have not addressed the two aspects jointly. Therefore, this dissertation studied the challenges and requirements of event driven information filtering.
No. 3: Design evaluation	The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.	The event reference model is empirically evaluated by a survey among IT experts. The reference architecture is evaluated by the implementation in a real-world system. The system's efficacy is evaluated by its implementation and the assessment of the filtering performance, using accepted information retrieval performance measures..

Guideline	Description	Addressed by
No. 4: Research contributions	Effective Design Science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies.	The findings of this thesis were published in various papers.
No. 5: Research rigour	Design Science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.	The system is evaluated according to the latest information retrieval evaluation methods.
No. 6: Design as a search process	The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.	The target system *StreamFI* was created iteratively using modern technologies like event processing engines and in-memory databases. The Java implementation was guided by the latest software engineering standards. The implementation of the concrete system and design of the reference architecture were influenced by each other.
No. 7: Communication of research	Design Science research must be presented effectively to both technology-oriented and management-oriented audiences.	The progress and findings of this thesis have been presented within the PhD seminar of the Information Science chair at the University of Regensburg. Furthermore, findings of this dissertation were published as scientific papers.

Table 1.1: Design Science guidelines and how these are addressed.

March et al. differentiate strictly between natural and Design Science. They argue "[w]hereas natural science tries to understand reality, Design Science attempts to create things that serve human purposes." (March & Smith, 1995, p. 253). It is obvious that this statement influenced the formulation of the

guidelines proposed by Hevner et al. , but it also underpins the principle to not theorize about abstract concepts, but to build and evaluate artefacts that people really need and really use. Building and evaluating are the basic activities within Design Science according to (March & Smith, 1995, p. 253) They also define four core artefacts Design Science is based on:

1. Constructs: These are the conceptual vocabulary you use within the given domain.
2. Models: They define the relations between the constructs.
3. Methods: They form a set of steps to perform a task within the research activity.
4. Instantiations: They operate on constructs, models and methods.

(March & Smith, 1995, p. 255ff)

All four artefacts were taken into account during the writing of this dissertation. Constructs are addressed by introducing an event reference architecture, models by the introduction of a reference architecture for text stream analysis systems, methods are described in the implementation section and the instantiation of these three products is *StreamFI*, the system used to evaluate the investigated aspects like term weighting or thresholds.

1.6 Thesis structure

The thesis consists of three main parts that contain the individual chapters. *Part I* contains chapters one to three, as well as the foundations of information retrieval and event processing. *Part II* is about the definition of a reference framework that contains an event reference model and a reference architecture. *Part III* contains a description of the implementation and the evaluation study. In detail, the dissertation comprises the following chapters:

Chapter one provides an introduction to the dissertation as well as the research contributions, the research methodology and the outline of the dissertation.

Chapter two introduces the relevant concepts as well as the state of the art in event processing. The chapter highlights the most important concepts in the relevant research fields and provides the required foundations for the rest of the thesis.

Chapter three discusses the relevant concepts as well as related research in the field of information filtering and information retrieval.

Chapter four introduces an event reference model for event driven text stream analysis. The purpose of this reference model is to provide means to dissect unstructured text documents and map them onto semantically concise events. By applying this mapping text becomes processable for event processing engines, and the advantages of stream processing and event processing become available for text stream data.

Chapter five describes a reference architecture for event driven text stream analysis. The goal is provide an architectural framework that covers all relevant aspects (properties, relationships and semantics) of components that are required to describe an information system that is capable to process high-velocity text streams using event processing means.

Chapter six highlights the implementation of the information system *StreamFI* that is based on the theoretical work from chapters four and five and will be used for the execution of the experiments in the subsequent chapters.

Chapter seven focuses on the aspect of *limited memory*. Limited memory in terms of event processing subsumes the usage of sliding windows which represented a limited view into the past of the data stream. In this chapter various sliding windows of different sizes are used to analyse three different social media corpora in terms of various linguistic aspects. The goal is to show if the size of the sliding window really matters in terms of the quality results. Furthermore this analysis provides the basis for an approach to dynamically determine an optimal sliding window size based on *index entropy*.

Chapter eight contains an extensive study of four fundamental information filtering components in an event driven context. This entails filter policies, term weighting, thresholds and relevance feedback. The goal is to evaluate the state of the art in information filtering in terms of their performance in a high-velocity text stream scenario. Additionally, specialized methods that explicitly address high-velocity text streams will be presented for each component. All methods are evaluated in terms of generally accepted performance measures like recall or precision. Furthermore all methods are compared on a macro-average level, on a profile category level (cf. (R. Jones & Diaz, 2007)) and on a search profile level, in order to elaborate which method works best in a special setting.

Chapter nine sums up the results of the dissertation and gives an outlook onto future research subjects.

Summary

In this chapter the problem context of this dissertation was described. Then the scope and limitations were defined followed by an overview of the contributions that can be expected from this dissertation, a discussion on the structure of the dissertation, and an overview of the applied research methodol-

ogy. In the next two chapters the basic principles and relevant related research in the areas of event processing and information filtering are introduced and discussed.

Chapter 2

Event processing – foundations and related research

This chapter introduces the foundations of event processing, including the relevant concepts required in later chapters of this dissertation as well as valuable background knowledge. Because event processing is a rather new aspect within information science research, the introduction is detailed. In addition, the chapter elaborates on the reason to choose event processing as a suitable means to text stream processing[1].

The chapter is structured as follows: section 2.1 describes the history and evolution of event processing. This should illustrate where event processing comes from and justify event processing as an accepted and sound basis for data stream analysis and hence for text stream analysis. Section 2.2 defines the term *event* for the purpose of this dissertation. Section 2.3 presents the buildings blocks and principles of event processing. This knowledge is required, in order to evaluate the event reference model and the reference architecture in chapter 4 and 5. The next section, 2.4, discusses event processing in the context of various *Big Data* technologies. Yet, the goal is to highlight the difference between various available technological approaches to process high-volume and high-velocity data. The chapter is concluded by section 2.5, which discusses the various aspects of *real-time*. This is because this aspects is crucial in the context of this dissertation and should therefore be clarified.

[1] It should be pointed out here, that this dissertation is an information science dissertation that uses event processing as a means to investigate the subject matter of *high-velocity text streams*. It is not concerned with any event processing related research topics like the implementation of event processing engines, quality-of-service aspects for event matching or similar issues.

2.1 The history of event processing

The concept of event processing has a long history in computer science. But, as previously mentioned, it has not yet been studied intensively in context of information science and related areas. In order to understand why event processing can be used in this context, it is helpful to understand how the concept evolved. Furthermore the history of event processing helps to introduce various definitions of events. This is important, because the concept *event* is rather vague and more precise understanding of this concept is required in the later chapters. Therefore various definitions are introduced, in order to be able to provide a concise definition for this dissertation in Chapter 4.

Origins of event processing

The origins of event processing can be traced backed as far as the 1950s. In the article "A Short History of Complex Event Processing" (D. Luckham, 2008) David Luckham identifies "discrete event simulation (DES)" as the first predecessor of modern event processing. DES systems were used to simulate situations during military combat (Nance, 1993, p. 2), waiting time of clients in front of an ATM (Matloff, 2008) or the management of parts inventories (Nance, 1993, p. 2). The DES systems that were used for those simulations had the following three characteristics:

> [They have] a state which evolves or changes with time. Changes in state occur at distinct points in simulation time. A state change moves the system from one state to another instantaneously. State changes are called events. (Preiss, 2000, p. 383)

Events as state change

This definition of DES systems also contains a definition for the concept *event*. In this context events were considered as a change of state. The state change only indicated that the system had to change some properties and advance the system clock to the timestamp of the next event in the ordered event set, which includes all pending events in the system (Matloff, 2008, p. 5f). At that time DES systems were considered "to schedule the flow of the events between components in [a] model, the execution of the components, and the ticking of the clock" (D. Luckham, 2008, p. 2). DES systems already highlight some important aspects of event processing. First, they introduce the notion that events represent a change of state. Second, they incorporate a temporal component (“ticking of the clock”) and third there is the dynamic character (“flow of the events”) of the systems.

Computer networks

The next step in the history of event processing can be considered the rise of computer networks in the 1960s (D. Luckham, 2008). J.C.R. Licklider, one

of the visionaries of the modern internet, already reflected on the need of a network of "thinking centers" that should help to manage the increased requirements in information storage and information retrieval in 1960 (Licklider & Taylor, 1990, p. 8).

This idea got picked up with the development of the *Advanced Research Projects Agency* (ARPA) net, which incorporated the idea of connecting distributed computer systems by using a technique called packet switching (Abbate, 2000, p. 8). Packet switching means that all information, be it text, audio or video, is chunked into equally small blocks called packets. These packets are digital, and contain all information they need, in order to reach their destination, as well as to be reassembled into the original information (Abbate, 2000, p. 17f). The packets are routed over different intermediary servers to their destination. As the packets do not depend on a predefined route, but can flow through any server that knows how to forward the package, the networks using packet switching become more reliable and more robust. The transmission or reception of such a packet was then also considered to be an event (D. Luckham, 2008, p. 3). Foremost the speed and amount of packets in a network required new tools for monitoring and surveillance of the transmission[2] of these events. Hence, these tools can be considered as the first software that dealt with event processing in a broader sense.

Events as packet transmissions

This evolution in the history of event processing introduces another aspect of the concepts of event and event processing. In this context event processing was considered to be a technical term which comprised the transmission of packets. But the concept of event again has the meaning of being something that happens at a certain time and can be detected, i.e. the emission or reception of a network packet was marked at a point in time and denoted the manifestation of something that happened at an observable point in time.

This development made the next step in the history of event processing possible. Reliable and fast connection of distributed computer systems offered new possibilities. But the connection setup was quite tedious and the protocols for data transmissions were complicated to handle. In order to relieve the developers of the burdensome effort of taking care of the network communication themselves, new technologies were invented. In 1976 (J. E. White, 1976, p. 7) proposed the first amendments to the ARPA net, in order to provide a new flexible and efficient application-independent protocol. These ideas lead to the invention of the *Remote Procedure Call*(RPC) protocol. The

Remote Procedure Call

[2]Ping, programmed by Mike Muuss in 1983, was one of the first network monitoring tools. With the increase in network traffic more sophisticated software products were built and resulted in tools now commonly used such as HP Openview, IBM Tivoli or CA UniCenter.

basic idea of RPC was that "[o]ne thread of control logically winds through two processes: the caller's process, and a server's process. The caller process first sends a call message to the server process and waits (blocks) for a reply message" (Sun Microsystems, 1988, p. 2).

This allowed for calling methods on remote computer systems and was the first step into the direction of distributed computing. In the sense of event processing the call message in this scenario can be considered as an event. It again constitutes something that happened at a certain time and is perceptible by a different system than the originator system, i.e. the originator sends an event to the receiver in order to trigger some action. This last fact is a relevant aspect for event processing, because the exchange of events, which can influence other systems, is a fundamental characteristics of event processing systems.

Multisensor data fusion

The next step in the development of event processing was *multisensor data fusion* in the late 1980s and early 1990s (D. Hall et al., 2012, p. 3f). In (F. E. White, 1991, p. 5) data fusion is described as

> [a] process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats as well as their significance.

In order to establish a common terminology the *Joint Directors of Laboratories Data Fusion Working Group* began to develop the JDL process model in 1986. The goal of this model was the unify the diverging nomenclatures and definitions within different research fields related to sensoring and data fusion (D. L. Hall & Llinas, 2001, p. 23f). Figure 2.1 illustrates the different level in the data fusion process.

Message-oriented Middlewares

Another milestone in the history of event processing was the introduction of *message-oriented middlewares* (MOM). They made it possible to asynchronously collaborate and exchange information between computer systems using messages. The first middleware was released in 1985 by Teknekron. The idea behind middleware systems was to facilitate the collaboration of computer systems by encapsulating the details of the communication between the systems and hide the complexity of the distributed message exchange. Teknekron, the predecessor company of Tibco, one of the now leading companies in event processing, came up with the idea of using a software bus, in order to "link different programs and carry crucial information between them" (Roush, 2010). The bus concept was derived from hardware systems and describes

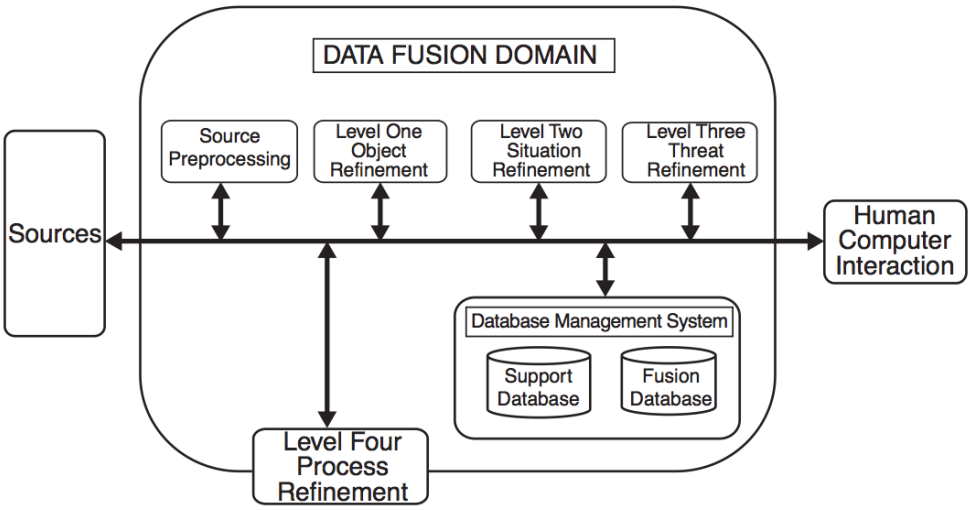

Figure 2.1: The *JDL* data fusion process model (D. L. Hall & Llinas, 2001, p. 25)

a "functional unit for the transfer of data between several participants (functional units for data processing) via a common transmission path, wherein participants are not involved in the transfer of data between other participants" (Commission, 2013).

(Etzion & Niblett, 2011, p. 20) sees MOM as a "transport layer that event processing may employ as an infrastructure to implement event channels". Thereby MOMs support two central principles of event processing: *decoupling* and *push-style interaction*[3].

Active databases

At the end of the 1980s, a further step in the development of event processing was the advent of *Active Database Management Systems* (cf. (McCarthy & Dayal, 1989b), (Dayal et al., 1988)). Database management systems in general became necessary within the rise of electronic data processing and storage of information in databases. Thus they have been in use since the early days of computing and have occupied an important place in the history of computing. A definition for DMBS is given in (Dittrich et al., 1995, p. 2):

> A database management system (DBMS) is a software system for reliably and efficiently creating, maintaining, and operating large, integrated, multi-user databases. It implements storage and retrieval of data, secondary storage management for large sets of objects (including access paths, clustering, etc.), concurrency control, and recovery. The collection of data including secondary information is stored in the database.

[3]For details on event channels and principles of event processing cf. chapter 2.3

Traditional DBMSs act passively, i.e. the data they store are queried by an operator – be it human or machine – and hence only delivers results when they are being asked to do so (McCarthy & Dayal, 1989a, p. 215). The results of the queries are delivered to the operator who is in charge of further processing the results.

In contrast to this active DBMS behave differently, because they offer reactive behaviour, i.e. an "active DBMS is a DBMS that allows users to specify actions to be taken automatically, without user intervention, when certain conditions arise" (McCarthy & Dayal, 1989a, p. 215).

Event-Condition-Action

The foundation of this behaviour is the definition of *Event-Condition-Action(ECA)* rules. These rules define certain conditions that one or several events have to match in order to trigger a predefined action. The intriguing parts of ECA rules in active database systems are the meaning of the terms "event" and "condition". While condition can be matched to the part of pattern matching – details are be described later in the chapter on the fundamentals of event processing – the definition of the term "event" has almost arrived at its final definition for this dissertation. (Dittrich et al., 1995, p. 2) define the term "event" in active DBMS as follows.

> An event can be primitive or composite. Primitive events correspond to elementary occurrences and can be mapped directly to a point in time determined by something occurring in the database system (e.g. a data modification operation), or by something occurring in the database environment (e.g. an absolute point in time). Composite events are defined as combinations of other primitive or composite events using a set of event constructors such as disjunction, conjunction, etc. Composite events are mapped to a point in time based on information about their component events, e.g. the point of time when the last component occurred.

Finanical trading

It was another domain that gave a major push to event processing: financial trading and more specifically algorithmic trading. The aforementioned capabilities of causal event modelling or defining event patterns and matching those patterns attracted the attention of the financial markets. The availability of stock feeds from various stock exchanges as well as the speed of stock tick events required this new way of data processing to be ahead of competitors. Financial trading applications started to use event processing systems for monitoring different sources of stock ticks, to correlate the different tick streams and to trigger actions – usually buy and sell orders – based on highly specialized algorithms[4]. These kinds of algorithms allowed traders to create

[4] An overview of analysis approaches for financial data that can be designed as event pro-

a winning margin because of being able to analyse data moments after they were created. Although the algorithms themselves are usually highly sophisticated, this kind of processing can be referred to as *simple event processing* (SEP) system, because "[i]n simple event processing, a notable event happens, initiating downstream action(s). Simple event processing is commonly used to drive the real-time flow of work taking lag time and cost out of a business" (Michelson, 2006, p. 3). The implementation of such event processing systems was facilitated by the application of the *event driven architecture* (EDA) paradigm. This paradigm describes a loosely coupled software system where the interoperating components are triggered by the exchange of events. The different components work in an asynchronous way and operate independently within the event processing systems. The asynchrony allows for fast and high performance systems that are able to deal with the large volume of events that have to be processed in event processing systems.

Simple event processing

Event driven architecture

The ever increasing volume of data required new, more sophisticated ways of designing, programming and using event based systems. Thus, in the mid of the 1990s this was addressed in several research projects, which were later incorporated into commercial event processing systems. David Luckham and his team created Rapide (D. C. Luckham & Vera, 1995), a system the incorporated event processing capabilities and was one of the first systems that claimed to be capable of doing *complex event processing* (CEP) based on "the Rapide concepts of (1) causal event modeling, (2) event patterns and pattern matching, and (3) event pattern maps and constraints" (D. Luckham, n.d.). Although the commercial spin-off "ePattern" failed (Vincent, 2010), the ideas and foundations were picked up and extended in various other projects, e.g. Infospheres[5], Aurora[6] or Amit (Adi & Etzion, 2004). This kind of event processing was not only focused on high-speed, low-latency real-time processing, but more on the casual relation between events, on event patterns and their detection (D. Luckham, n.d.). CEP is centred around the concept of an *event cloud*[7], where e.g. events can perpetuate and suddenly become relevant, even a long time after their occurrence due to a pattern match. This means that complex event processing operates on a *partially ordered set (POSET)* of events.

Complex event processing

Complex event processing also operates on cross event types, i.e. the combination of different kinds of events originating from different streams, by relating events over a long period of time not necessarily paying attention

cessing systems can be found here `http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators`

[5] `http://www.infospheres.caltech.edu/`

[6] `http://www.cs.brown.edu/research/aurora/`

[7] More details on event clouds can be found in chapter 2.3.2.

to real-time constraints and by looking for temporal, spatial or causal correlations (Michelson, 2006, p. 2). The integration of machine learning and data mining algorithms is also a characteristics of CEP and is used to find unknown patterns within the stream of events.

In this context the term event is amplified beyond its basic meaning (a marker in time of some kind of incident) by adding variety in the application, combination and relation of single events. The basic event steps into the background, is basically used as a fundamental concept to gain deeper and unobtrusive insight into a situation. As an example, when SEP is to calculate the *volume weighted average price* (VWAP) on a 10-minute sliding time window in a single stock tick stream, then CEP is to detect a Ponzi fraud within a numerous amount of stock ticks, news and social media streams.

These more sophisticated systems also allowed extending the application of event processing to new domains. (D. Luckham, n.d.) stated the following possible application areas for event processing systems:

- Hierarchical event viewing applied to the Enterprise IT layer
- Analysing business processes
- Network level monitoring and management
- Cyber security: network intrusion detection
- Enterprise monitoring and management
- Modelling and simulation of collaborative business processes
- Business policy monitoring
- Analysis and debugging of distributed systems

Another area of research is *semantic complex event processing* (cf. (Teymourian & Paschke, 2009), (Zhou et al., 2012),(Riemer et al., 2012)). This field is concerned with the integration of semantic concepts like inference and reasoning into the domain of event processing. This means the goal is to make a logical conclusion about an incoming event by inferring additional knowledge about that event using an external knowledge base.

Garnter Hype Cycle

In 2014 the *Gartner's Hype Cycle* saw complex event processing at the "Peak of Inflated Expectation" (Gartner, Inc., 2014). This means that the early adopters have already investigated the topic, the mass media hype is over, suppliers have proliferated and now the negative press begins (Wikipedia, 2015d). The plateau of productivity for complex event processing is seen to be reached in 5 to 10 years. The fact that CEP is contained in the Gartner's Hype Cycle shows the relevance of event processing and the need for research related to this topic.

In summary, this is the state of event processing at the time of writing this

thesis. The future of event processing will be interesting due to the increased capabilities of modern software and hardware systems. It is becoming possible to analyse and relate thousands of different streams in (almost) real-time, which will offer new insights into the complex interactions happening on the World Wide Web and the real world. An example where EP could go to in the future might be using event processing as an exo-cortex for human beings (Ehresmann et al., 2012) or for extending sensing capabilities in augmented reality environments.

In this chapter the history of event processing was presented along with different definitions of the concept "event". The next chapter introduces the basic principles and foundations of event processing are introduced that are required throughout the rest of the dissertation. It also picks up the previous given event definitions and provides a single definition that is used later on.

2.2 The event concept

In this section various definitions of the term *event* are discussed; based on this discussion an event definition will be derived that is used for the purpose of this dissertation and all related event processing contributions.

2.2.1 Different event definitions

Common definitions

The term event is of Latin origin and literally means "outcome". The online version of the *Merriam Webster* dictionary lists five main meanings, which include the notion of event as being the result of an action, as something that just happens, as a contest in a program of sport, as the geo-temporal definition of a point and finally as the potential outcome of an experiment (Merriam Webster Online, 2012). BusinessDictionary.com is more focused on the temporal aspect in its definition. It gives two descriptions:

> 1. Occurrence happening at a determinable time and place, with or without the participation of human agents. It may be a part of a chain of occurrences as an effect of a preceding occurrence and as the cause of a succeeding occurrence. 2. Arrival of a significant point in time. In project management, an event marks the point in time when a task is completed. (BusinessDictionary.com, 2012)

Event definitions in IT

Besides those general purpose definitions, the term event also has various different meanings within the domains of information technology and computer sciences. Table 2.1 provides a brief overview.

Area	Definition
Discrete event simulation	An event is a state change that "moves the system from one state to another instantaneously." (Preiss, 2000, p. 383)
GUI programming	An event represents an interaction that is triggered by a user, the user interface or the used multimedia application itself. These events are recognized and processed by the application and guide the interaction of the user with the application. (Schneider & Werner, 2001, p. 411)
Computer networking	An event is the reception or transmission of a network packet (D. Luckham, 2008, p. 3)
Topic Detection & Tracking	"A TDT event is defined as a particular thing that happens at a specific time and place, along with all necessary preconditions and unavoidable consequences. A TDT event might be a particular plane crash, or a single meeting, or a particular court hearing. " (Linguistic Data Consortium, 2004, p. 4)

Table 2.1: Overview of different meanings of the term *event*

All definitions have in common that an event represents something that has happened virtually or in the real world. Additionally the occurrence of an event its temporally determined, i.e. it is possible to state the time when an event happened. Finally, an event marks a change. In DES systems it is a state change; in GUI programming it is a change in the representation of an interaction artefact. These properties will now be used to derive an event definition for this dissertation.

2.2.2 Event definition

This section provides a definition for the concept *event* as it will be used within this dissertation. This will also include four main characteristics that an event must have in order to be properly processable. The event definition for this dissertation and its properties will be derived from popular event definitions in the field of event processing, and from the properties that were discussed in the previous section.

Definitions by Etzion and Luckham

In (Etzion & Niblett, 2011, p. 4) an event in terms of event processing is defined as

> ...an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain. The word event is also used to mean a programming entity that represents such an occurrence in a computing system.

This definition contains that an event is domain-specific and that the concept can be overloaded to represent two things: the occurrence itself and its digital representation. A similar definition is given in the *Event Processing Technical Society* (EPTS) glossary, where an event is also referred to as being "[a]n object that represents, encodes, or records an event, generally for the purpose of computer processing" (D. Luckham & Schulte, 2011, p. 5). (Chandy & Schulte, 2009) are even more unspecific and define an event simply as "something that happens" (preface, paragraph 4). These definitions illustrate how difficult it is to arrive at a common definition for the concept *event* in the area of event processing, because even domain experts cannot agree on a common definition.

Fortunately, (D. Luckham, 2002, p. 88) gives a more concrete definition of the term event and provides a list of three required properties:

- *Form:* the form of an event is an object. It may have particular attributes or data components. A form can be something as

simple as a string or more often a tuple of data components. ...

- *Significance:* An event signifies an activity. We call this activity the significance of the event.. . .
- *Relativity:* An activity is related to other activities by time, causality, and aggregations. ...The relationships between and event and other events are together called its relativity. ...

Luckham's definition is more detailed and provides three important aspects of the concept event. First, it has a certain "form", i.e. it contains properties that define the content of the event. Figure 2.2 shows what kind of properties these could be. Second, it has a certain "significance", i.e. something relevant has happened at a certain point in time. And third, an event is related to other events and its form can be used to reconstruct this relationship that is called "relativity".

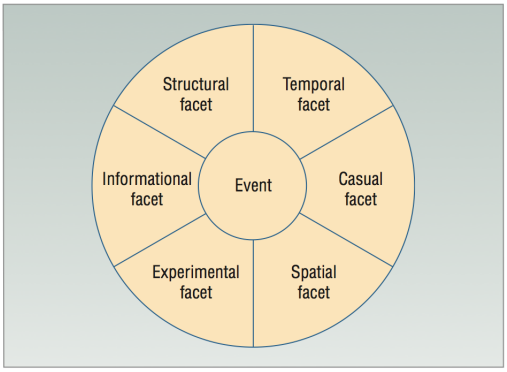

Figure 2.2: Different facets of an event (Jain, 2008, p. 5)

Used event definition

All three characteristics provide a sound basis for the definition of the concept *event* that will be used for this dissertation. As there is no official and standardized definition of the term event, the following four characteristics will be used to define an event as it will be used throughout this dissertation:

1. An event is temporally unambiguous.
2. An event represents a single semantic concept depending on the context it is used in.
3. An event is final, i.e. it cannot be changed after its occurrence.
4. An event can be referenced.

These properties are now discussed in more detail.

2.2.2.1 Temporal unambiguousness

Temporal unambiguousness

An event has to have an exact temporal dimension within a given context. This means an event can have several timestamps attributed, e.g. the detection timestamp at the sensor and the entry timestamp into the event processing network. But each timestamp must be unique within its context, i.e. there must be only one detection timestamp provided by the sensor context and a unique entry timestamp from the event processing network. This property also includes an *expiry* timestamp, which define that validity range of an event. An event is a time-sensitive entity, therefore this property must be strictly obeyed.

2.2.2.2 Semantic unambiguousness

Even types can be organised in hierarchies or other organisational schemes, in order to highlight their relationship, but each event in this scheme must have a well-defined and atomic semantic meaning. For instance there must not be a *Word Event* and a *Term Event* that both represent a raw and unprocessed textual entity, because this would impede a proper and unambiguous analysis of the subject matter as both event types denote the same thing but with different names. In general such semantic clearness is a cornerstone of a thorough system design.

2.2.2.3 Finality

As in the real world, the past cannot be changed. This is also true for events in the context of event processing. An event that has been created, stays as it is and cannot be changed[8]. A violation of this property introduces a kind of uncertainty and indecisiveness that would impede any sensible evaluation and analysis of the events within the event processing system, because it would not be possible to determine a correct and foremost current state of the system at any time.

2.2.2.4 Referenceability

Up to now each event has been defined to be final, and temporally and semantically unambiguous. But neither of these properties makes it unique within the context of an event processing system, because there can still occur situations where all three properties are equal for two events. Therefore

[8] A change of the information that is carried by the event would e.g. require some kind of "amendment event" that revises the original event by masking the original contents.

an event requires an id that makes it globally unique and thereby referenceable. Referenceability is required for various reasons. First, as just defined in order to refer exactly to a certain event and second, in order to reconstruct the context from which other events, based on the initial event, were derived. For instance a document can be re-constructed by collecting all the *term events* that have the same *document id* from the *document event* attached as property, from which they were derived.

2.2.3 Event types

Event types

Event processing systems use different event types that are based on the previously introduced event concept. (Etzion & Niblett, 2011, p. 62) defines an event type as "...a specification for a set of event objects that have the same semantic intent and same structure; event object is considered to be an instance of an event type". In context of event processing the Event Processing Technical Society (D. Luckham & Schulte, 2011, p. 4ff) differentiates between the following types of events as relevant for event processing systems.

- **Raw event**: The unprocessed form of the incident that should be processed in the EP system, this can be an event from a physical sensor or the raw unprocessed document e.g. a Tweet.
- **Derived event**: An event whose existence is due to the processing of another event.
- **Complex event**: "An event that summarizes, represents, or denotes a set of other events" (D. Luckham & Schulte, 2011, p. 17).
- **Simple event**: A complement to complex events, i.e. everything.
- **Virtual event**: An event that is created within the event processing system without any reference to a real word, physical event. Can be considered similar to a database view.
- **Composite event**: An event comprises other simple or complex events.
- **Instantaneous event**: An event where the start and the end is identical.

This categorisation is used in this dissertation, because it was derived by renowned researchers in the field of event processing and because no other appropriate event type categorisations exists. It provides a concise and understandable description of fundamental event concepts and can therefore be used as basic principles when it comes to the design of the reference model in Chapter 4.

Figure 2.3 shows the relationship of the various even types.

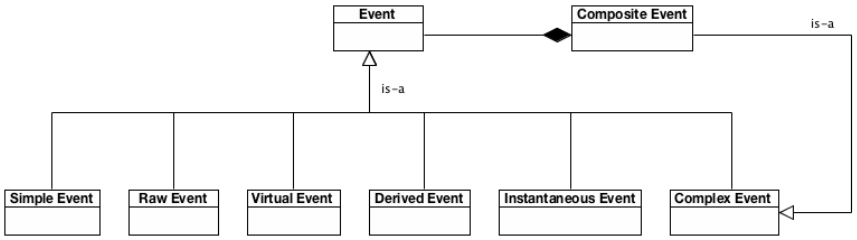

Figure 2.3: Event types according to (D. Luckham & Schulte, 2011, p. 4ff)

The introduction of these event types is important, because event driven systems have to deal with different types of events on different levels of abstraction. Furthermore, some event types are also used in the event reference model that is introduced in the next chapter.

2.3 Principles and building blocks of event processing systems

This section explains the main principles of event processing, followed by the main building blocks of an event processing system. This domain knowledge helps to understand the ideas presented in Chapter 4 and Chapter 5.

2.3.1 Basic architectural principles

(Chandy & Schulte, 2009, p. 10) five essential design and processing principles of event-driven systems are described that also form the basis for the investigations in this dissertation:

1. **Individuality** Every events is atomic and emitted individually by an event sender.
2. **Push** The event source emits, i.e. pushes, the events to the sources.
3. **Immediacy** The event consumer processes the incoming event immediately.
4. **One-way** An event-driven system works in a *fire-and-forget* mode, i.e. the event source does not wait or expected any response from the consumers of the events.
5. **Free of commands** An event represents only *that* something has happened and the event consumer can do something with it. The event does not prescribe *how* the consumer reacts to the message.

2.3.2 Event stream processing and event clouds

The title of this dissertation contains both *text streams* and *event processing*. While both topics seem closely related, there are essential differences between both that will be highlighted in this chapter, in order to keep a clear understanding of the topics. The main difference is the subject matter they operate on. As the name implies event stream processing is operating on event streams. In (Kotov et al., 2011) a definition of text streams is given which can be used to describe event streams in general.

Definition 1. *An event stream S of length M represents a temporally ordered sequence of events* $D_1, D_2, \ldots, D_M$ *over T discrete, non-overlapping time intervals* $1, 2, \ldots, T$*, where each event has an associated time stamp.*

The important aspect is the sequential and chronological order of the data entities. Event stream processing is only concerned with processing data that arrives in a well-defined order. In contrast event processing can be seen as a superclass of stream processing, because it not only covers the operation on sequential data, but also allows the operation on *partially ordered sets of events* that form a so-called *event cloud* (cf. (Vincent, 2009), (D. Luckham, 2006)). An event cloud contains many different streams whose order of arrival is not clear or predefined and therefore it is not strictly ordered like in event stream processing.

In this dissertation the text streams were strictly processed in an event stream processing manner, i.e. pattern matching or temporal logic were used. Nevertheless the subtitle "an event processing based approach" of the dissertation is still valid, as the event reference model as well as the reference architecture cannot only be used on event streams, but also for event clouds.

Finally, a note on *stream processing*. This term is also closely related, but refers to a much broader area of data streams, like video or audio streams, that are not subject to event stream processing and therefore not covered in this thesis (Etzion & Niblett, 2011, p. 21).

2.3.3 Windowing mechanisms

Windowing mechanisms

In event processing and in data stream processing windowing mechanisms are an essential concept. The idea is to face the immense volume of incoming data by only looking at the k most recent data points.

The size k can be defined by using different types of windows. There are two main types of windowing mechanisms, *count-based* and *time-based* sliding

windows. The first only keeps a fixed amount of k elements in the window, while the second keeps all events in the window that occurred within the last k seconds.

These windowing mechanisms work internally in two modes: *tumbling* and *sliding* windows. The first one works like a *micro batch*, i.e. it accumulates k events or events for k seconds, then discards all events from the window when k is reached and starts gathering events again. In contrasts, the second one in con works like a fixed sized jig on the data stream, i.e. old events leave the window gradually while new events flow into the window.

One is a *count-based* sliding time window which keeps the n-last events within the window. The other is a *time*-based approach which keeps the events that occurred within the last n seconds (Mouratidis & Pang, 2011, p. 1).

2.3.4 Pattern matching and recognition

Pattern matching is another strength of event processing[9]. Initially, this concept was introduced for finding patterns in text. (Crochemore & Lecroq, 1997, chapter 13.1) defined pattern matching as "... the problem of locating a specific pattern inside raw data. The pattern is usually a collection of strings described in some formal language".

Translated into the area of event processing, a pattern can be considered as sequence a (partially ordered) sequence of events that are related using some logical statements. These statements can be based on temporal logic (cf. next section) or propositional logic or other types of logic. In fact, the goal is express the problem of identifying a relevant and interesting constellation of events from the data stream. This capability is one of the major drivers for the integration of event processing (engines) into data stream analysis systems.

Pattern types

There exist two categories of patterns in event processing. *Analytical patterns* and *heuristic patterns*. This dissertation defines the former group as a representation of patterns that are generated using some algorithm or data mining approach, i.e. the pattern creation process is executed autonomously by the systems. An example for this approach can be found in (Ma & Hellerstein, 2001).

The second group is defined as patterns that are based on a *heuristic*. A heuristic, according to Judea Pearl (cited after (Dechter et al., 2010, p. 547f)), is a "criteria, methods, or principles for deciding which among several alter-

[9] This section is not about patterns as introduced by (Alexander et al., 1977, p. X). That kind of patterns is used to provide a structural template to avoid "re-inventing the wheel" in software development or software architecture.

native courses of action promises to be the most effective in order to achieve some goal". Heuristics and thus heuristic patterns are created by consulting "'simplified models of the problem domain' particularly those 'generated by removing constraints which forbid or penalize certain moves in the original problem [Pearl 1984, p. 115]"' (Dechter et al., 2010, p. 548). In contrast to analytical patterns these types of patterns are derived by domain experts, i.e. the pattern creation process is executed by a human being. Such patterns are formulated as *standing queries*, i.e. the pattern is defined upfront and it operates on the event stream and the event cloud while events keep arriving.

These patterns concepts are relevant, because, first, there will be special event types introduced in the reference model that are used in this context and, second, in chapter 7 different types of filter policies will be studied that fall into these two categories.

2.3.5 Temporal logic

A considerable benefit of using event processing engines like e.g. *Esper*[10] is that many of them offer an implementation of *Allen's Interval algebra* (Allen, 1983). The goal was to express temporal knowledge and to allow temporal reasoning on data. Therefore Allen introduced *temporal intervals* as temporal representation primitives and provided "a method of representing the relationships between [these] temporal intervals in a hierarchical manner ..." (Allen, 1983, p. 832).

As there are no patterns in this dissertation that use temporal logic, further details on the usage of temporal logic can be found in (Allen, 1983). In (JBoss, 2015) examples are shown of how Allen's interval algebra can be used in event processing.

2.3.6 Stream fusion

Stream fusion

Another relevant event processing concept is *stream fusion* or *data fusion*. (F. E. White, 1991, p. 5) defines data fusion as "[a] process dealing with the association, correlation and combination of data and information from single and multiple sources ...". This concept is important, because only the merging of various event sources allows generating new and insightful information that a single data stream would not provide if it was studied separately.

In this dissertation multiple event streams are used and joined, in order to calculate measures for term weights, scores for documents or generate filter

[10] `http://espertech.com`

policies. These tasks can only be completed if data streams can be joined using some means of technology. And this capability is provided by event processing framework that represent technological manifestation of the here discussed principles (cf. chapter 2.4.1).

2.3.7 Decoupling and push-style

The final two principles of event processing systems are decoupling and push-style data exchange. The first aspect stresses the idea that consumer, producer and event processing agents[11] should work independently without any mutual dependencies (Etzion & Niblett, 2011, p. 34). Thereby it is possible that consumers or event processing agents receive events from various producers and producers can emit several event types without knowing how, when or where the events are consumed. (Etzion & Niblett, 2011, p. 34) called this the *principle of decoupling*.

Decoupling is also a guiding principles for event processing systems, because in settings with high data volume and velocity aspects of "flexibility, scalability, and fault tolerance in the communication infrastructure" (Rozsnyai et al., 2007, p. 64) can be better addressed in decoupled systems, as each component can be tuned, enhanced, replaced or extended without affecting other components.

A vocabulary is required

A consequence of this decoupling is the need for a standardized "vocabulary" to exchange events. The basic of event types were presented in (Rozsnyai et al., 2007) and (Etzion & Niblett, 2011, p. 61ff) and led to the necessity of an event reference model for text stream analysis that is presented in Chapter 4.

Decoupling can only be achieved if messages are exchange in a push-style manner, i.e. event producers and event processing agents emit their processing results without being asked to do so from another component. Any other exchange style, such as *request-response* for example, would impose an unwanted coupling of components.

2.3.8 Building blocks

Event processing systems consist of several building blocks that are essential when building such types of systems. (Etzion & Niblett, 2011, p. 50) defined the seven components each event processing system consists of:

[11] Cf. next section for more details on building blocks.

- Event producer
- Event consumer
- Event channel
- Event types
- Event processing agent
- Context
- Global state

This enumeration of components is a sound framework, as it provides a concise overview of concepts that were discussed by (D. Luckham, 2002), (Michelson, 2006), (D. L. Hall & Llinas, 1997) and (Berry et al., 2011).

This section discusses characteristics of those components, as in this dissertation bases all data processing on events and data streams and these building blocks are the foundation for the systems, which operate in that way. Furthermore, this is valuable domain knowledge, in order to better understand the reference architecture is presented in Chapter 5.

Event Producer

Event Producer In order to be able to operate on events, the events first have to be made available to the event processing systems. This task is done by an *event producer*. An event producer detects and receives events outside of the event processing network (EPN), transforms them into event types processable by the event processing network and emits them in to the EPN. (Etzion & Niblett, 2011, p. 42) defines an event producer as " an entity at the edge of an event processing system that introduces events into the system". Within an *even processing*(EP) system there can be several event producers, which can either emit the same or different types of events.

The instantiations of event producers can be manifold. Event producers can be sensors in hardware, e.g. a pressure sensor in a car, log file monitors that emit new log file events into a system or a business workflow that emits an event after reaching a certain state within the workflow.

Another feature is that event producers work in an push-style. This means events are sent without the explicit request of other components within the event processing network. Event producers emit events as one-way messages, i.e. they do not expect any response from a consuming component[12]. This is an important feature as it facilitates a loosely coupled processing style which is a characteristic of EP systems[13].

[12] It is, of course, possible to integrate message acknowledgement into the event processing system if it is required.

[13] (Etzion & Niblett, 2011, p. 34ff)

The EPTS provides a reference architecture for event processing systems. Figure 2.4 shows the logical view of the reference architecture, which sums up the just presented concept for event producers.

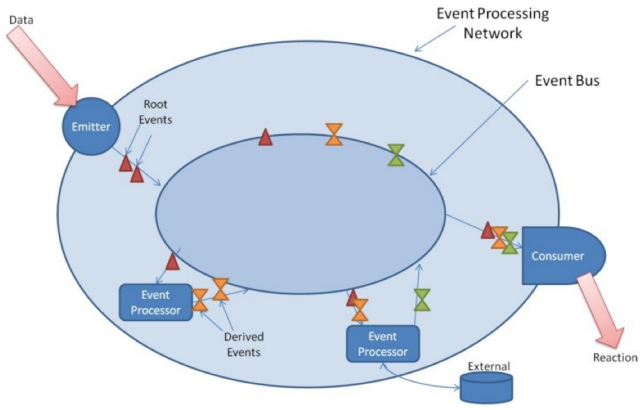

Figure 2.4: Logical view on the EPTS reference architectures (Paschke, Vincent, Alves, & Moxey, 2012, p. 325)

Event consumer

Event Consumer Event consumers receive events from the event stream. They are the final sink within the processing of an event. (Etzion & Niblett, 2011, p 42) defines an event consumer as "[a]n event consumer is an entity at the edge of an event processing system that receives events from the [event processing] system". This means an event producer is the interface to systems outside of the event processing system. Thus, it corresponds to the function of event producers. There can be several event sinks within an event processing system, e.g. a log sink that writes events in an external data store or a dashboard sink that visualizes events in a graphical manner, etc.

Event channel

Event Channel Event channels are used to transfer the events within the event processing network. They are used to move the events from a producer to a consumer, from a producer to an event processing agent or from an event processing agent to a consumer or a sink. Event channels can receive events from one or various sources, make decisions on where to route an incoming event and forward the event unchanged to its destination (Etzion & Niblett, 2011, p. 134f). Their purpose is to ensure that events make it from one component to the other within the event processing network. This is usually done by using a message broker such as *RabbitMQ*[14], *Apache Kafka*[15] or ØMQ[16] that scale up to several million events per second depending on

[14] `https://www.rabbitmq.com/`
[15] `https://kafka.apache.org/`
[16] `http://zeromq.org/`

the used infrastructure and setup.

Event Processing Agents Another central component of an event processing system is the *event processing agent*(EPA). It takes over important functions like event filtering, event transformation validation, enrichment or formatting. Because an EPA executes more sophisticated actions within an EPN, its background and basic principles are explained in more detail.

Agents in computer science

Agents in computer science

First, a short overview of the concept *agent* is provided, in order to distinguish event progressing agents from other types of agents in computer science.

The term agent in computer science can have different meanings, because it is used in different research fields such as artificial intelligence, robotics or information systems. Hence there are different definitions for this term. (Russell & Norvig, 2010) define agents as "...anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" (Russell & Norvig, 2010, p.31). This is a generic definition, but already provides a good idea of the concept.

In (Nwana, 1996, p.8) a detailed overview of different types and definitions of agents are given. In fact, they identified seven types of agents.

- Collaborative agents: emphasise autonomy and cooperation
- Interface agents: "emphasise autonomy and learning in order to perform tasks for their owners"
- Mobile agents are considered to be personal digital assistant (nowadays smartphones)
- Information/internet agents "perform the role of managing, manipulating or collating information from many distributed source"
- Reactive agents: "...they act/respond in a stimulus-response manner to the present state of the environment in which they are embedded"
- Hybrid agents: refers to an integrated setup of at least two or more agents which belong to two or more different agent classes
- Smart agents: type of "future" agents that anticipate the users' needs and autonomously support their actions

The *Actor Model*

Actor Model

Also closely related to agents is the concept of *actors* that was introduced

by (Hewitt et al., 1973). The idea is to consider "'Actors' as the universal primitives of digital computation" (Hewitt, 2010, p. 1). The *Actor Model* that is based on this idea intends to be a universal model for the implementation of concurrent systems. As in event based systems, many events are being processed in parallel. Hence this model is relevant, because it introduces several concepts that can be used for the design and implementation of event processing systems[17]. Among others, these are according to (Hewitt, 2010)

1. "When an actor receives a message, it can concurrently send messages to (unforgeable) addresses of Actors, create new Actors, designate how to handle the next message it receives" (p. 3).
2. "An Actor can only communicate with another Actor to which it has an address" (p. 3).
3. The Actor Model also forces the decoupling of sender and receivers and proliferates the use of asynchronous communication (p. 3).
4. The Actor Model fosters security by only being able to send messages to other Actors it knows (p. 6).

Event processing agents in detail

Event processing agents fit best into the category of "information/internet agents". They receive events from one or various sources, perform different operations on the incoming events and then emit them as new event streams for further processing. In this sense an EPA works like a hybrid component composed of an event producer and an event consumer.

Structure of event processing agents

The structure of an event processing agents was described in detailed by (Cristea et al., 2011, p. 36). An event processing agent consists of an *event input adapter* that listens for relevant events. When it receives a relevant event, it transforms it into the required format that the internal logic of the event processing agents requires and hands it over to the event processing logic.

The event processing logic can be implemented using custom code and / or using a *domain-specific language* (DSL) of the event processing engine that is used. In general custom code should only be used for purposes that are not covered by the DSL of an event processing engine. The reason for this is that event processing engines are optimized for fast processing of large amounts of stream data[18]. They offer specialized capabilities e.g. for temporal logic, windowing mechanisms, pattern matching and stream fusion.

[17] The basic principle were also used for the definition of the reference architecture and the derived patterns (cf. 5.2.2.3)

[18] Note: These capabilities were the main reason for combining text streams and event processing and thus the impulse for this dissertation.

In addition, an EPA can have a *rule base* and a *knowledge* base, which represent the intrinsic knowledge of the EPA about the world. New events update both and actions are trigger based on the results from the rule base and the knowledge base.

After the implemented logic is executed, the *event output adapter* transforms the results into the target event type of the event processing agent. Figure 2.5 visualizes the structure of an event processing agent.

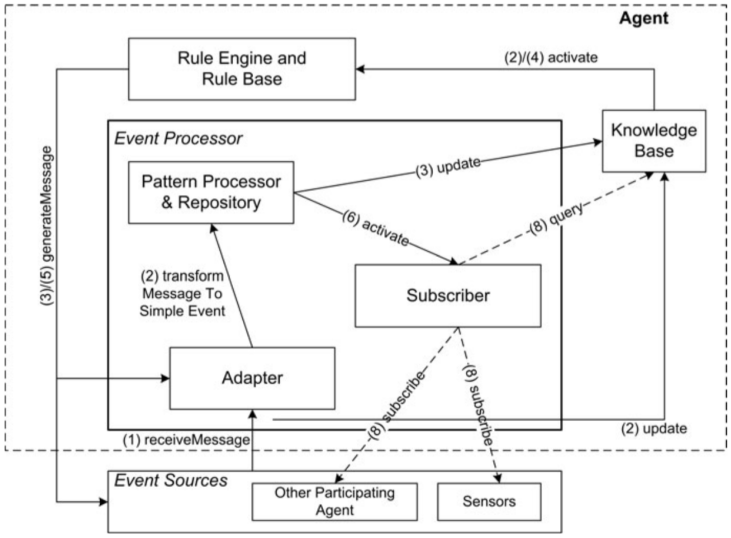

Figure 2.5: Structure of an event processing agent according to (Cristea et al., 2011, p. 36).

Depending on their purpose, event processing agents consume and emit different types of events. There are usually a multitude of different event processing agents within an event processing system. Hence an event reference model, like the one that will be proposed in the next chapter for text stream analysis, is useful during the design and implementation of an event processing, in order to provide a structured and common *event vocabulary* during the design, implementation and evaluation phase.

Types of EPAs

There are also different types of event processing agents. (Etzion & Niblett, 2011, p. 52) provide an overview of different types of event processing agents. The main three types are:

- Filter event progressing agents remove irrelevant or uninteresting events from the event stream, in order to facilitate further processing
- Transformation event progressing agents change the internal structure of events and derive new events. The main types are:
  - Translation agents take each single event and transform each event into a new one.

  - Split agents take a single incoming event and split this event into several new events, e.g. a sentence is split up into various tokens.
  - Aggregation agents receive several input events and aggregate them into a new single event, e.g. counting the occurrences of an event and emitting a single count event.
  - Composition agents receive several different events from the stream and creates a new event by selecting and combining attributes from the different events into a new event.
- Pattern detection event progressing agents receive events and look for patterns of event occurrences or absence, e.g. modelling a fraud event using the same credit card within 10 minutes from two different automated teller machines in two different countries.

Figure 2.6 shows the relationship visually.

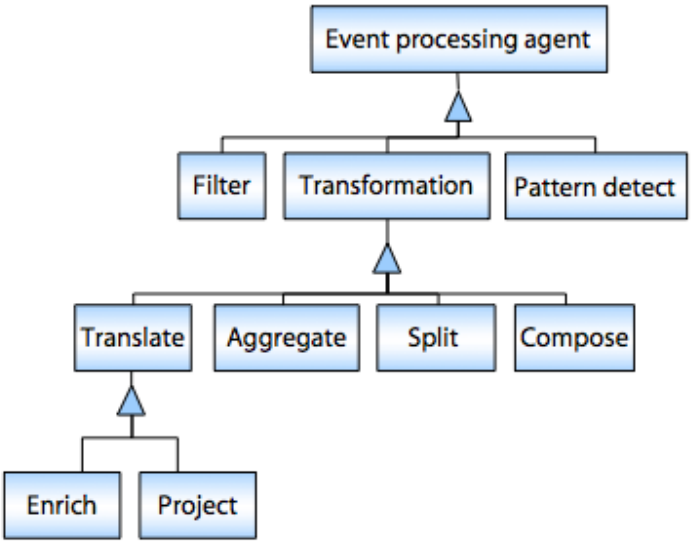

Figure 2.6: Different types of event processing agents by (Etzion & Niblett, 2011, p. 52).

Event processing agents are fundamental to this dissertation as they encapsulate all the logic that is required to execute the experiments.

Event processing networks When the aforementioned building blocks are combined, they start to form an *event processing network* (EPN). The EPN is *the* central component, because it unifies and connects the various buildings blocks to represent the purpose of the event based system. The source, the sinks and the event processing agents are composed, in order to form a communication network that represents the goal of the event processing system

(D. Luckham, 2002, p. 208f). The design and implementation of EPNs is a challenging tasks that is intended to be alleviated by the introduction of a reference architecture in Chapter 5. Figure 2.7 shows an example structure of an event processing network.

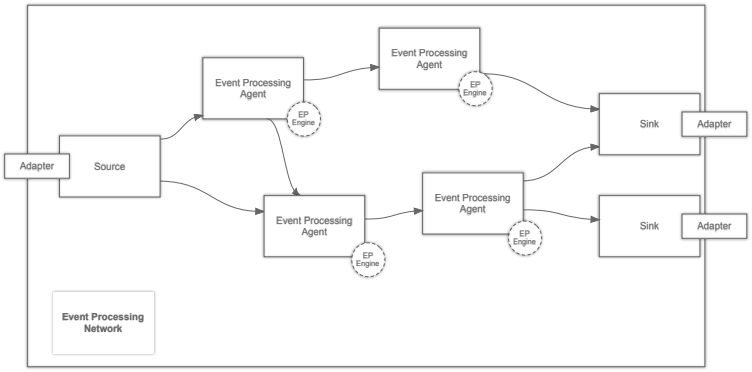

Figure 2.7: Sample structure of an event processing network.

2.4 Big Data technologies

The basic principles of event processing have been introduced. Now it is necessary to put them into the context of *Big Data*, because this term has been *the* prominent concept in terms of rapidly processing large data volumes for several years. Thus their relationship has to be clarified.

Big Data has been one of the dominating IT topics in recent years. This buzzword is rather broad and lacks a standardized definition, but taking various definitions (cf. (Bitkom, 2012, p. 7), (Zikopoulos et al., 2012, p. 4f), (Boyd & Crawford, 2012, p. 663f)) into account *Big Data* can be briefly summarized as the promise to process "incredibly huge and fast" volumes of data and extracting unprecedented quantities of "wisdom" from them.

The term is controversial, because the adjective *big* is always dependant on the time it is used. Big Data in 2005 would not be considered big in 2015. Therefore the term is always subjective and leaves space for discussion. Also it is not clearly defined which types of technologies are genuine Big Data technologies and which existing technologies were re-branded by sales representatives of various technology vendors. Thus, there is no current and final scope of this term and probably never will be. In summary, the discussion about Big Data was depicted by (Zikopoulos et al., 2012, p. XVII) as follows:

> While the term Big Data might be almost universally unloved, it is also now almost universally understood to refer to the realization of greater business intelligence by storing, processing, and analyzing[sic] data that was previously ignored due to the limitations of traditional data management technologies.

5 Vs of Big Data

Despite the controversial views on Big Data, this dissertation is concerned with the processing of Big Data, if, for the purpose of this dissertation, Big Data is defined based on the *5 Vs*: velocity, volume, variety, veracity and value (Zikopoulos et al., 2012, p. XVIIf). This means, one is processing Big Data if these five properties are "big(er)" relative to what has been studied in this area earlier. In terms of velocity, volume and variety the rise of social media content that is generated by Twitter, Facebook or Instagram for instance is definitely larger than the social media data that were available in the past for information filtering research.

The term veracity "highlights the fact that tried and tested data management principles – data quality, data cleansing, master data management, and data governance – are not completely redundant in a world of Big Data" (Zikopoulos et al., 2012, p. XVIII). These requirements are addressed by the introduction of an event reference model that should facilitate their implementation. And finally all the evaluation and algorithms should contribute a certain *value* to the processing of highvelocity (thus Big Data) short document text streams.

Nevertheless the goal of this section is to show the variety of different approaches at the time this dissertation was written, to further strengthen the difference between event processing and other *Big Data* technologies and to elaborate why event processing was chosen.

2.4.1 Stream processing and event processing engines

As previously described, stream processing and event processing are different concepts. Hence there are also different implementations of these concepts. In this section, notable implementations of event processing and stream processing engines are introduced, in order to provide an overview of the choice of technology stacks that exist at the time that this dissertation is written.

The goal is also to present a selection of technologies that can be used to reimplement the algorithms used in this dissertation using the event reference model and reference architecture that are presented in Chapter 4 and 5.

For this dissertation the event processing engine *Esper*[19] was used. The reason was that Esper is a free available event processing engine that is written in Java and offered a variety of temporal window capabilities when the thesis was started in 2009. As all of the non-event processing code was supposed to be developed in Java, it was a natural choice to use Esper. The pattern matching and temporal logic capabilities offered by Esper where not used. Thus Esper was used in a pure stream processing way. Other event processing engines that offer the same event processing capabilities are *Drools Fusion*[20], *IBM ODM*[21] or *SAP Sybase Event Stream Processor*[22].

2.4.2 Map/Reduce

A major enabler for the adaptation and the spread of *Big Data* was the invention of *Map/Reduce*. It allowed parallelising the processing of large volumes of data in huge server farms built from commodity hardware. This helps tremendously in reducing the cost of setting up and running server farms for analysis purposes (J. Dean & Ghemawat, 2008, p. 1).

Map/Reduce is a central tool when it comes to the analysis of Big Data. Foremost Apache Hadoop is the most frequently used Map/Reduce implementation aid at the moment. It also is the foundation of many technology companies focused on *Big Data* like Hortonworks. As Map/Reduce is rather a framework than a tool, it is flexible and thus open to different types of analysis (Miner & Shook, 2012, p. 1). This flexibility allows implementing large-scaled analysis systems for different purposes. This entails network analysis as well as machine learning scenarios.

For this dissertation Map/Reduce is not relevant as it is batch-oriented. Furthermore it is best at addressing large volumes of data and not continuously flowing data, whereas this dissertation is focus on the continuous and real-time analysis of incoming data.

Nonetheless, Map/Reduce is also relevant in a streaming context. In mid of 2013 Twitter released a hybrid framework that combines Hadoop and the stream processing engine Storm called SummingBird. The goal is to provide real-time analytics for data that needs to be analysed immediately, and batch-oriented large-scale data processing for later analysis using Hadoop.

[19] `http://espertech.com`

[20] `http://www.drools.org/`

[21] `http://www-03.ibm.com/software/products/en/category/operational-decision-management`

[22] `http://www.sap.com/germany/pc/tech/database/software/sybase-complex-event-processing/index.html`

This framework supports the notion that Big Data allows manifold approaches to address data processing problems. It also underpins that event processing is gaining more interest and that Map/Reduce is not suited for streaming data alone. This type of hybrid architecture is now also known as "Lambda Architecture". It was proposed by Nathan Marz (Hausenblas & Bijnens, 2015) with the goal to provide an architecture that provides can address real-time requirements as well as large data processing requirements within one architectural style.

2.4.3 NoSQL

NoSQL, is another concept that is closely related to *Big Data* and modern data processing. In the beginning when the term first came up "No" really meant "No", i.e. SQL should be completely avoided and instead it should be replaced by new types of databases. After some time it became apparent that the decades of research in relational database management systems have not outdated traditional relational database management systems and thus the term has changed into "Not only" SQL meaning that new types of data stores should be used in situations when adequate, and be combined with traditional RDBMs when necessary. (Cattell, 2011, p. 14) groups NoSQL data stores into four groups:

- Key-value stores: Data items are stored and retrieved using a unique key
- Document stores: Schema-free data stores that save a complete document
- Extensible record stores: Uses rows and columns like traditional RDBMs, but provides scalability by distributing both over various nodes
- Scalable relational systems: *Traditional* RDBMs but able to adapted for scaling over various nodes

(Cattell, 2011) also provides also further details on each type of data store including a list of products.

In context of this dissertation only document stores are relevant for the reference architecture presented later on. This type of data store is often used in scenarios where one has to deal with un- or semi-structured such as text. Within an event processing network a raw document is not available as an attribute of an event, because it adds too much overhead due to its size and is not needed for processing; an event processing agent only operates on a set of event types and not on a raw document. Nevertheless, the document

itself still is necessary when results are presented to the user or when parts of the document should be analysed further. Then the required document can be easily restored from such a document store.

2.4.4 Big Data ecosystems

All the aforementioned technologies also require platform ecosystems, in which they can reside, coexist, interact and foremost be scaled to address the data loads. Such platforms act as ties between the technologies, because each of them addresses a special aspect in Big Data processing, thus different technologies need to be able to cooperate.

The case for ecosystems

For instance, the data streams that are analysed by the stream processor need to be stored and archived, so they are available for a retrospective analysis. There, messages infrastructures based on e.g. Apache Kafka or ØMQ need to be available that move the data to a scalable data storage that is e.g. based on *Apache Hadoop*. In order to provide fast access to the stored data, in-memory databases like *memcache* can be used to serve queries quickly. Finally, an overall configuration management is required that allows to configure and administer the ecosystem. This could be e.g. based on *Apache ZooKeeper*.

Through this example it becomes obvious that only the setup of such an ecosystem can be laborious and complex. Therefore, there are several providers on the market that provide off-the-shelf, integrated technology stacks that alleviate the start into Big Data processing. Such companies are for instance *Hortonworks*[23], *Teradata*[24], Databricks[25] or IBM and SAP.

2.5 Real-time processing explained

Topics like large data volumes, data streams and the necessity of effective and fast data processing have been already stressed throughout this chapter. In this context the term *real-time* has already been mentioned several times. Because this term will be used often in the context of event stream and event processing and because it will also be use throughout in this dissertation, the different notions of this term will be explained briefly.

It has been shown that popular Hadoop based Big Data approaches are usually not real-time due to their inherent properties like fixed schedules and

[23] `http://hortonworks.com`
[24] `http://www.teradata.de/`
[25] `https://databricks.com/`

batch-oriented processing and that event processing in contrast supports real-time. But as there are different notions of real-time in computer science, it is necessary to provide the prevailing definition for this dissertation. Thus, the proposed implementation that is used in this dissertation can be considered as a system that processes incoming data "without perceivable delay" (Wikipedia, 2015h).

In terms of definitions used in computer science, this definition is close to "soft" real-time. This means that a few or slight violations of temporal limits are tolerated, whereby the system performance is not significantly affected. Examples are flight booking systems or call centre applications, which tolerate slight violations of temporal constraints.

In contrast a system – hardware or software – is "hard" real-time, when data processing must guarantee the execution of processing steps within a fixed defined time limit and must not exceed this limit at any time and under any circumstances. Hard real-time systems are often flight surveillance systems, medical monitoring systems or automotive control systems like air bags. These systems must not exceed a defined reaction limit otherwise severe damage can occur (see Schneider & Werner, 2001, p. 267).

Real-time analysis has advantages and disadvantages. One advantage is to process and analyse data as it is being generated and provide immediate feedback and results to the user. This is useful if the user has to make decisions based on information that is volatile, which quickly loses relevance and requires fast reaction. As information production speeds up daily, real-time analytics are of major interest.

Human interaction

One big disadvantage is that a thorough analysis cannot (yet) be done in real-time, i.e. you still need an analyst who correlates the information provided by the analysis system and draws conclusions from that. This process involves thorough considerations and discussions with other analysts or process partners. No major company will (probably) take important business decisions on an ad-hoc analysis without a thorough check of the results or without having defined rigid rules of trusting automated system decisions.

While real-time analysis offers great opportunities, it also contains risks. So reacting on data too rapidly or acting on information without knowing why is always fraught with some danger. Therefore this dissertation wants to address this aspect by providing insight into how text scoring and filtering works in high-volume text streams. This insight should help to judge high speed filtering systems and enhance their performance. The proposed system cannot exist on its own. Of course it should be incorporated into a *Big Data* stack that also offers facilities to analyse historical data, but on its

own it already provides interesting information on the operating mode of text stream properties.

2.6 Summary

In this chapter the basic principles and relevant concepts of event processing have been introduced. First, the history and evolution of event processing has been described. This was done to highlight the position of event processing as a solid and sound technology, whose application for the processing of high-velocity text stream is valid and adequate. Then the concept *event* was discussed in detail, and four necessary characteristics were stipulated: Temporal and semantic unambiguousness, finality and referenceability. Afterwards relevant *event types* were introduced as well as the main building blocks of event processing systems. In addition, relevant event processing concepts like windowing mechanisms and stream fusion were described. The chapter was concluded by a brief overview of related *Big Data* technologies, in order to delimit event processing from these technologies. In summary, the goal of this chapter has been to provide valuable domain knowledge that is useful to assess the contributions in terms of event reference model and reference architecture that are presented in Chapters 4 and 5. The next chapter introduces the basic principles of information retrieval and filtering that are relevant for this dissertation.

Chapter 3

Information filtering and information retrieval

This chapter introduces basic principles of information filtering and information retrieval that are relevant in the context of this thesis. The goal is to present the relevant[1] topics for event-driven text stream processing.

The structure of this chapter is as follows: Section 3.1 discusses the commonalities and differences between information filtering, information retrieval and text stream processing. The objective is to avoid any confusion of the three topics. The term information filtering is used when explicitly referring to the necessities and requirements of information filtering. In section 3.2 basic principles of information filtering are presented. This also includes the evolution and the current state of information filtering as well as the challenges posed by new high-velocity text stream sources. Section 3.3 introduces the relevant information retrieval concepts such as term weighting, scoring, query expansion and evaluation. The chapter is concluded with section 3.4 where related areas of research are discussed which are similar to the objectives of this dissertation, but nevertheless are not in scope.

3.1 Information filtering, retrieval and text stream processing

As mentioned, both topics are closely related. Nevertheless, the distinction between both areas is important, because

[1] This means only the topics that are in fact used in this dissertation are described. n consequences this also means that seminal topics like e.g. the *Vector Space Model* are left out if they are not used in this dissertation.

> [a]lthough many of the underlying issues are similar in retrieval and filtering, since they share the common goal of information delivery to information seekers, the design issues (e.g. timeliness of data, identification and representation of user needs) and also the techniques and algorithms devised to satisfy these information needs differ significantly. (Tryfonopoulos et al., 2009, p. 10:5)

In (Arampatzis, 2001, p. 4) a comprehensive overview of the main differences between information filtering and information retrieval is presented which is shown in table 3.1.

	Information retrieval	Information filtering
Information request	Short term	Long term
Information source	Static	Dynamic
Feedback	Usually one time	Repeatedly
Comparison	Ranking of documents	Binary decision

Table 3.1: Overview of differences between information filtering and information retrieval

As shown in figures 3.1 and 3.2, the general models of information filtering and information retrieval reflect the points from table 3.1 and summarise them visually (Belkin & Croft, 1992, p. 31).

To further illustrate the differences between information filtering and retrieval, (Oard, 1997, p. 6) provided a concise overview of different information seeking processes, which also intends to clarify the difference between information filtering and retrieval compared to related research areas (cf. table 3.2).

Process	Information Need	Information Source
Information filtering	Stable & specific	Dynamic & unstructured
Alerting	Stable & specific	Dynamic & Structured
Data Mining	Stable & specific	Stable
Information retrieval	Dynamic & specific	Stable & unstructured
Database Access	Dynamic & specific	Stable & structured
Exploration	Broad	Varied

Table 3.2: Overview of information seeking processes (Oard, 1997, p. 6)

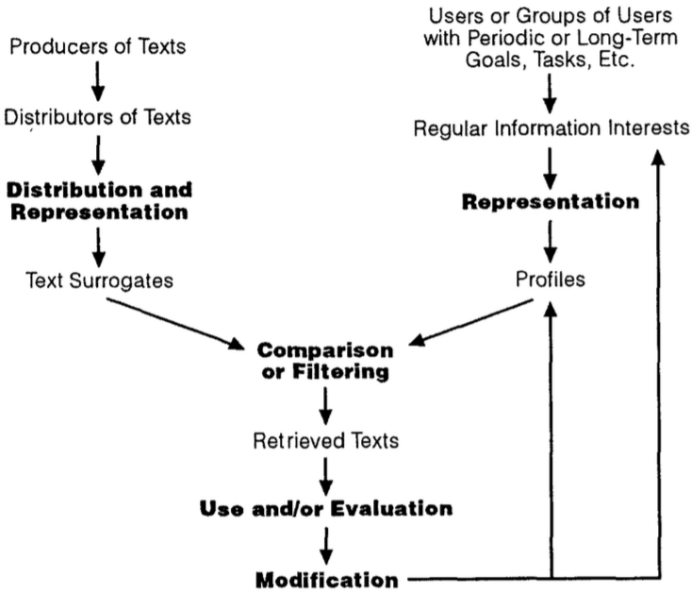

Figure 3.1: General mode of information filtering according to (Belkin & Croft, 1992, p. 31)

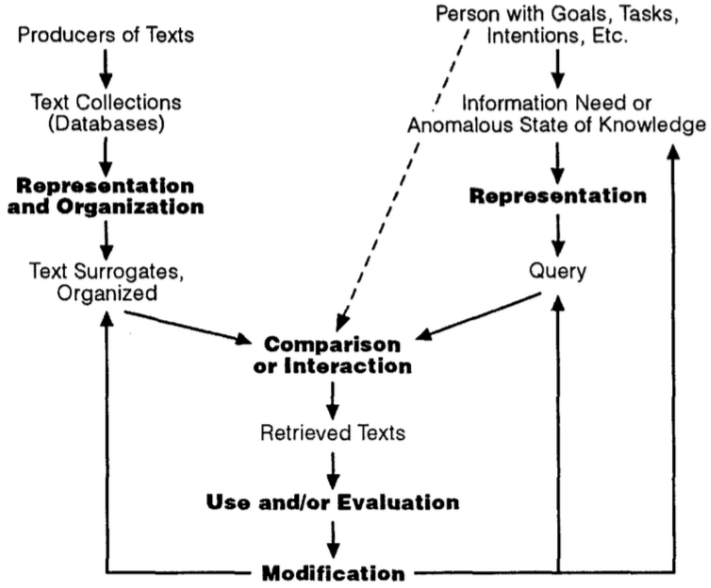

Figure 3.2: General model of information retrieval according to (Belkin & Croft, 1992, p. 31)

Definition information filtering

In summary, information filtering – as used in this dissertation – uses standing queries, in order to filter documents from a text stream and to satisfy a long-term and stable information need. The information need is encoded in *profiles* or *queries* that "are relatively specific ..., and that ... change relatively slowly with respect to the rate at which information sources become available" (Oard, 1997, p. 3).

According to the classification criteria of information filtering systems presented in (Hanani et al., 2001, p. 205ff), the system[2] used in this dissertation can be classified as outlined in table 3.3.

Criteria	Value
Initiative of operation	Passive IF system
Location of operation	At filtering server
Filtering approach	Cognitive (content-based)
Methods of acquiring knowledge on users	Explicit

Table 3.3: Classification of the StreamFI filtering systems

Definition information retrieval

In contrast, Information retrieval is an interactive process to find relevant documents in a rather stable text collection. The user issues a query, gets a list of potentially relevant documents, evaluates these documents, adjusts his initial query and repeats this process until his information need is satisfied or he gives up (Oard, 1997, p. 3).

In addition to information filtering and information retrieval, document management and text processing tasks are required. These are important, as they are responsible for making the textual content accessible for filtering and retrieval. Without these tasks it would be impossible to unlock the content contained in the documents. According to the *Association for Computing Machinery*(ACM) *Computing Classification System* (CCS) *document management and text processing* comprise four areas: document search, document management, document capture and document preparation. For this dissertation, mainly document management and its subcategory document meta data, document capture and its subcategory document analysis and document preparation with its subcategories annotation and formatting are relevant.

Text stream processing

For the purpose of this dissertation, the term *text stream processing* subsumes all three areas in the context of data streams, and is thus the hypernym of

[2]The system used in this dissertation is called *StreamFI*. This is derived from *Stream FI-lter*.

them. This means for instance if methods or algorithms are studied in the context of text stream processing, then they apply for one of the three areas in an text stream context. This generic term is also required, in order to avoid enumerating all three areas repeatedly.

3.2 Basic principles of information filtering

3.2.1 Background and literature review

The last section compared information filtering to information retrieval and defined the meaning of the term in the context this dissertation. This section presents the background of information filtering in more detail, as information filtering is the main subject of this dissertation. Additionally a review of related literature is provided, in order to credit influential research work and to contrast these with the findings of this dissertation.

Information dissemination

Hans Peter Luhn proposed in (Luhn, 1958) a *business intelligence systems* that would disseminate information among users based on profiles that reflect some area of interest(p. 315]). Luhn described the mode of operation of the system as follows (Luhn, 1958, p. 316):

> As soon as a new document has been entered into the system and its pattern developed, this pattern is set up in a comparison device which has access to all of the action point profiles. The comparison are carried out on the basis of degree of similarity, expressed in terms of a fraction, for each of the profile patterns. This fraction is subject to change as time goes on ....

Peter J. Denning

The term *information filtering*, which is used nowadays, was coined by Peter J. Denning in 1982. In an article, he argued that "[i]t is now time to focus more attention on receiving [rather than generating] information – the processes of controlling and *filtering information*[emphasis added] that reaches the persons who must use it" (Denning, 1982, p. 163).

Cognitive filtering

In (Malone et al., 1987) a system called "Information Lens" was proposed that was intended to tackle the "information-sharing problem, which has to do with disseminating information so that it reaches those people to whom it is valuable without interfering with those to whom it is not"(p. 390). The system was an information sharing system that address three types of information filtering that they identified in a survey for this paper: *cognitive filtering*, *social filtering* and *economic filtering*. The first describes the sifting through

messages using key words, the second is filtering message by the people that are involved in the messages and the third is concerned with evaluation of cost-benefit ratios of a certain message, e.g. is the cost of reading a lengthy message yielding an adequate benefit. In this sense this dissertation falls into the category of cognitive filtering, because the system that is presented later is exclusively concerned with filtering messages using textual and content-based features.

Belkin's two sides of a coin

A seminal paper on information filtering was "Information filtering and information retrieval: two sides of the same coin?" by (Belkin & Croft, 1992). In this paper information filtering and information retrieval were compared side by side and the commonalities and differences were elaborated. The findings of this paper have already been used in the introduction of this chapter. More details can be found there.

In the 1990s many information filtering systems[3] were invented to cope with the challenges of the rising WWW. For instance, in (Moukas & Maes, 1998) a system called Amalthaea was proposed. A multi-agent filtering system for "personalized filtering, discovery and monitoring of information sites" (Moukas & Maes, 1998, p. 59). They introduced the concept of *information filtering agents* that act "as a very specialized filter that is applied only in a narrow sector of the domain" (Moukas & Maes, 1998, p. 60)[4].

Lanquillon

(Lanquillon, 2001) presented a PhD dissertation on information filtering that discussed threshold determination for efficient class separation and introduced quality control mechanism for adapting to changes in the text stream. The topics covered in that research study are similar to the one presented here, but nevertheless there are major differences. The following points were not studied (or not on the same level of detail) by Lanquillon.

- The evaluation of state of the art term weighting schemes and their effect on the quality of the filtering system
- The introduction of new term weighting schemes addressing high-velocity text stream properties
- The study of a variety of different threshold and relevance feedback mechanisms
- The application of limited memory (=temporal windows) and their effect on filtering performance
- The combination of event stream processing techniques to address highly dynamic, volatile, heterogeneous text stream source

[3] An overview can be found in (Lanquillon, 2001, p. 17ff).

[4] This definition maps to *filtering event processing agents* that are introduced later.

- The development of a relevance feedback algorithm that models the dynamic nature of a text stream as well as the (constant) information need of the user

Another distinction to his research work is that the experiments presented in this dissertation are conducted using larger and more heterogeneous corpora. In (Lanquillon, 2001) the corpora contained only a few thousand documents from homogeneous sources; in addition for the experimentations these were split in equal and homogeneous batches. This dissertation deals with streamed text without any modifications to the text stream. In essence this dissertation applies and extend the findings from (Lanquillon, 2001) and intends to advance them to the current age of data processing.

Arampatzis

In terms of thresholds for document classification in text streams, (Arampatzis, 2001) presented a threshold algorithm extending findings presented in (Baumgarten, 1999). He proposed a score-distribution algorithm that "takes into account not only the means of these distributions but also the spreads of the data and the relative density of relevant to non-relevant documents in a stream" (Arampatzis, 2001, p. 45).

Text REtrieval Conference

An important actor in the field of information filtering is the *Text REtrieval Conference*(TREC). The first filtering track was held during TREC-4 in 1995 (Lewis, 1995b). As mentioned previously, information filtering gained interested during the 1990s due to the rise of the internet. Many filtering systems were developed, and thus a scientific frame like TREC helped researchers to investigate this field and compare their approaches and results. This subject evolved over time into three different filtering tasks: *adaptive filtering*, *batch filtering* and *routing* (S. E. Robertson & Hull, 2000, p. 1). Adaptive filtering is the *core* filtering task at TREC and was described in (Hull, 1998, p. 1):

> Given a topic description build a filtering profile which will select the most relevant examples from an incoming stream of documents. As the document stream is processed, the system may be provided with a binary judgement of relevance for some of the retrieved documents. This information can be used to adaptively update the filtering profile.

In contrast, both batch filtering and routing had all relevance judgements available upfront and could optimize the user profile in advance. Batch filtering had to return a relevance assessment (relevant vs. irrelevant) like adaptive filtering while routing was supposed to return a list of ranked documents.

Real-time filtering

However, the interest in information filtering dropped at the beginning of the 2000s, because there was no separate information filtering task between 2003 and 2011 at TREC. There were several tracks that incorporated aspects of information filtering like the *novelty track* in 2003 (Voorhees, 2003). But it was only until 2012 when the *real-time filtering pilot task* was introduced due to the increasing importance of social media platforms like Twitter and the accompanying flood of documents.

The goal of this track was described in (Macdonald et al., 2012, para. 28) as follows[5]:

> In the real-time filtering task, the aim is to decide if subsequently posted Tweets are relevant for a query entered at a particular point in time. The real-time filtering task can be thought of as orthogonal to the real-time ad-hoc task: in the ad-hoc task the user is interested in Tweets before the queryTweettime[6]; in the filtering task, the user has seen the Tweets before the queryTweettime, they are now interested in new relevant Tweets. This allows a user to keep up to date about a developing topic on Twitter. Indeed, this can be seen in the "20 new Tweets" aspect of the Twitter search page.

Due to changes in the *application programming interface* (API) usage of Twitter, the filtering task was dropped from TREC in 2013 on and replaced by the "Tweet timeline generation task" that asks for a topic summary for a given query. This task evolved into a separate, independent track with TREC called the *Temporal Summarization (sic) Task*.

In summary, it seems to be a natural evolution of information filtering to provide summaries instead of single documents, because due to the amount of available sources of information an evolving summary of an event is easier to consume than single documents. This stage of information filtering is not covered in this dissertation due to scope and time restrictions, but this is a promising next stage for the results of this dissertation.

3.2.2 Classic vs. *new* information filtering

Classic information filtering

The research works that were described till the cancellation of the filtering

[5] These guidelines were also used later for conducting the experiments in this thesis, because the used corpora are both Twitter based and TREC guidelines are an accepted scientific framework for comparable Twitter research.

[6] "[Q]ueryTweettime tag contains the timestamp of the query in terms of the chronologically nearest Tweet id within the corpus" (paragraph 31).

tracks at TREC 2002 can be summarised as "classic information filtering". The reason is that the text streams that were studied then had different quality in terms of velocity, volume and heterogeneity compared to the ones that are used in this dissertation and that have been subject to many research initiatives since the rise of social media applications like Twitter and Facebook in the mid of the 2000s.

For instance in (Lanquillon, 2001, p. 191) the four corpora used contained 8,000 to 25.,00 documents. In (Arampatzis, 2001) the *OHSUMED* corpus was used that contained 348,566 documents from the online medical information database *MEDLINE*. In (Harman, 1994) the corpora that were used till TREC 2001 were described. They were based on documents from the *Wall Street Journal*, *AP Newswire* or the *Federal Register* and contained in total 226,087 documents (Harman, 1994, p. 3). For the TREC 2001 edition a new corpus was composed: *Reuters Corpus Volume 1* (RCV1). This corpus contained 800,000 news documents from Reuters from August 1996 to August 1997.

New information filtering

In contrast, the research work that has been done since the rise of the *new* text streams in the mid of the 2000s is based on corpora of a different quantity and quality. The previously used corpora were relatively small and foremost from homogeneous sources like MEDLINE or Reuters. The corpora used for "new information filtering"[7] are much larger and the content is much more heterogeneous, because the content is no longer created by a curated source, but by millions of internet users.

Notable in this context are the corpora that were composed for the TREC microblogging tracks. The *TREC 2011* corpus contained more than 16 million documents (Twitter Tweets) with more than 50,000 relevance judgements (McCreadie et al., 2012). The 2013 corpus already contained approx. 240 million Tweets (Efron et al., 2013). The *ICWSM 2011* corpus contained over one terabyte of data and more than 386 million blog posts, news articles and social media content (Burton et al., 2011).

The growth of the corpora is based on the simple consumption of public APIs that are available from different social media or new sites on the internet. Thus, it is easy to build large scale corpora for research purposes. Nevertheless it is important to have curated corpora with well defined search topics that allow for comparison of different analysis methods. The TREC microblog track provides the only viable research corpus and methodology to compare filtering and search performance on microblog documents. As the two corpora that are used later on for the evaluation are Twitter based corpora, the filtering methodology described in the TREC 2012[8] guidelines

[7] *New* because of the new quality and quantity of the investigated corpora

[8] First results of this dissertation using event processing based methods for information

for the real-time filtering task are used to guide how information filtering is done in this thesis[9].

3.3 Basic principles of information retrieval

Previously, this dissertation discussed the difference between information filtering and information retrieval, as well as the basic principles of information filtering. Now relevant concepts of information retrieval are introduced, because there are many components that are virtually identical in information retrieval and information filtering (cf. (Oard, 1997, p. 18), (Belkin & Croft, 1992, p. 35)) and thus are important in the context of this dissertation.

3.3.1 Text collections and corpus types

The base of every text analysis system are the documents on which the system operates. A *document* is the basic unit for which the analysis system is built (Schütze et al., 2009, p. 4), i.e. system implementation can defer due to the types of document they process. For instance, system that take blog posts a their input documents are different to the short documents like Tweets, which are used here, in terms of index structure or processing requirements. The entirety of the documents is referred to *text collection* or also *corpus* (p. 4).

Dynamic sub-corpora

In this dissertation the term *dynamic sub-corpus* is essential. In the title of this dissertation the term *limited memory* is used and this aspect is represented via sliding windows on which dynamic sub-corpora are based. In the sense of this dissertation, a dynamic sub-corpus is therefore a corpus that is based on a fixed or varying sliding window and only contains documents that have occurred within the given sliding window. Thus, the corpus is highly dynamic and corpus content and statistics change rapidly and continuously of the course of an analysis. This is important to keep in mind, because all experiments and evaluations are based on this idea.

Other corpus types

This is fundamentally different to systems and evaluations that use static, semi-static, incremental, non-incremental or distributed corpora[10]. A static corpus is a fixed set of document, i.e. no new documents are added or deleted and the corpus statistics do not change over time. This type of corpus is not

filtering were submitted to the TREC 2012 conference (Bauer & Wolff, 2012) real-time filtering task and yielded rank 13 out of 60 submitted runs.

[9] The approach was described in the previous section

[10] These types of corpora are derived from the types of indices discussed in (Büttcher et al., 2010, p. 104ff, p. 228ff, p. 489ff).

relevant in this context. Semi-static corpora allow that new documents are inserted or deleted, but these modifications are not reflected in the corpus and in the pertaining index immediately (Büttcher et al., 2010, p. 228). The important fact here is that documents can also be removed when a new version of the corpus is built. Non-incremental are similar in terms that documents can be deleted. But the approach is more dynamic and is based on the idea to invalidate and then remove documents based on e.g. date or scores (p. 243f). Incremental corpora are different to these two approaches as they grow over time and already contained documents do not get deleted or removed (p. 228).

3.3.2 Term weights and scoring

In this dissertation *term weighting* is described as the process of attributing a value to an index term that represents its relevancy as a single number. *Scoring* is defined as the process of calculating a total score of a document using a scoring function. In information retrieval such scoring functions are often linear (Lewis et al., 1996, p. 298f)), i.e. the score of a document is calculated by adding up the values that are assigned to single document features. This leads to *monotonic increasing scores* and the higher the score the higher the probability that the document is relevant. This dissertation only focuses on this type of scoring functions. This is due to the fact that linear functions are widely used and straightforward to calculate and are therefore a good choice in a high-velocity text stream scenario.

More sophisticated approaches using e.g. *learning to rank* are not used, because they require more training data than available in the corpora and are not as lightweight as linear functions. There are already approaches that use learning to rank methods for text streams and social media streams. As term weighting is one of the four components that are evaluated in chapter seven, term weighting schemes are described that are used later on for evaluation purposes. But first the basic structure of term weighting schemes is described.

Within-document term frequency

In general there are three important components for term weighting that should be considered (Salton & Buckley, 1988, p. 514f). The first component is a *within-document term frequency factor*. This factor is a local one, i.e. the weight of the term depends on its occurrence within a document and is only considered in this context.

Its overall occurrence is only taken into account in context of the global term weighting. The idea behind term frequencies is that terms that appear often within a document can be considered as relevant for the content of the docu-

ment. Hence it is reasonable to assign these terms with higher values within the term weighting scheme. (Salton, 1968) showed that weighting terms by their frequency outperformed weighting schemes where only the presence or absence of terms were used. The term frequency is denoted $tf_{t,d}$ where t denotes the term and d the document (Schütze et al., 2009, p. 117).

The assumption to take the frequency of terms within a document as representation of the document is closely related to the concept of considering a document as a *bag-of-words* (BoW), i.e. that the order of terms within a document is not relevant, only the presence and frequency of term (Schütze et al., 2009, p. 117). This has the consequence that documents with different word orders have the same BoW representation and thus the same score when it comes to document ranking, e.g. "the fox is faster than the turtle" has the same BoW representation as "the turtle is faster than the fox". Even though this is a simplification, it is obvious that documents with similar BoW representations are supposed to have similar content (Schütze et al., 2009, p. 117). Thus, the BoW representation for documents is used throughout this thesis.

Inverse document frequency

The second component is a document collection wide factor. If you do not consider the term frequency, all terms would be equally important over all documents. This would mean that their discriminative power decreases, which is why (Salton & Buckley, 1988, p. 516) proposed to include additionally a document collection dependant factor that takes into account how often a term is used throughout document collection. The best known approach for this is idf. The foundations for this approach were published by (K. S. Jones, 1972) in 1972 and only later became known under the term idf (S. Robertson, 2004, p. 1). The idea behind idf is to assess the overall importance of a term within a document collection by its overall occurrence across all documents within the collection. This is based on heuristic observations made by (K. S. Jones, 1972, 493ff).

Document normalisation

The third term weighting factor is a *normalisation factor* (Salton & Buckley, 1988, p. 517). The idea is that long documents have a higher chance to match a search term than shorter documents, because they contain more words for a potential match. But as all documents should be considered as equally relevant, a normalisation is used to account for this situation.

Various term weighting schemes that are used in the evaluation chapter use this threefold approach like *BM25*, *Divergence from Randomness* or the newly proposed approaches.

3.3.2.1 Probabilistic retrieval models

Now the probabilistic retrieval models are briefly introduced, because probabilities and count-based calculations are at the centre of the methods used in this dissertation. This is because of the fact the counting is well operationalizable in a stream setting and thus count-based methods are suited to be used for document representations and scoring.

The probabilistic retrieval model was first introduced by (M. E. Maron & Kuhns, 1960) in 1960 and then further investigated by (Cooper, 1976) and (S. E. Robertson & Jones, 1976). (S. E. Robertson, 1997) then detailed the work regarding the probabilistic retrieval model. In contrast to the *Boolean retrieval model*, probabilistic retrieval does not deal with binary decisions but with probabilities, i.e. the results provided by a retrieval system for a given query only describe the probability of each single result being relevant to the query. This is due to the fact that not all information is available to the retrieval system that is necessary to make an absolute decision about the relevance or rank of a document. A retrieval system would need an optimal user profile that perfectly describes the interests of the user, a perfect query which completely reflects the information needs of the user and a perfect context, which describes the situation of the user, the system and the query.

Probabilistic Ranking Principle

It is obvious that this cannot be achieved, be it due to the fact that a user cannot perfectly express his information needs, or that not all information about the user is available. This is also called the "'library problem' (i.e. , the problem of information search and retrieval)" (M. E. Maron & Kuhns, 1960, p. 217). The core of the probabilistic retrieval model is the *Probabilistic Ranking Principle* (PRP). The PRP was introduced by (Cooper, 1976) (cited in (S. E. Robertson, 1997, p. 295)) and is defined as follows:

> If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of the data.

3.3.2.2 BM25

The first model that is introduced is *BM25*[11]. It is not a formal model, but based on probabilistic principles and has shown to be a well-performing ranking algorithm in scientific settings like TREC or in commercial and open source search engines (Croft et al., 2010, p. 254)[12]. It was introduced in (S. E. Robertson & Walker, 1994).

For a document D the BM25 score given a query Q with the search terms $q_1, \ldots, q_n$ is defined as in formula

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}, \tag{3.1}$$

where $f(q_i, D)$ is the frequency of query term q_i in the document, k_1 is a hyperparameter, which corresponds to a binary model, i.e. term frequency is ignored, when set 0 or to the raw term frequency if set to 1 (Schütze et al., 2009, p. 233). b is a hyperparameter to scale the influence of the document length. If it is set to 0, the length normalisation expressed by $b \cdot \frac{|D|}{\text{avgdl}}$ is removed, while setting the value to 1 means the term is scaled according to the full length normalisation.

3.3.2.3 Divergence from Randomness

The divergence from randomness model is also a probabilistic retrieval model. The idea is to "derive term-weighting models by measuring the divergence of the actual term distribution from that obtained under a random process" (Amati & Van Rijsbergen, 2002, p. 1). In their paper they investigated several different random processes like binomial distributions or *Bose-Einstein* statistics.

The process is detailed in (Terrier Team, 2011). First, a randomness model like *Bose-Einstein statistics* is selected. The idea is straightforward. The more different a term frequency within a document is from its frequency within the collection, the more important is the term. Step two and three are concerned with normalisation. In the second step terms are normalised or smoothed. And in the third step a document normalisation is applied.

[11] BM = Best Match (Ounis, 2005)

[12] *Elasticsearch* implements a version of BM25

The score for a term is calculated using formula 3.2.

$$score(d, Q) = \sum_{t \in Q} qtw \cdot w(t, d) \tag{3.2}$$

t is a query term within query q. Qtw corresponds to the weight of the query term, which is calculated as $\frac{qtf}{max(qtf)}$, where qtf is the frequency of the query term within document d. $w(t, d)$ is the weight of document d given the query term t.

$w(t, d)$ can then be calculated using formula 3.3:

$$w(t, d) = [-\log\ P(t|C)]\ \cdot\ [1 - P(t|d)] \tag{3.3}$$

with C being the document collection. The probability $P(t|d)$ can be calculated by using one of the random processes mentioned previously. For instance, one of the evaluated models is the "Inverse Document Frequency model with Laplace after-effect and normalisation 2" that is calculated using formula 3.4 (Terrier Team, 2011).

$$P(t|d) = \frac{1}{tfn + 1}\left(tfn \cdot \log_2 \frac{N + 1}{n_t + 0.5}\right) \tag{3.4}$$

3.3.2.4 Language Models

A language model is a statistical, generative model. The idea is that a document is generated based on a statistical model and a query that matches this document well is also generated by the same model (Schütze et al., 2009, p. 237). This is in contrast to the previously presented probabilistic model. While the latter is intended to maximize the relevance of a document to a query, the former tries to build a statistical model that generates a document and ranks the documents according to their probability to generate the given query (Schütze et al., 2009, p. 237). The main types of language models are *unigram* and *bigram* models. The calculation of language models is based on the *chain rule*, which allows deconstructing the probability of a language model into a disjoint product of probabilities. Formulae 3.5 and 3.6 show how the probabilities for unigram and bigram language models are calculated.

$$P_{uni}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4) \tag{3.5}$$

$$P_{bi}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_4|t_3) \tag{3.6}$$

3.3.2.5 Related term weighting research

Above the *classic* term weighting schemes as used in the area of information retrieval were described. In this section a few research works will be presented that explicitly address microblogging content.

Core Word Score

In (Gupta et al., 2012) the *Core Word Score* (CWS) was proposed, which was intended to replace the *term frequency* factor in the *tf.idf* term weighting schemes. The CWS is defined as the ratio of the term frequency compared to the overall average of the term in the current time interval. Thus, this term weighting scheme is a temporal one and is also used for comparison in the evaluation section. The goal is to assess the performance of this weighting scheme in an information filtering setting. In (Gupta et al., 2012) it is used for event detection and is not compared to other term weighting schemes. Therefore an assessment in an information filtering context is missing.

BursT

Another term weighting approach that is evaluated is *BursT* presented in (Lee et al., 2011). The final term weight consists of two factors: *BS = burst score* and *TOP = term occurrence probability*. The value is calculated using formula 3.11.

$$weight_{w,t} = BS_{w,t} \cdot TOP_{w,t} \tag{3.7}$$

with

$$ar_{w,t} = \frac{1}{at_{w,t} - at_{w,t-1} + 1} \quad \text{(arrival rate)} \tag{3.8}$$

$$E(ar_{w,t}) = \mu_{w,t-1} + \frac{1}{n_{w,t}}(ar_{w,t} - \mu_{w,t-1}) \quad \text{(incremental mean)} \tag{3.9}$$

$$BS_{w,t} = max\left\{\frac{ar_{w,t} - E(ar_{w,t})}{E(ar_{w,t})}, 0\right\} \quad \text{(Burst Score)} \tag{3.10}$$

$$TOP_{w,t} = P(w_t|c_t) \quad \text{(Term occurence probability)} \tag{3.11}$$

3.3.3 Information retrieval models

As stated in the previous section, it is mainly probability based methods that are studied and used in this dissertation due to their count-based nature and

the good operationalizability of counts in data streams. But information retrieval models can not only be characterized by their mathematical nature. (Kuropka, 2004) presented an interesting categorization along various dimensions that is briefly discussed here, in order to discern which retrieval models are in scope and which are not in scope. Figure3.3 shows the categorization proposed by (Kuropka, 2004, p. 43ff). He discerned the most popular information retrieval models along their mathematical foundations and the type of term dependence they incorporate.

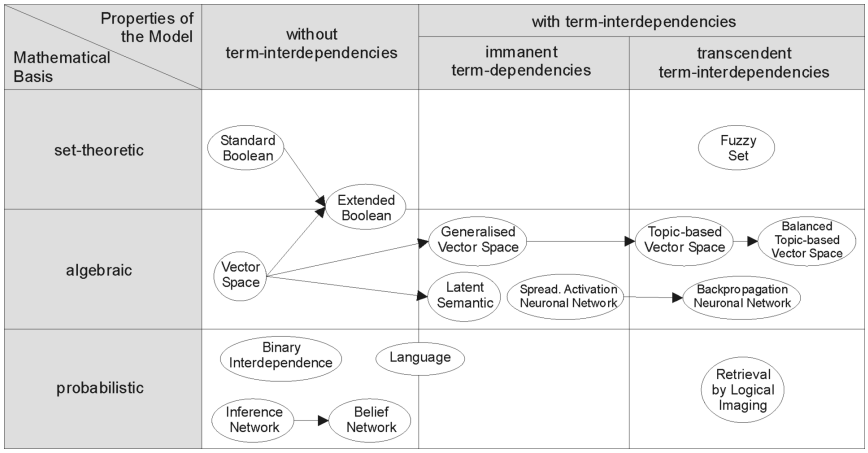

Figure 3.3: Overview of common information retrieval models as proposed by (Kuropka, 2004, p. 44) cited after (Wikipedia, 2015e).

Categories of retrieval models

In the mathematical dimension, there are set theoretic models, algebraic models and probabilistic models. Set theoretic models are based on Boolean logic and include the *Standard Boolean Model*, partially the *Extended Booelan Model* and models using *fuzzy logic*. In this dissertation basic Boolean models are used for filter policies and the determination of query matches that are used for scoring. As mentioned by (Kuropka, 2004, p. 50) these models are good for a basic matching due to their easy application, but they only provide a binary similarity measure of a document and a query. However, as a first stage in the filtering process these models are useful. Of course, the matching stages in the filter process of the event driven information filtering system that is used in this dissertation can be extended by using fuzzy logic, for instance. But in order to keep the focus on essential methods, this can be subject to later research.

Algebraic models are the *Vector Space Model*, *Latent Semantic Indexing* or *Neural Networks*. These methods imply the execution of matrix operations on high-dimensional spaces, in order to derive document similarities. But as mentioned previously, the focus here is on probability and count based methods due to their good applicability to data streams. Nevertheless, the study

of algebraic methods is an interesting subject for further research.

The final category is probabilistic mode. These models form the focus of this dissertation.

In the term dependence dimension, there are also three categories. The first category contains models without any assumed term dependence (Kuropka, 2004, p. 49). This means the occurrence of a term is independent of the occurrence of other terms. All terms are therefore *orthogonal* and form a so-called *bag-of-words*. This class includes, among others, the *Standard Boolean Model*, the*Vector Space Model* or the *Extended Boolean Model* (p. 49). Due to the assumed term independence they are easy to calculate, which makes their application interesting in a high-volume, dynamic text stream scenario. Thus, mainly the bag-of-words approach is used in this dissertation. In addition, there are two more classes that do incorporate term dependencies (Kuropka, 2004, p. 63ff). The first class is called models with *term-immanent dependencies*. Such models usually represent term dependencies using co-occurrences that are present in the documents. The second class is models with *term-transcedent dependencies*. This means the dependency of terms is not based on the occurrence of terms within the document collection, but it is based on factors that come from outside of the document collection. Such factors can either be modelled by humans or learned by machine learning algorithms like *backpropagation neural networks* (p. 73).

3.3.4 Thresholds and text classification

Information filtering tries to solve the problem of classifying documents as relevant or not relevant for a given information need, i.e. it tries to solve a *binary decision problem*. In order to classify a document a decision boundary is required that separates the classes. Such a decision boundary can also be called a *threshold* or *decision hyperplane* (Schütze et al., 2009, p. 302).

Linear classifiers

As described in chapter 3.3.2, the scoring schemes that are used in this dissertation are linear. Thus, a threshold in such a setting represents a linear separation between the relevant and the non-relevant document classes, i.e. it functions as a linear classifier (Schütze et al., 2009, p. 301ff). (Aggarwal & Zhai, 2012, p. 193) defines linear classifiers as

> ...those for which the output of the linear predictor is defined to be $p = \bar{A} \cdot \bar{X} + b$, where $\bar{X} = (x_1 \ldots x_n)$ is the normalized[sic] document word frequency vector, $\bar{A} = (a_1 \ldots a_n)$ is a vector of linear coefficients with the same dimensionality as the feature space, and b is a scalar."

This category of classifiers includes for instance *Support Vector Machines*(SVM), regression based classifiers or neural network classifiers (Aggarwal & Zhai, 2012, p. 194ff).

The problem with linear classifiers is that the two classes are not clearly separable, as there are *noise documents* (Schütze et al., 2009, p. 303). Such noise documents are problematic, because they "do not fit well into the overall distribution of the classes" (p. 303), i.e. non-relevant documents yield high scores or relevant scores yield low scores and thus the decision boundary is distorted. Therefore, thresholds are hard to find (p. 304) and it is important to study if the dynamic sub-corpus approach studied in this dissertation is able to provide reasonable results compared to machine learning based classifiers.

The case for simple thresholds

While these classifiers and their application to text classification is well studied, threshold strategies are considered an under-explored area of research, because it was argued that the threshold strategy does not have any significant influence or the optimal threshold is trivial to find (Yang, 2001, p. 137). Nevertheless, the threshold based approach is simple to determine, but in a high-volume text stream scenario efficient methods are welcome and thus it is worthwhile to study how such methods perform compared to more sophisticated methods.

The application of a threshold as a linear classifier is described in algorithm 1(Schütze et al., 2009, p. 302).

Algorithm 1 Linear classifier algorithm

1: **function** APPLYLINEARCLASSIFIER($\vec{w}, b, \vec{x}$)
2: $score \leftarrow \sum_{i=1}^{M} w_i x_i$
3: **if** score > b **then** return 1 [
4: **else**return 0]
5: **end if**
6: **end function**

The crucial part of this algorithm is to determine a value for threshold b. (Sebastiani, 2002, p. 21ff) describes two ways of determining thresholds: *analytically* or *experimentally*. The first means that there is a theory by which it is possible to calculate a threshold using a given formula. If this is not possible, then one reverts to the second possibility and the threshold is determined by trying different values and by evaluating which value maximizes a given performance measure like precision or recall.

Probability Thresholding Principle

Analytical methods, which are used to determine a threshold, are often based on distribution mixture models and the optimization of an effectiveness mea-

sure[13]. The idea is that scores for the relevant and the non-relevant document classes are distributed based on the *probability density function* (PDF) of a certain distribution (Arampatzis & van Hameran, 2001, p. 287). The parameters required for determining a distribution, like mean or standard deviation, can be calculated using scores from a training set of relevant and non-relevant documents. When the parameters are calculated, the score of a new document can be turned into a probability, and the *Probability Thresholding Principle* (PTP) applies. (Lewis, 1995a, p. 248) defines this principle as follows

> For a given effectiveness measure, there exists a threshold $p, 0 \leq p \leq 1$, such that for any set of items if all and only those items with probability of class membership greater than p are assigned to the class, the expected effectiveness of the classification will be the best possible for that set of items.

How the probability p can be calculated was shown in (Duda & Hart, 1973, p. 15):

$$p = \frac{\lambda_2 - \lambda_4}{(\lambda_3 - \lambda_1) - (\lambda_2 - \lambda_4)} = \frac{1}{1 + \lambda} \tag{3.12}$$

where

$$\lambda = \frac{\lambda_3 - \lambda_1}{\lambda_2 - \lambda_4}. \tag{3.13}$$

The parameters $\lambda_1 to \lambda_4$ correspond to a cost or gain for each of the categories in contingency table 3.4 (Arampatzis & van Hameran, 2001, p. 286). Based on this, a linear utility function for instance could look like this:

$$utility(d) = \lambda_1 tp + \lambda_2 tn + \lambda_3 fp + \lambda_4 fn \tag{3.14}$$

For instance, when λ_1 and λ_4 are set to 0 and λ_3 and λ_2 are set to 1, the probability p becomes 0.5 using formula 3.13, which says a document is considered relevant when the probability of the score using the PDF of the relevant class is larger than 0.5 (Lewis, 1995a, p. 249).

Distribution mixture models

The distributions mixture approach was first proposed in (Swets, 1969, p. 13) and various combinations of distributions have been studied since then. (Y. Zhang & Callan, 2001a) used a Gaussian distribution to model the relevant-documents and an exponential distribution for the non-relevant. They used a generative model to create the distributions, i.e. they estimate the parameters for the distributions from a training corpus and use the intersection as the threshold.

[13] Cf. chapter 3.3.6

(Arampatzis, 2001, p. 53ff) also used a mixture model based on incrementally updated means and deviation values. It was called *score-distributional threshold optimization* and was geared towards optimizing a given utility function. He also used Gaussian and exponential distributions. In (Baumgarten, 1999), (Callan, 1998), (Lanquillon & Renz, 1999) and more recently in (Arampatzis et al., 2009) the distribution mixture models are used. In general the Gaussian-exponential model is considered as the most popular (Arampatzis & Robertson, 2011, p. 27).

All the mentioned research offers different approaches for the determination of distribution parameters, but in general it is difficult to correctly estimate a distribution and its pertaining parameters, or to correctly assume a distribution for a document class (S. Robertson et al., 2013, p. 86). This is foremost true in a highly volatile and noisy environment like text streams. In contrast, the threshold methods studied in this dissertation avoid the tedious task of initial document training and try instead to continuously adjust to the text stream by using adaptive sliding windows and the possibilities of fine-grained score monitoring offered by the application of event processing. The goal of this dissertation is to study whether such a computationally simple approach is effective.

Empirical thresholds

The second group, experimentally derived thresholds, is classified in three categories (cf. (Sebastiani, 2002, p. 22f), (Yang, 2001, p. 1)): "rank-based thresholding (RCut), proportion-based assignments (PCut) and score-based optimization (SCut)' '(Yang, 2001, p. 1). *SCut* thesholding is based on testing various thresholds given a set of scored documents, i.e. various thresholds are tried for a search profile and the one with the best value for a given measure like precision is chosen. *RCut* incorporates the idea to calculate a score value for each document given a category (= search profile). Then, for k categories with the highest score this document is considered relevant. *PCut* sets the threshold based on the proportion of relevant and non-relevant documents in a training set, i.e. if the proportion of relevant documents in the test set is 2%, then also the top 2% percent documents of the incoming ones should be considered as relevant. The value k for RCut and PCut based approaches are also tuned in terms of the optimization of a given performance measure. According to (S. Robertson, 2002, p. 3) the analytical approach can also be called a *model-based* one, while the experimental one is a *model-free* approach.

The thresholding principles described above have been used in the classic information filtering setting and have yet not been studied in the high-velocity text stream context. Therefore section 8.4 will focus on the evaluation of different approaches to parameter estimation for the model-based approach, as

well as the investigation of the performance of various approaches on the model-free side. Additionally some straight forward methods are studied based on *statistical moments* like mean or mode. Lastly, these approaches are compared in terms of their performance against machine learning classifiers like Naïve Bayes and logistic regression.

Thresholds in high-velocity text streams

Finally, there were approaches that investigated thresholds in a high-velocity text stream scenarios, but in general they were geared towards top-k document monitoring. This means they studied efficient ways to identify the top-k documents for a query and to keep this list up-to-date while new documents arrive. This was done e.g. in (Mouratidis & Pang, 2011), (Mouratidis et al., 2006) or (Haghani et al., 2010) and their research work adapted the *thresholding algorithm* presented in (Fagin, 2002). The results from these studies were focused on the efficient processing and on aspects of computer scientific. The focus was not explicitly on information scientific questions such as how such a top-k threshold behaves in contrast to other approaches. Therefore a top-k based threshold approach will also be evaluated in terms of its performance compared to other threshold mechanisms.

3.3.5 Query expansion and relevance feedback

Users are usually not able to describe their information needs exhaustively and in full detail. Thus, there is always uncertainness in a query, because the user does not know all of the concepts needed to formulate the query or is still in the process of shaping his information need. That is why a system cannot deliver a perfect match of documents and hence the results provided by the system are only the system's *assumptions* of what might be interesting to the user.

Local and global methods

There are two methods to address this issue: *Relevance feedback* and *query expansion*. In (Schütze et al., 2009, p. 177f) the first method is referred to as a *local* method, because it operates only the document by adjusting the initial score. Query expansion in contrast is described as a *global* method, because it works independently from a result document and adds query terms to the initial query based on e.g. thesauri.

Both concepts are introduced for two reasons. First, relevance judgements for the studied corpora are available. This valuable information should be used in order to address the just introduced *synonymy* and query formulation problem (Schütze et al., 2009, p. 177f). Second, the used corpora are social media corpora and the subject of this dissertation are short documents that usually have a problem with *sparsity* (Albakour et al., 2013, p. 421f); query expansion can help to address this problem.

Relevance feedback (Allan, 1996, p. 1) describes relevance feedback as follows:

> [T]echniques [that] provide means for automatically correcting a query to more accurately reflect the user's interests: a set of "good/bad" relevance judgements on documents are "fed back" into the query to generate a better query.

The first kind of relevance feedback is called *explicit relevance feedback* (Schütze et al., 2009, p.178ff). The user examines the delivered results and provides feedback to the system that is then used by the system to adjust its initial scoring and present a new ranking order to the user. This process is looped over until the user is satisfied with the results or decides to abort the search process.

But users are often reluctant to provide feedback, because this puts additional cognitive load onto the user, or systems only offer simple numeric feedback mechanisms (Oard, 1997, p. 23). That is why there are other approaches. For instance a system can use *pseudo relevance feedback* strategies in order to improve the result list. The idea here is that a system *assumes* what might be relevant to the user by considering the top-k documents that the system found to be relevant to the user (Croft et al., 2010, p. 212). Then these top-k documents are analysed and the top-k scored terms within these documents are used to modify the initial query.

Another approach is *implicit relevance feedback*, which was presented e.g. in (Oard, 1997, p. 23f) and (Shen et al., 2005). The idea here is to incorporate the context of the search into the feedback. This means the system e.g. examines the search history of the user, click-through rates of websites or the spatio-temporal context of the user. In general the interaction and the environment of the system and the user are recorded and incorporated to improve search results.

The last approach that that is to mention is *negative relevance feedback*. When a user is looking for information it is quite probable that more than 99% of all documents in a corpus are not relevant. (X. Wang et al., 2007) presented an approach to use only *top-k* negative documents and use these results to penalize documents that were similar to those documents.

In this dissertation only explicit and pseudo relevance feedback will studied. This is due to the fact that the other approaches would require additional information like search logs or click through rates that were not available for the used corpora.

Query Expansion In contrast to relevance feedback where only the initial search result is re-weighted using additional information, *query expansion* modifies the initial query by adding additional terms or phrases (Schütze et al., 2009, p. 189f).

In (Carpineto & Romano, 2012, p. 1:10ff) the process of generating query expansion consists of the following steps:

- Pre-process (stemming, stop word removal, etc.) the document for which relevance feedback became available
- Generate a ranked list l_{qet} of potential query expansion terms (QET)
- Select a fraction$f$ of l_{qet} as a candidate list for query expansion

They also identified four common categories of how to generate such a ranked list l_{qet} in (Carpineto & Romano, 2012, p. 1:12ff).

1. One-to-one associations: Use e.g. *WordNet*[14] to find synonyms for the query terms
2. One-to-many associations: Extend the previous approach by investigating the identified relationships in the context of the other query terms
3. Feature distribution of top ranked documents: Consider the first few relevant documents for a query as an *extension* of the initial query and use the top scored features from these documents
4. Query language modelling: Build a statistical model for the query and use the features with the highest probability.

The first two approaches are not studied in this dissertation, because external evidence is not used for the experiments[15].

Association hypothesis

The approaches 1 and 2 can also be considered as *local techniques*. They are based on the *association hypothesis* proposed by (Rijsbergen, 1979, p. 104), which states "[i]f an index term is good at discriminating relevant from non-relevant documents then any closely associated index term is also likely to be good at this". This hypothesis was mainly influenced by (E. Maron, 1964). (Rijsbergen, 1979, p. 103) summarises the idea behind this as "namely, to enlarge the initial request by using additional index terms which have a similar or related meaning to those of the given request".

[14] `http://wordnet.princeton.edu`

[15] External evidence is also links embedded in a Tweet. As such external features augment the context under which the document is studied, external evidence is ignored, in order to provide a clear view of the capabilities of the document features. Nevertheless, for further research the influence of external evidence is supposed to be an interesting area of research.

In contrast to this, there exist *global expansion techniques*. In order to apply global expansions techniques it is necessary to obtain corpus wide statistics like co-occurrences (Xu & Croft, 2000, p. 80). The corpus wide statistics are responsible for the name of the technique, as statistics have to be available for the whole corpus. Global expansion techniques are e.g. term clustering (Spark Jones, 1971), latent semantic indexing or the aforementioned feature distribution of top ranked documents or query language modelling.

The Rocchio algorithm

One of the first to investigate relevance feedback and query expansion was *Rocchio* in 1971 (Croft et al., 2010, p. 234). He developed this method for the *Vector Space Model*. The idea is to update the initial query vector depending on feedback from the user(cf. formula 3.15)

$$P' = \alpha P + \beta \frac{1}{|Rel|} \sum_{D_i \in Rel} D_i - \gamma \frac{1}{|Nonrel|} \sum_{D_i \in Nonrel} D_i \tag{3.15}$$

In this dissertation only the positives feedback documents and their terms were used to modify and expand the initial queries. This corresponds to setting $\gamma = 0$.

Lavrenko's *Relevance Model*

The *Relevance Model* was introduced in (Lavrenko & Croft, 2001). The goal of the approach was to model the probability $P(w|R)$ of observing a word w in the relevant class R for a given information need without having any relevance or training data available. They argued that in a real word setting there is only a query, a large document collection and no relevance information (Lavrenko & Croft, 2001, p. 121). Hence, there is no information available to estimate the probability $P(w|R)$. In order to estimate this probability, they assumed that both the documents and the query are generated from an unknown relevance model R(p. 122). This is in contrast to the query language model described in the term weighting section, where the argumentation is that the query is generated from the document model. They tried to approximate the probability to observe the word w given the probability of observing the term $q_1, \ldots, q_k$ in query Q. Based on a unigram model and based on a conditional sampling strategy, they calculated the probability $P(w|R)$ using the following formulae(p. 122f):

$$P(w|R) \approx P(w|q_1, \ldots, q_k) \quad \text{can be calculated as} \tag{3.16}$$

$$P(w|R) \approx \frac{P(w, q_1, \ldots, q_k)}{P(q_1, \ldots, q_k)} \quad \text{with} \tag{3.17}$$

$$P(w, q_1, \ldots, q_k) = \sum_{M_i \in \mathcal{M}} P(M) P(w, q_1 \ldots q_k | M) \tag{3.18}$$

$$P(w, q_1, \ldots, q_k | M) = P(w) \prod_{i=1}^{k} \sum_{M_i \in \mathcal{M}} P(M_i|w) P(q_i|M_i) \tag{3.19}$$

$$P(M_i|w) = P(w|M_i) \frac{P(w)}{P(M_i)} \tag{3.20}$$

$$P(w) = \sum_{M \in \mathcal{M}} P(w|M) P(M) \tag{3.21}$$

$$P(w|M) = \lambda \frac{tf(w, D)}{\sum_v tf(w, D)} + (1 - \lambda) P(w|G) \tag{3.22}$$

where M being an arbitrary unigram model from the model universe $\mathcal{M}$ and $P(M)$ is an arbitrary probability distribution from which the unigrams are sampled. $q_1, \ldots, q_k$ are the query terms of query Q and w is a word from the document. λ is an empirical parameter that was set to 0.6 in their experiments. $P(w|G)$ is calculated by dividing the collection frequency of word w by the total number of words in the collection. $P(q_i|M)$ represents the probability of observing query term q_i given a model M_I from the model universe $\mathcal{M}$.

This model was shown to perform rather well in experiments (Schütze et al., 2009, p. 250) and was therefore chosen as one of the reference implementations for the evaluation chapter in this dissertation.

Kullback-Leibler divergence based adjustment

Another approach where language models are used is to rank the similarity of a document for a given query by using the *Kullback-Leibler* divergence of the document model θ_d and query model θ_q.

$$sim(q, d) = \sum_{t \in V} p(t|\theta_q) log \frac{p(t|\theta_q)}{p(t|\theta_d)} \tag{3.23}$$

The Kullback-Leibler divergence is an asymmetric measure that indicates how well θ_d models θ_q (Schütze et al., 2009, p. 251). It was shown in (Lafferty & Zhai, 2002) that an approach that compares two models like the Kullback-Leibler divergence works well.

Decay-based adjustment

Finally a decay based approach is briefly introduced. The idea is to discount the initial score of the top ranked documents features using an exponential decay function of the form

$$score(w') = e^{-(t_{i-1}-t_i)} score(w) \tag{3.24}$$

where t_{i-1} is the timestamp when word w was seen for the last time and t_i is the timestamp of document. This means the value of $score(w)$ decreases exponentially to 0 if the term does not re-occur. The idea is to remove terms that apparently become non-relevant, because they do not re-appear.

3.3.6 Evaluating retrieval and filtering systems

It is straightforward to calculate a score value for a given document using a specific scoring method. But the performance of an information filtering or information retrieval system can only be truly assessed with suited evaluation methods. This section gives an overview of the most important evaluation measures that are used in this dissertation.

Cranfield paradigm

The first major milestone in assessing and evaluating document retrieval systems were the Cranfield experiments, which were run by Cyril W. Cleverdon at the Cranfield University between 1957 and 1967 (Cleverdon et al., 1966). These experiments led to the *Cranfield paradigm* that advocates for reproducible benchmarks. One goal of this dissertation to guarantee reproducibility by providing traceable and understandable evaluation results.

To assess and verify the performance of an information retrieval system, it is important to have sound evaluation methods at hand. Evaluations methods can be grouped into two group classes: one for ranked and one for unranked results. In this dissertation the latter group is relevant, because in information filtering is not about the generation of a ranked list of documents, but an approach to categorize documents in classes of relevance and non-relevance. Figure 3.4 provides an overview of available methods (Zuva & Zuva, 2012, p. 36) and indicates that for the purpose of this dissertation "non-graphical representation" methods apply. This means only the plain numbers for precision, recall or f1 are reported and no visual graphs like precision-recall graphs or *Receiver Operating Characteristic* (ROC) curves are used.

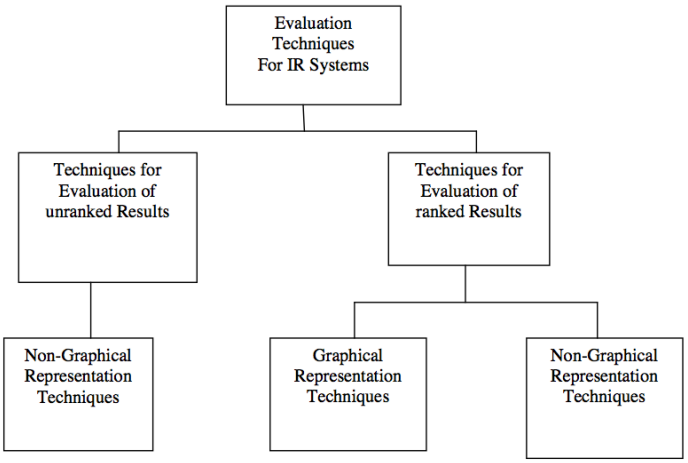

Figure 3.4: Hierarchy of evaluation techniques for information retrieval systems.

3.3.6.1 Defining relevance

Prior to introducing the evaluation measures used, an important aspect in this context should be discussed first: *relevance*. The calculation of any performance measure relies on the fact that retrieved or filtered data have been judged in terms of relevance, so one is able to compare the results of a retrieval or filtering systems with what human estimators considered as relevant. Tefko Saracevic defined relevance as "...a measure of the effectiveness of a contact between a source and a destination in a communication process" (Saracevic, 1975, p. 321).

The problem is that this effectiveness can be considered from different points of view. It is dependent on questions like

> what we think we want and how we ask for it; how we understand what is asked and what we think is really asked; what is wanted in contrast to what is really needed; who is asked, who is asking; what the situation is, what will be done with what is provided; and so on. (Saracevic, 1975, p. 325)

(Hjørland, 2005, para. 2) provide a more formal definition.

> Something (A) is relevant to a task (T) if it increases the likelihood of accomplishing the goal (G), which is implied by T.

This is a task oriented definition, i.e. the system should support the user in fulfilling his goals and satisfying the given information needs.

Relevance is subjective

Relevance is also always a subjective term, i.e. one retrieval or filtering re-

sult can be estimated as relevant for user A, but might be rejected by user B. Therefore inter-rater reliability should be assessed and reported for relevance judgements that are provided with a corpus by applying *Cohen's Kappa* measure (Schütze et al., 2009, p. 165)(cf. formula 3.25).

$$Kappa = \frac{p_0 - p_c}{1 - p_c} \tag{3.25}$$

(M. E. Maron & Kuhns, 1960, p. 220) also mention this problem. They described relevance as being a "primitive notion", which has to be explicated, i.e. the system should help to better understand his information needs.

An important aspect of relevance is *marginal relevance*. This concept states that a document can on the one hand be highly relevant to a query, but on the other hand does not offer any new information, as it is a duplicate of an already found document (Schütze et al., 2009, p. 167). *Marginal relevance* is important in a high volume text stream scenario. Given the example of Twitter and its concept of ReTweets[16], it can easily happen that during the height of an event, a document (Tweet) is perceived as interesting by many users, and is therefore often re-emitted. As a consequence we have a high volume of similar Tweets which are each relevant but do not offer any novelty or new aspect.

In this dissertation the TREC 2011 microblog corpus is used, which consists of approx. 16 million Tweets[17]. It is not feasible to assess the relevance of each Tweet. Hence at TREC usually a *pooling* strategy is used, i.e. only *top-k* documents retrieved by a baseline retrieval system are assessed.

The following evaluation methods have been applied during the experiments to assess effectiveness and are briefly described in the next sections:

- Precision
- Recall
- F1-measure
- T11SU

[16] A reTweet is the re-submission of an existing Tweets with or without slight modifications. ReTweets are usually marked e.g. by prepending *RT*

[17] This is only a theoretical value, because only the IDs of the Tweets are provided and the Tweet itself needs to be downloaded. As Twitter changes its storage and access policies not all Tweets can be downloaded

3.3.6.2 Precision and recall

Two seminal evaluation methods are *precision* and *recall*. One of the first to emphasize the importance of those two concepts was C. Cleverdon (Cleverdon et al., 1966, p. 2). Both are used extensively in the evaluation chapters and hence a detailed description of their calculation is provided here. In (Schütze et al., 2009, p. 155) precision is defined as the "fraction of retrieved documents that are relevant). Formula 3.26 shows how precision is calculated.

Precision

$$Precision = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items\ in\ total)} = P(relevant|retrieved) \tag{3.26}$$

.

Recall *Recall* in contrast is the "fraction of relevant documents that are retrieved" (Schütze et al., 2009, p. 155). Recall is calculated using formula 3.27.

$$Recall = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items\ in\ total)} = P(retrieved|relevant) \tag{3.27}$$

In the evaluation chapter the terms *true positives*, *false positives*, *true negatives* and *false negatives* are used extensively, in order to assess the quality of scoring methods or relevance feedback algorithms. Therefore it is useful to express precision and recall also using these concepts. The contingency table 3.4 shows how the various concepts are related to the terms used in formulae 3.26 and 3.27.

	Relevant	Non-relevant
Retrieved	true positives (tp)	false positives (fp)
Not retrieved	false negatives (fn)	true negatives (tn)

Table 3.4: Contingency table.

Using the terms from the contingency table, then according to (Schütze et al., 2009, p. 155) precision is calculated as

$$Precision = \frac{true\ positives}{(true\ positive + false\ positives)} \tag{3.28}$$

and recall as

$$Recall = \frac{true\ positives}{(true\ positive + false\ negatives)} \tag{3.29}$$

Different recall calculations

It is important to highlight that in contrast to formula 3.29, the recall values in the evaluation chapter of this dissertation are not calculated by summing the true positives and the false negatives, but directly use the number of total relevant documents provided by the assessors of the corpus. This means the recall value is independent of the systems results in terms of true positives and false negatives. This usually leads to lower recall figures, because the number of relevant documents is larger than the documents that a system filters for the group of true positives and false negatives. This is because of the relevance problem discussed in chapter 3.3.6.1. Nevertheless, the recall value in the sense just described is presented as *sensitivity* (cf. chapter 3.3.6.5).

3.3.6.3 *F*-measure

The *f-measure* is a single figure evaluation metric that is based on a weighted harmonic mean of precision and recall (Schütze et al., 2009, p. 156). The general formula for calculating the f-measure is given in formula 3.30. β indicates the weight that regulates the influence of precision or recall. If $\beta = 0$ then the f-measure corresponds to precision. If β approximate ∞ it is equivalent to recall (Lewis, 1995a, p. 249). Formula 3.30 shows the general form of calculating the f-measure (cf.(Schütze et al., 2009, p. 156), (Wikipedia, 2015a)).

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{3.30}$$

Using the previously introduced categories of *true positives*, *false positives*, etc. from the contingency table 3.4, the f-measure can be calculated using formula 3.31.

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{true positive}}{(1 + \beta^2) \cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}} \tag{3.31}$$

The problem in information filtering is to remove irrelevant documents and leave the relevant ones. Because the irrelevant documents outnumber the relevant ones by far, it is important that algorithms work precisely. Hence the precision of the employed algorithms is more important than the recall and therefore the $f_{0.5}$*-measure* ($\beta = .5$) is used in this dissertation, which allocates

twice the importance to precision than to recall.

3.3.6.4 T11SU

In terms of information filtering the *T11SU* measure is important and will also be reported later in the evaluation section. This measure was introduced for the TREC information filtering tracks and is a linear utility measure, which is based on the following equations (Soboroff et al., 2012, p.6).

$$T11U = 2 \cdot \text{num(relevant documents) - num(non-relevant documents)} \tag{3.32}$$

$$MaxU = 2 \cdot \text{num(relevant documents)} \tag{3.33}$$

$$MinU = -0.5 \tag{3.34}$$

$$NormU = T11U / MaxU \tag{3.35}$$

$$T11SU = \frac{max(NormU, MinU) - MinU}{1 - MinU} \tag{3.36}$$

Regarding the benefit of linear utility function, (S. Robertson, 2002, p. 6) states that a

> [u]tility [measure] translates directly into a retrieval rule: optimizing a linear utility measure is equivalent to retrieving a document if its probability of relevance exceeds a certain value (derived directly from the parameters of the utility function). This implies that we need a direct estimate of the probability of relevance of each document.

Finally it should be noted that a value of $\frac{1}{3}$ is known as the *zero effort* baseline, i.e. a run with a T11SU value of $\frac{1}{3}$ is considered to deliver no reasonable results to the user (Soboroff et al., 2012, p. 6).

3.3.6.5 Sensitivity and specificity

Sensitivity and *specificity* are additional performance measures that can be derived from the contingency table 3.4. Sensitivity is in fact the same as recall, but the difference in this dissertation (as previously mentioned) is that the recall values are not based on the sum of true positives and false negatives. Therefore sensitivity is introduced as calculated using formula 3.28 (Schütze et al., 2009, p. 162). Sensitivity is also referred to as the *true positive rate*, i.e.

the proportion of found relevant documents within the entirety of all relevant documents.

Specificity in contrast is defined as the *true negative rate*. In general this figures is not very useful for unranked documents like in an information filtering setting, because there are usually much more irrelevant than relevant documents (Schütze et al., 2009, p. 162). Nevertheless, this value is reported for the *TREC 2011 corpus*, as this corpus explicitly provides numbers for the *true negatives*. Therefore, this number is reported, in order to show how well an algorithm can detect true negative documents.

3.3.6.6 Micro and macro averages

In the evaluation chapter, 68 different search profiles are assessed over two corpora. In addition to a comparison on search profile and profile category level[18], the chapter also presents a comparison between all profiles available for each corpus. This means that the performance figures that were introduced up to now, are averaged over all search profiles. These averages can be calculated in two fashions: *micro averaged* and *macro averaged*.

Macro averages only

(Sun & Lim, 2001, p. 523) defined the micro average as the average calculated using the entirety of all true positives, false positives, true negatives and false negatives that were filtered for m search profiles. Formula 3.37 and formula 3.38 show how precision and recall are calculated. The macro-average is the algorithmic mean over the performance figures of all search profiles (cf. formula 3.39 and 3.38).

$$Precision_{micro} = \frac{\sum_{i=0}^{m}(TP_i)}{\sum_{i=0}^{m}(TP_i + FP_i)} \tag{3.37}$$

$$Recall_{micro} = \frac{\sum_{i=0}^{m}(TP_i)}{\sum_{i=0}^{m}(TP_i + FN_i)} \tag{3.38}$$

$$Precision_{macro} = \frac{\sum_{i=0}^{m}(Precision_i)}{m} \tag{3.39}$$

$$Recall_{macro} = \frac{\sum_{i=0}^{m}(Recall_i)}{m} \tag{3.40}$$

In this dissertation only the macro averaged values are reported, because they are robust against outliers. The micro average numbers are prone to

[18]There are three profile types: atemporal, temporally ambiguous and temporally unambiguous (R. Jones & Diaz, 2007). These are introduced in more detail in the evaluation chapter.

be dominated by single topics, e.g. if there are 20 topics evaluated, where 19 topics deliver reasonable true positives, false positives, etc. rates between 100 to 200 documents per category and the 20th profile has 20,000 false positives, the whole results are spoilt by this single topic. Macro-average is preferred, because the relevance judgements for the used corpora are skewed and the results should provide a consistent view on the performance of the various studied methods. This notion is also supported in (Pestian et al., 2007, p. 99).

3.4 Putting into context – related research areas

This dissertation started in mid of 2009. During this period, content generated by social media platforms like Twitter attracted much attention, which in turn led to major research activities. Hence a large volume of research related to social media, foremost Twitter has been produced, but also research related to other social media platforms like Facebook, Youtube or Google+.

Therefore this chapter will give an overview of research works that is relevant in the context of this dissertation. The goal is to give credit to seminal research work in this area and also to contrast the findings that are presented later on in this dissertation. First, an overview of related research using Twitter is presented, because the corpora used in the evaluation chapter are Twitter based and the aspects that are studied in chapter 8 cover aspects that have been studied in other research works as well. After that a few other research fields are briefly introduced that are also related to text stream analysis, but that are not covered by this dissertation.

3.4.1 Overview of Twitter related research

In this dissertation two Twitter corpora are used to study information filtering methods in high velocity text streams. Twitter is a popular research topic, because it is easy to collect a large amount social media content using its public API. Therefore there is much research related to Twitter. This section describes some research[19] related to the topics of this dissertation are described, in order to contrast the results of this dissertation to previous research.

In 2009 (Sankaranarayanan et al., 2009) presented a system called "Twitterstand" that used a Naïve Bayes Classifier to classify incoming Tweets into junk or not junk. They also exploited additional features like embedded

[19] A search on Google Scholar on 31.05.2015 returned over 13.000 hits for "Twitter microblog". Microsoft Academic Search lists over 1.000 papers for Twitter in the field of computer science research studies

URL or geo-information to cluster incoming Tweets. This system was an elaborated information filtering system focused on Twitter. This system was developed during the infancy of Twitter and focused mainly on the feasibility of a system which produces microblog messages like Tweets. The basic principles e.g. in terms of weighting functions, which are developed in this thesis, could be incorporated in that system and similar systems, in order to improve their performance.

Various other research works focus on the study of the performance of various machine learning for classifying incoming Tweets. (Irani et al., 2010) used the *information gain metric* to reduce textual features for text classification. They studied various machine learning algorithms like Naïve Bayes or *C4.5 decision tree* in terms of their performance of categorizing Tweets. They concluded that decision trees performed better than Naïve Bayes, but argued to use the latter as it does not perform significantly worse, but requires less training time. (Horn, 2010) also studied the classification of Tweets, but concluded that SVM are best suited for this purpose.

(A. Dong et al., 2010) presented an approach on how to exploit Tweets to improve real-time web search. They used the URLs contained in Tweets as well as textual features to train a classifier that was used to improve the time-sensitiveness of the search results from a regular web search engine. (Zubiaga et al., 2015) focused on trending topic detection on Twitter using SVM. Additionally they studied the statistical characteristics of various features from the Twitter stream such as hashtags, document length, punctuation or links, which is similar to what is presented in chapter 7.7.

(Shraer et al., 2013) studied a *top-k* publish-subscribe system to monitor topics on Twitter. They used an incremental corpus and scoring function based on BM25 and a decay-based recency factor.

(Cataldi et al., 2010) introduced an approach to detect emerging topics within Twitter. An interesting aspect of this paper was, that they introduced the concept of *content energy*. The idea is to calculate a *nutrition value* for a Tweet by calculating the product of term score values and the user authority of the creator of the Tweet and then decaying this value over time.

Text stream mining

In (Weng & Lee, 2011) a system to monitor social media streams was introduced that used *Wavelet-based* clustering of words from the Twitter stream. Again the goal was to detect events in the stream, and did not focus explicitly on information retrieval aspects. (Bifet et al., 2011) focused on sentiment change detection in Twitter streaming data. The focus was to validate an algorithm for automatic change detection in a social media stream. Thus, this work did not focus on the information retrieval aspect, but rather on the tem-

poral window and pertaining algorithms. In this regard, this work is relevant in the context of chapter 7.6. (Abel et al., 2012) introduced a system called "Twitcident" that was intended to monitor information related to real-world incidents and crises. They enriched initial search queries with semantic information from *DBPedia* and filtered the Twitter stream on a keyword basis using these queries. Again the focus was not on the information retrieval aspects of the text stream, but on another system that tries to address event detection and tracking on Twitter, what is not the focus here.

Event processing and social media

(Riemer et al., 2012) presented an event processing approach to filter information from social media streams. Their focus was on the integration of background knowledge and semantic relationships into event processing languages[20]. They implemented a tool to formulate semantic queries on a social media stream and provided a visual dashboard to visualize the results. Their work is similar in terms of the application of event processing to social media streams, but unlike this dissertation, they did not focus on information retrieval specific questions.

Finally, (Albakour et al., 2013) presented a research study that also used the TREC 2011 corpus and was concerned with filtering rather than retrieval from this corpus. They used an incremental Rocchio algorithm to expand the initial query to overcome the sparsity of the profiles. This aspect is also studied in this dissertation in chapter 8.5. Furthermore an incrementally updated logistic regression classifier was used to classify the incoming Tweets. Classifiers will also be used in this dissertation, but the feature engineering is different as the features in this dissertation are based on sliding windows while in (Albakour et al., 2013) they were incremental. They addressed topic drifting by attributing different weights with long-term and short-term interests. The first can be considered as being an incremental corpus based term scoring, while the latter is based on score values based on n recent Tweets or the values from the last day. In this sense the approach is similar to using a temporal window like in this dissertation, but they did not provide any findings on the dimensioning of such windows.

In summary, each of the previous mentioned research works covered related aspects of this dissertation – in terms of social media text stream analysis – like scoring, sliding windows or event processing in social media streams, but none of them covers the aspects that are addressed in this dissertation in the same depth (cf. 1.3) in the same depth or scope.

[20] A similar approaches were also presented in (Barbieri et al., 2010) or (Teymourian & Paschke, 2010)

3.4.2 Real-time search

In 2.5 the term real-time was already discussed. In terms of information retrieval and high-velocity text streams it is logical to think of *real-time search* as valuable concept to find information in such text streams.

In contrast to information filtering, a real-time search engine is focused on information retrieval, i.e. it focusses on finding information in an incremental corpus. The major challenge for such search engines is to retrieve information in *real-time* in the sense that new information is immediately indexed and available for querying almost instantaneously after its creation (Busch et al., 2012). Such kinds of real-time search engines like *topsy.com*[21], *48ers.com*[22] or the search engine used by Twitter to find recent Tweets are examples of how such real-time search engines work. The *Google Real-time search* was also notable in this context. This was a feature of the Google search engine active between 2009 and 2011. It blended real-time content from Twitter or Facebook directly into the regular web search results. After the deal with Twitter expired the feature was deactivated (Wikipedia, 2015b).

3.4.3 Top-k filtering and retrieval

Top-k retrieval is a natural way in information retrieval to process retrieved documents. People usually tend to examine only a subset of the results delivered by a retrieval system, e.g. for web search engines (McGee, 2011) summarised that 52% of Google's and 26% of Bing's user clicks happen on results found on the first page. In (Qin et al., 2012, p. 10) top-k retrieval is defined as "each result is associated with a score and the k results with largest scores are reported as the top-k results".

In data and text stream processing top-k retrieval has been a relevant topic for several years. The goal is to maintain a continuously updated list of top-k items, i.e. a new item is checked against the existing list of top-k items, and if it is within the scoring of the top-k elements, it gets inserted in this list and pushes the least scored item out of the list of top-k items. Formally top-k queries and their results are defined in (Vouzoukidou et al., 2012, p. 1067) In practice, top-k retrieval resembles to maintaining a dashboard of top-k items. So it is obvious that this is closely related to information filtering.

Notable research regarding real-time top-k filtering has been presented in (Vouzoukidou et al., 2012), (Mouratidis et al., 2006), (Haghani et al., 2010),

[21] `http://www.topsy.com`
[22] `http://www.48ers.com`

(Haghani et al., 2012) and (Haghani et al., 2009). The approach presented in (Haghani et al., 2010) tries to efficiently filter incoming documents for a given profile. In this regards their work is similar to the one presented here. The main difference and contribution in this dissertation is that text streams are studied from an information science rather than from an algorithmic perspective, i.e. in contrast to (Haghani et al., 2010) the focus is to study a broad variety of term weighting, thresholds and relevance feedback methods rather than to evaluate optimal algorithms in terms of memory usage or computational complexity.

3.4.4 Topic detection and tracking

Topic Tracking and Detection(TDT) is concerned with the detection of unknown events in a stream, i.e. there are no query terms or search intentions provided to the system in charge, but the system autonomously detects potential real word events in the stream. Commonly, five challenges are addressed within TDT (Fiscus & Doddington, 2002, p. 2):

1. Story Segmentation
2. Topic Tracking
3. Topic Detection
4. First-Story Detection
5. Link Detection

In a nutshell TDT is about event detection. But the term is not used in the sense of the event processing approach used in this dissertation, but in the sense of real-word events, each of which consists of several coherent single events. In TDT such events are called *topics*. (Allan, 2002, p. 19) defines a topic "a seminal event or activity along with all directly related events and activities". This dissertation will not be focused on discovering unknown events. For further reading relevant basic research papers for this topic are (Allan, Carbonell, et al., 1998),(Allan, Papka, & Lavrenko, 1998) or (Yang et al., 2000).

Burst detection

Closely related to TDT is *burst detection*. (Kleinberg, 2002, p. 1) argued "that the appearance of a topic in a document stream is signalled by a 'burst of activity,[sic]" with certain features rising sharply in frequency as the topic emerges". Thus, burst detection can be considered as an extension to TDT, which is not only focused on bursty topics.

There are several examples for burst detection. A recent one was presented in (Nikolov, 2012). He applied a non-parametric way of identifying trend-

ing topics in Twitter. He measured the distance of the *shape* of the current stream to previously recorded patterns for trending and non-trending topics and used the distance to indicate a trend. Other approaches were for instance presented by (He & Parker, 2010) and (Kleinberg, 2002). The first used the physical concepts of *mass* and *velocity* and modelled burst as a dynamic phenomenon instead of using arrival times of terms. The second applied an infinite-state automaton to model the burst of events as a natural state transition.

In general, all these areas such as event and topic detection are different from the research questions in this paper, because the former topics are always concerned with detecting new and unknown real word events from the stream, like a bomb explosion in a major city. There are no defined queries or dedicated information needed, except the interest in something new. The research in this dissertation relates to information filtering, i.e. a user's information need is expressed in some way and the system tries to filter relevant documents, regardless of whether the information need is to find bursty, new information or to find "boring" non-trending information.

3.5 Summary

In this chapter the basic principles and relevant concepts of information retrieval and information filtering have been introduced. First, the difference between the two areas information filtering and information retrieval has been discussed. It has been shown that even though both areas share common subjects, the information seeking process differs significantly, because information filtering is focused on discerning relevant information from non-relevant, while information retrieval tries to return the most relevant documents for a search intention. Furthermore, the distinction between *new* and *classic* information filtering has been elaborated. The goal has been to highlight the difference between information filtering using relatively slow information sources like magazines or newspapers (classic information) and high-volume, high-velocity text sources like social media applications (new information filtering). It has been argued that the high-volume, high-velocity setting is a new context that requires the re-evaluation of existing information filtering concepts as well as the study of new methods that explicitly address this context. Additionally, information retrieval concepts have been introduced that are relevant in the context of this dissertation. This included term weighting schemes like BM25 or Language Models, thresholds, query expansion and relevance feedback. All the introduced methods represent a sound baseline for being re-evaluated in the high-velocity text stream con-

text. Furthermore, they act as references to new methods that are presented later on. The chapter has been concluded by a brief discussion about relevant related research topics, in order to differentiate them from the subject and goals of this dissertation.

Part II

Describing a reference framework

Chapter 4

A reference model for event based text processing and information filtering

As stated in the introductory chapter, event processing provides means that can help to analyse high-volume text streams. The problem with this approach is that these topics have not yet been studied jointly in a scientific way. As a consequence, there are no fundamental artefacts in place to operationalise text streams for event processing. This and the next chapter address two essential artefacts: a domain-specific reference model and a domain-specific reference architecture.

This chapter introduces a domain-specific reference model is introduced that relates relevant concepts and entities from the information retrieval, information filtering and text processing domain to event processing. The goal of the reference model is to provide a common vocabulary and a common understanding of the relationship of the events required to design event based information filtering and text stream analysis systems.

Goals

The chapter consists of two main sections. Section 4.1 introduces basic principles and characteristics of reference models. Then methods for evaluating reference models are introduced, followed by a design process description. The section is concluded with the presentation of other event processing related reference models. Section 4.2 describes the proposed reference model in detail. The reference model is designed by applying the previously introduced process, and its components are depicted in more detail. The reference model is structured along four dimensions that contain different, semantically unambiguous event classes.

This chapter serves two purposes for the overall structure of the disserta-

tion. First, in terms of information science focused research, it addresses the genuine information scientific objective of modelling and representing information and knowledge. Second, in terms of the *Design Science* approach, this chapter addresses the requirement to define constructs and models (March & Smith, 1995, p. 255ff).

4.1 Reference models in general

In this section basic principles of reference models are introduced. The goal is to provide useful information, which characteristics reference models should fulfil, how they are designed and evaluated.

4.1.1 Characteristics of reference models

In (OASIS - Organization for the Advancement of Structured Information Standards, 2006, p. 3) a reference model is defined as

> ...an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details.

Properties of reference models

This definition contains several aspects that are essential properties of a reference model. Firstly, a reference model is about entities. An entity in computer science and specifically in data modelling is defined as "a 'thing' which can be distinctly identified. A specific person, company, or event is an example of an entity" (P. P.-S. Chen, 1976, p. 10). Secondly, it describes significant relationships between entities, i.e. a reference model does not contain every possible detail, but abstracts and includes only the relevant ones. Thereby it facilitates, thirdly, the development of new architectures by providing, fourthly, only the minimum of required concepts within a specific domain (fifth aspect). The sixth property is that a reference model does not require a specific technology or is designed towards a certain software stack. The reference model is supposed to be technology *agnostic*.

Similarly, (Fettke & Loos, 2004, p. 331) defined a reference model as a set of actions that intend the creation and application of reusable models. This

means reference models should be designed in a way that they can be re-used for different purposes within a certain domain. Re-usability as an essential aspect of reference models was also put forth by (Esswein et al., 2004, p. 289) and (Matook & Indulska, 2009, p. 61) and requires that a model should provide a "certain level of abstraction ... [in order to] have the potential to create more specific models and facilitate re-use" (Matook & Indulska, 2009, p. 62).

This requirement is referred to as *generality*. Generality means that a reference model should be adaptable by different organisational units within an enterprise[1]. Additionally, this aspect is only valid within a certain *scope*, i.e. the reference model should not be too broad and should address a specific domain, allowing a variety of users to apply it to different use cases (Matook & Indulska, 2009, p. 62).

Properties in detail

In addition, (Matook & Indulska, 2009) and (Esswein et al., 2004) elaborated further characteristics of reference models based on seminal research in the reference modelling domain. *Flexibility*[2] emphasizes the capability of a reference model to adapt to new requirements within a given domain. *Completeness* describes the demand towards a reference model to cover all relevant aspects in terms of a predefined scope. In terms of scope (Fettke & Loos, 2002, p. 8) define three levels. An *elementary model* provides information about single entities or object classes and is the smallest scope level. An *enterprise model* has the biggest scope and is often used to model entities within a complete enterprise. For large enterprises, such models can contain several thousand entities and even more properties. A *domain model* is in between the two previously mentioned models in terms of size and scope. It only covers certain aspects such as processes or entities.[3] This also implies that reference models can have a different degree of detail depending on their purpose and that there is no minimum or maximum number of artefacts that a reference model must provide. *Understandability* highlights the goal of a reference model to be implementable by users. This means a reference model should provide enough information while being not too complicated, in order to allow users of the model to derive their own models from the reference model (Matook & Indulska, 2009, p. 62). (Esswein et al., 2004, p. 289) referred to this aspect as *extendibility*. *Usability* in this context means the ease of use – analogous to user interfaces, i.e. a reference model should provide enough understandable artefacts that an architect or developer can use the model for various purposes like software development, documentation, data mod-

[1] Reference models are often designed by companies and are well studied in the context of Information System research. Therefore organisational and economic aspects are relevant. More details later in this section

[2] Also referred to *adaptivity* by (Esswein et al., 2004, p. 62)

[3] The reference model presented in this dissertation are domain models (cf. section 4.2.1).

elling or business processes (Frank, 2007, p. 124). Finally, *acceptance* refers to the requirement or rather to the possibility of a reference model of being accepted and adapted by other users than the initial target group.

These requirements sound often rather similar to a certain degree and thus it is also logical that these characteristics are highly interdependent, i.e. that the quality of one characteristic has direct influence on the quality of another characteristic. Hence it is important to consider the interaction of the various properties during the design process (Esswein et al., 2004, p. 291ff). Figure 4.1 shows the interaction of the various characteristics. In summary, the fol-

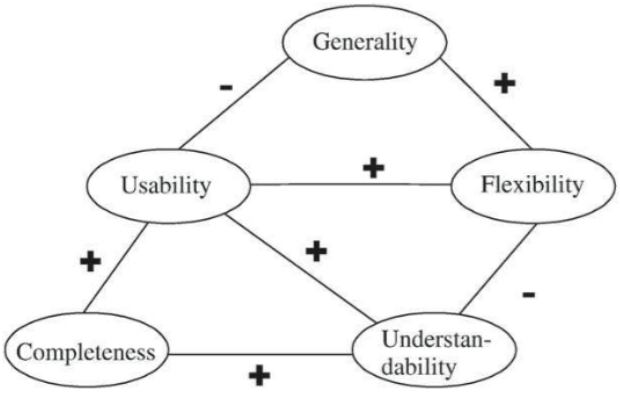

Figure 4.1: Interactions of reference model characteristics according to (Matook & Indulska, 2009, p. 63)

lowing seven characteristics were taken into account during the design of the reference model based on (Matook & Indulska, 2009, p. 62) and (Esswein et al., 2004, p. 289).

1. Generality
2. Flexibility
3. Completeness
4. Usability
5. Understandability
6. Re-usability
7. Acceptance

Another definition by (Bass et al., 2003, p. 48) also stresses the same aspects as just introduced in more concise manner. They defined a reference model as

> ...a division of functionality together with data flow between the pieces. A reference model is a standard decomposition of a known

> problem into parts that cooperatively solve the problem. Arising from experience, reference models are a characteristic of mature domains.

Also relevant in the context of reference models is the concept of *Domain Driven Design* which was introduced by (Evans, 2004) and where domain models from an integral part of the design methodology. He defined a domain model as

> ...the core of a common language for a software project. The model is a set of concepts built up in the heads of people on the project, with terms and relationships that reflect domain insight. These terms and interrelationships provide the semantics of language that is tailored to the domain while being precise enough for technical development. (Evans, 2004, p. 23)

By this definition it becomes clear that for a successful software project a common language is required and this common language can be derived the application of reference models.

Even though a set of characteristics can be identified, (Fettke & Loos, 2004, p. 332) argued that the term *reference model* was vague and space for interpretation was left. Therefore they proposed a systematisation of different notions of the term that is shown in figure 4.2[4].

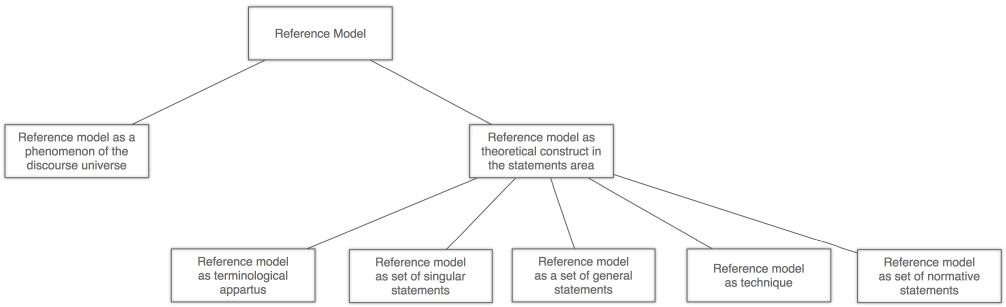

Figure 4.2: Systematisation of various notion of reference models

Terminological apparatus

According to the definition given by (Fettke & Loos, 2004), the reference model proposed in this dissertation belongs to the category *reference model as terminological apparatus*. They described this type of reference model as a collection of terms contained within a conceptual frame of references (p. 332).

[4]For further details on the other meanings cf. (Fettke & Loos, 2004, p. 332f)

Examples Renowned examples of reference models are for instance the OASIS SOA reference model (OASIS - Organization for the Advancement of Structured Information Standards, 2006), Reference Model of Open Distributed Processing (Wikipedia, 2015i) or the *Open Systems Interconnection model* (OSI Model) (International Organization for Standardization, 1994).

In summary, the properties and characteristics presented in this section were taken into account during the design process (cf. next section), because they were introduced in definitions from the *Organization for the Advancement of Structured Information Standards* OASIS and from renowned researchers in the field of reference modelling[5].

4.1.2 Evaluation of reference models

Evaluation A reference model also needs to be evaluated, in order to assess the quality of the model. Therefore approaches for evaluation are discussed and an evaluation approach for the proposed reference model is selected.

(Frank, 2007) presented a multi-perspective evaluation framework for reference models. He defined four perspectives (economic, deployment, engineering and epistemological), in order to increase the objectivity of model evaluation by looking at a reference model from different points of views. For each of these categories he provided criteria to assess the quality of a model, but also mentioned that not all criteria or perspectives must be used for an evaluation and are only intended to provide guidance through the evaluation process (p. 123). Thus, the proposed criteria can be used to derive questionnaires, surveys or interviews and could also be used to derive computable metrics. But there are no authoritative or standardized questionnaires or metrics[6] that could be used for the evaluation of the proposed model. Therefore a few criteria presented in his paper will be used for the design of an evaluation questionnaire. Questionnaires and surveys were also proposed as a suitable evaluation method by (Matook & Indulska, 2009), (Ahlemann & Gastl, 2007, p. 91ff) and (Esswein et al., 2004).
(Fettke & Loos, 2003) also presented a multi-perspective evaluation framework and also argued to evaluate a reference model using different points of views, because there is no single best evaluation approach (p. 88). They provided an exhaustive list of potential evaluation methods and categorised

[5]Prof. Dr. Peter Fettke and Prof. Dr. Peter Loos are researcher at the *Deutschen Forschungszentrum für Künstliche Intelligenz* (DFKI), who have been doing research in the area of reference modelling for several years. (Matook & Indulska, 2009, p. 61) also highlighted the importance of German researchers in this field, to which the aforementioned researchers belong. Thus it is reasonable to base the proposed reference model on their findings.

[6]This can be subject to further research, but is out of scope for this dissertation

them into *analytical* and *empirical* evaluation methods. The analytical ones for instance comprised metric-base, feature-based or meta model based methods. The problem with these methods is that there are hardly any real-world instantiations of such methods that could be used to evaluate reference models. The empirical approaches comprised surveys, laboratory experiments, field and case studies (p. 86ff). But also the empirical ones showed several drawbacks. For instance, the problem with surveys is the low response rate; laboratory experiments are often too artificial(p. 87). A compelling approach is based on *case studies*. (Fettke & Loos, 2003, p. 88) described it thus:

> Unlike in the laboratory experiment and the field study, the researcher does not try to measure influences of independent variables or to control intervening and confounding variables. Instead the practical solving of a specific reference modeling[sic] problem is observed. The case study describes the addressed reference modeling[sic] problem, proposed solution and gained results.

Even though they criticised the limited objectivity and a weak generalizability, this approach together with a questionnaire is the most suited for the context of this dissertation. Also (Matook & Indulska, 2009, p. 64) also proposed to conduct *exploratory case studies* to evaluate the quality of a reference model.

Tailoring

Another aspect that must be considered is that reference models are often designed and implemented in companies, in order to decrease costs, facilitate compliance requirements or standardize development efforts (Matook & Indulska, 2009, p. 60). Also, reference modelling is an area of research that is driven by requirements from *Business Information System* research and thus considers economic and organisational requirements. These are relevant aspects for the evaluation (Frank, 2007), but this dissertation is not an Information System thesis and was not conducted within a company. Therefore economic and organisational aspects are not a focus of the evaluation and are only considered were appropriate, e.g. a few questions within the evaluation of the reference model addressed the economic perspective.

4.1.3 Design process and modelling guidelines for reference models

The previous section discussed basic properties of reference models as well as evaluation approaches. This section focuses on the design process and on

modelling guidelines for reference models. This order was chosen, because the author thinks that it is easier to describe the *How* after the *What* has been defined.

Design process

Process choice

The modelling process was guided by principles presented in (Esswein et al., 2004) and in more detail in (Matook & Indulska, 2009), who synthesized seven core process steps from seminal approaches in reference modelling (Matook & Indulska, 2009, p. 61). The reason to choose this design was based on three factors. Firstly, they conducted an extensive literature study and scientifically compared various seminal approaches to reference modelling. Secondly, they took the results from the literature analysis and derived a best practice approach. Thirdly, it is a relatively recent research paper, so it is valid to assume that it covers the most recent developments [7]. The following process steps and their definition below were described in (Matook & Indulska, 2009, p. 61f) and are now explained in detail.

1. Problem definition
2. Requirement analysis
3. Information gathering
4. Setting conventions
5. Documentation
6. Construction
7. Evaluation

Problem definition In this phase the purpose of the reference model should be defined, starting with what artefacts the model should contain. This means for instance to define whether the reference model contains process models, functional models, organisational models or data models and also whether it contains one, two or all model types. Secondly, the previously introduced characteristics addressed by the reference model are enumerated. Again this aspect is tailored to the required characteristics. Thirdly, the scope of the reference model is defined by a domain expert who states which areas are covered and which are not (p. 61f)[8].

[7] Their process design was tailored to the specific context of this dissertation. This became necessary for the same reason mentioned previously in the evaluation section.

[8] The tasks of the domain expert are performed by me, as I consider myself proficient enough for this task due to a Master degree in Information Science and several years of experience in the IT business.

Requirement analysis This phase defines the level of granularity a model should have. Also the appropriate modelling language should be defined and requirements based on existing reference models should be discussed. The whole process should yield an estimation of the supposed cost to design the model (p. 62).

Information gathering information gathering describes the process of rating and relating of information sources. This means the goal is to gather relevant sources, and to show on what literature, research or other models the reference model was built.

Setting conventions Setting conventions is the step to agree on and implement mandatory agreements for the design process. This also includes glossaries and common terminology. Thereby the development of the reference model should be facilitated.

Documentation In general, documentation should happen throughout the design process. But an explicit requirement phase helps to assure that the information is complete, in order to progress with the available set of information to the next stage of the process.

Construction The construction phase is the penultimate process step. This step represents the actual design of the reference model based on the result, conventions, goals and agreements from the previous stages.

Evaluation Finally, the evaluation phase takes place that assesses the quality in terms of the fulfilment of the requirements posed by the reference model characteristics, the model's scope and also of the target group of the model. The evaluation process was already introduced in the previous section.

Modelling guidelines

The *Guidelines of Modelling* as described in (Schütte, 1998, p. 119ff) and (Becker, 1998, p. 3ff) defined a regulation framework for the modelling of reference models. The guidelines provide a set of statements that support the reference model design in attaining the properties that were introduced in the previous section.

There are six guidelines in total (Becker, 1998). The first guideline is the *principle of correctness* that states that the model should represent the subject matter in an adequate manner (Becker, 1998, p. 4). This principle is not about formal correctness in a mathematical or an absolute sense, but it focuses on the aspect that a model should help to operationalise a given subject matter by providing e.g. naming conventions and structural models. The *principle of relevance* says that a model does not need to reflect every aspect of the real world, but it should reflect all aspects that are defined by the scope of the reference model. The *principle of economic viability* focuses on the cost of creating a model compared to the benefits of its application. This principle does not apply in the context of this dissertation. The *principle of clarity* corresponds with the understandability and usability aspects introduced earlier and the *principle of comparability* requires that models by using different modelling techniques, e.g. *Unified Modelling Language* (UML) activity diagrams vs. Petri nets, can be compared. This means the designer should adhere to the standard of the applied modelling language so that models can be converted forth and back for instance by using a meta relationship model (p. 7). The last guideline is the *principle of systematic structure*. This guidelines requires that concepts within different views of a reference model are consistent, i.e. if a reference model contains process models and data models it must be ensured that the process models only use data entities that were actually defined in the data model and does not operate on non-existing entities.

In summary, the introduced properties of reference models and guidelines for reference modelling along with process steps for reference modelling, can be considered as a sound base for the design of the reference model that is presented in section 4.2.

4.1.4 Reference models in event processing

Before the reference model is introduced, a few, concrete examples of relevance models in the event processing domain are given.

The *Conference on Distributed Event-Based Systems* (DEBS) is a conference dedicated entirely to the research of event based systems. At the first DEBS conference in 2007 already (Rozsnyai et al., 2007) argued that a general event model is required to facilitate the design and implementation of event based systems. They presented an *event object library model* that defined the basic syntactic requirements that an event should fulfil, in order to be processable by event processing systems. Based on these requirements, an *event object type* (=event), which is contained in the event object library, should have the following items(p. 65):

- Event object type namespace.
- Event object type name and display name.
- Event implementation type.
- Attributes of the event object type.
- Parent event object type (in case of an inheritance).
- Event object types which are virtual roots (in case of an exheritance).

Syntactic reference models

The goal of the event object type description was to facilitate the implementation using certain event stream or complex event processing tools. It was not focused on any domain-specific needs and can therefore be considered as a *syntactic reference model*.
The need for syntactic reference model and the definition of basic and foremost common properties of an event was also supported in (Hinze, 2003, p. 29ff), (Etzion & Niblett, 2011, p. 62ff), (Westermann & Jain, 2007) and also in (Stuehmer, 2014), because thereby a common structure can be achieved that facilitates implementation, discussion and comparison. Therefore these aspects are incorporated in the domain-specific reference model that is presented in this chapter. (Westermann & Jain, 2007, p. 25) argued that "[a] common event model should be able to express events' temporal and spatial relationships as well as their structural and causal relationships". Figure 4.3 summarises various aspects that events in a reference model should cover (Westermann & Jain, 2007, p. 23).

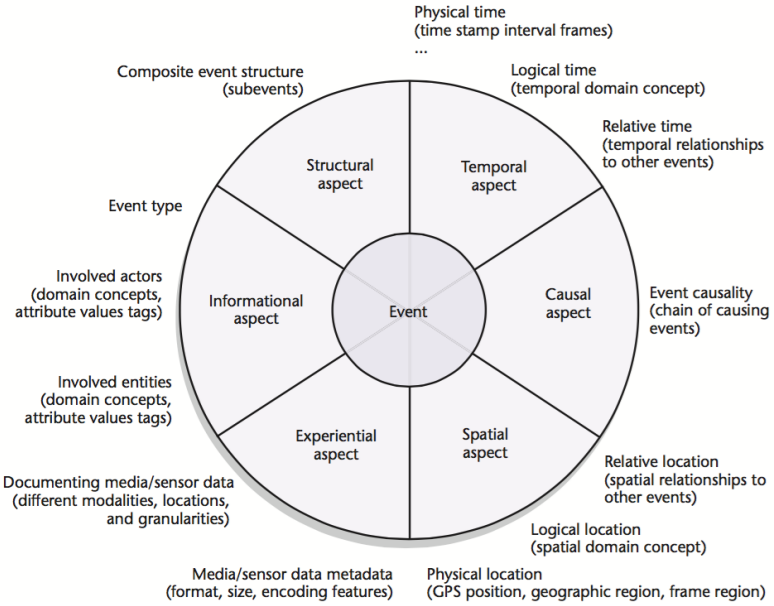

Figure 4.3: Basic aspects of events.

Dagstuhl 2010

At the *Dagstuhl* seminar in 2010 the *event processing manifesto* was adopted

(Berry et al., 2011). Influential researchers in the domain of event processing discussed the current state of event processing and presented their view on topics like characteristics of event processing, relation to other areas or event processing standards. The latter were addressed by the proposal of an *event processing standards reference model* (Berry et al., 2011, p. 41ff) that related aspects from the *model-driven architecture* domain, like *computer independent model* (CIM) or *platform independent model* (PIM), to domain use cases and domain reference models. They argued that such models should be developed "to enable visibility in companies and show the potential of event focus and EP as well as providing guidance on the implementation of EP applications" (p. 47). These are also the reasons why a domain-specific reference model is developed in this dissertation.

There are already various reference models for event processing, which are now introduced briefly. A generic approach to model event processing systems is presented in (Zang & Fan, 2007) where a formal meta model for events is suggested providing a description of the intrinsic relationships between events operators, context and processes. This model is aimed at unifying the different components used to model *Event Processing Networks* (D. Luckham, 2002, p. 207ff). It is not limited to a specific domain, but is used as the foundation of an event processing engine implemented using this meta model. The notation is UML-based, which makes the model generally understandable by IT people.

(Springer et al., 2009) also used a UML-based notation to design reference models for the retail and finance domain. Additionally, they included concepts from the *Models to Text Transformation Language* (MOF), the *XML Metadata Interchange* (XMI) and the *Notification Event Architecture for Retail* (NEAR). Their goal was to introduce a standardized approach to model event based business process management systems and to underline the requirements for standards in event based modelling. Unfortunately, there is still no standard with industry-wide acceptance in place, but their work shows that using already accepted standards like UML is a viable way to model event processing systems and thus UML was also used in this dissertation.

The model presented in (Hummer et al., 2012) is more detailed than in (Zang & Fan, 2007) and shows how components in an event processing system are related to each other. Therefore it can be considered as a sound source of inspiration for deriving a reference model. In (Hinze, 2003, p. 28f), a reference model for event notification was presented. This model headed into the direction of information filtering, but is not concerned with domain-specific problems of text stream processing such as e.g. the semi-structured and ambiguous nature of text. Hinze's model contained concepts such as profiles

and event repositories, which are also relevant for designing an event based information filtering system.

(Dunkel et al., 2009) and (Renners et al., 2012, p. 3) presented a domain event model for sensor events. Structurally this research is similar to the findings of this dissertation in terms of the goals, but the focus here is on information filtering and text stream processing.

4.1.5 Standards in event processing modelling

Finally, a look on potential standards or better quasi-standards in the domain of event processing relating modelling and design is taken. The goal is to illustrate how many different approaches exist in terms of event processing related modelling and that there is no single approach with industry- or research-wide acceptance.

EPTS

Prolific efforts in terms of reference modelling and reference architecture in the area of event processing have been made by the Event Processing Technical Society (EPTS) and its members. For instance, (Paschke et al., 2011) presented an exhaustive overview of potential standards for modelling in the area of event processing and consolidated several approaches. Thereby, they introduced a *CEP Standards Reference Model* that should help "...to understand the multiple viewpoints and business and technology areas to which standards may be applied" (Paschke et al., 2011, p. 129). In fact, they consolidated a set of essential artefacts that are beneficial in the area of designing event processing systems. This included domain reference models, domain use-cases or functional models. A domain reference model in their sense is not only focused on the entities and the *common vocabulary*, but also includes best practices and business process. The reference model that is presented here also has this scope, i.e. a set of reference processes and the domain-specific entities are described. Figure 4.4 shows the aspect they addressed in their proposal. Additionally, they detail the reference model with a *computer independent model* that "describes the business functionality or system without reference to IT specifics" (p. 131). This comprises *standard vocabulary*, *process descriptions* or *complex event definitions*. All of these aspects are addressed here with the event reference model. Additionally, they introduced a *platform independent model* that "represents behaviour, data, design, and messaging, independent from a particular EP platform" (p. 131). Thus, the reference architecture, presented in the next chapter can be mapped to this area of their model. Finally, they discussed existing standards for each of the aspects. Again, this showed that there is no single, accepted approach. Instead, UML is often mentioned as a basis, which is sometimes extended

by some specific event processing additions. For designing event processing systems they also refer to *ISO/IEC 42010:2007 Recommended Practice for Architectural Description of Software-intensive Systems* (ISO, 2011), which is also used as the base for the reference architecture in Chapter 5. In summary, their exhaustive comparison supports the notion that standards like UML and *International Organization for Standardization* (ISO) recommendations are valid tools to design event processing systems. That is why these tools were also used in this dissertation.

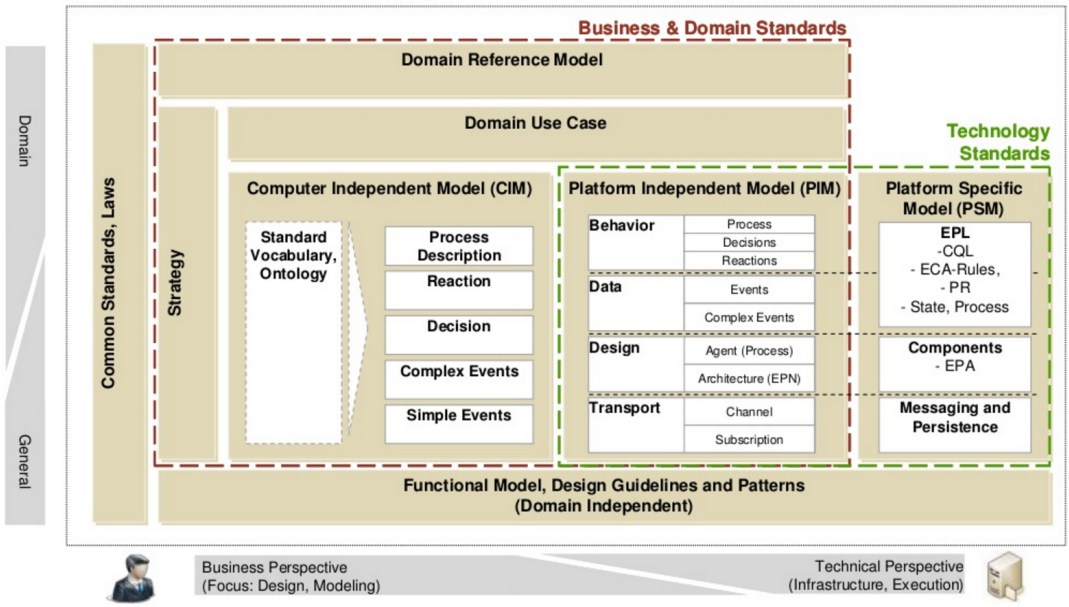

Figure 4.4: Complex Event Processing Standards Reference Model according to (Paschke et al., 2011, p. 129).

In (Springer et al., 2009) existing and future standards for complex event processing were discussed. They presented an approach to extend the *Networked Software and Services Initiative* (NESSI) Open Service Framework Reference Architecture to support event-driven business process management. They also used UML related notation, thus, the choice to use this notation here as well is again supported. Nevertheless, no evidence was found that the extension of NESSI was concluded. Thus, the ideas presented there can be seen as a source of inspiration, but do not represent an industry standard.

4.2 A reference model for event-driven text processing

In the previous sections the relevant properties of reference models were introduced as well as a design process, modelling guidelines and an approach to evaluate the model. In this section these principles are applied and an reference model for event-driven information filtering and text stream analysis is presented.

4.2.1 Problem definition and scope

The reason for the design of the reference model is that text stream analysis, information filtering and event processing have yet not been studied jointly. But as pointed out earlier, event processing provides efficient means to analyse high volume data streams and therefore it is promising to apply event processing to process high volume text streams as well. In general, to be able to process a data stream using event processing means an event model for the subject domain needs to exist. This is because without a semantically precise definition of the required event classes, the input and output of event processing components become arbitrary and the results generated by the components of the event processing network are neither comprehensible nor reproducible.

The case for semantic mapping

In principle, domain-specific concepts have to be mapped onto a distinct event class[9] and their relationships have to be defined, because this

1. allows to clearly distinguish between the semantics of each event
2. provides a common vocabulary for discussing the goals of a system
3. facilitates assigning events to a certain purpose within an event processing system
4. facilitates building event patterns
5. fosters re-use and extension of event definitions and event patterns

The scope of the reference model is to provide a *terminological apparatus* (Fettke & Loos, 2004, p. 332) for text processing and information filtering concepts that are required to achieve the just introduced goals by using event processing means. As stated in the introductory chapter the scope of this

Scope of RAs

[9] This argumentation is supported by an article from mid of January 2013 by David Luckham and Roy Schulte, two renowned researchers in the field of event processing (D. Luckham & Schulte, 2013, p. 4).

dissertation is on high-volume text streams where text documents are rather short such as Tweets or comments. Thus, the dissertation mainly introduces entities, as well as their relationships and the hierarchical grouping that are required to address this special domain subset. Nevertheless, the models also contains a few event classes that are not required in the context of this dissertation, but could act as examples of how to extend the model.

Also in scope are various process models that describe relevant tasks for information filtering using the reference model. These process models describe the flow of the events as well as the basic steps, and the goal is to facilitate the implementation of event-driven information filtering systems. Based on the description of information filtering and retrieval from Chapter 3.1, a set of reference process models is defined that are essential for an event based information filtering system. This process models only reflect the basic flow events, and provide a high-level depiction of steps required to accomplish the defined task. More details on the processes are provided in the reference architecture section, which contains more concrete on how to implement an event based information filtering system; the reference model is focused on providing a semantically concise description of the overall applicable domain concepts.

Relevant tasks

Based on the basic principles of information retrieval and filtering[10] as well as using the ACM CCS[11] the following processes were identified as relevant and are therefore studied in this and the next chapter. Processes were combined where a topical closeness is given, e.g. relevance feedback and query expansion are closely related, because query expansion incorporates the relevance information from relevance feedback.

- Document pre-processing and index maintenance
- Stream features modelling
- Scoring and term weighting
- Relevance feedback and query expansion
- Document filtering and classification
- Filter policy generation

Based on the categorisation of reference models presented in (Fettke & Loos, 2002, p. 8) the here introduced entity reference model and the process models form a domain model.

Stakeholders

Finally, the stakeholders of the reference model need to be defined. As this

[10] The selection of tasks was guided by recent seminal works in information retrieval like (Croft et al., 2010), (Schütze et al., 2009) or (Büttcher et al., 2010).

[11] Tasks were selected from the ACM CCS (version 2012 `http://www.acm.org/about/class/class/2012`) concepts in section *Information systems Retrieval tasks and goals*.

model has not been designed within a business context, there are no explicit stakeholders available. Nevertheless, the potential target group of the model is derived from the goals of this dissertation: to facilitate and foster the use of event processing in text analysis and information filtering. Therefore the stakeholders are IT solution designers and IT architects who need to implement text analysis systems from scratch, or who need to integrate textual data into existing event based systems.

4.2.2 Requirement analysis and setting conventions

UML

The *requirements analysis* and *setting conventions* phases are combined here, because their purpose is similar (cf. section 4.1.3). The event reference model is designed using UML 2.0 class diagrams. The reason is that UML is an accepted industry standard for modelling in computer science and software development. Therefore, this notation is suited, because it is widely recognized, facilitates comparison and common understanding. Also, there are several studies that also rely on UML as means for modelling event reference models (cf. 4.1.4).

The process models are designed using data flow diagrams as proposed by (DeMarco, 1979). DeMarco's standard notation will also be used[12] for the diagrams. The reason for this is that in event processing systems events flow between the several components and, thus, a dataflow diagram is a natural fit for this. One of the reviewers who evaluated the reference model and filled out the questionnaire argued that dataflow diagrams are not widely accepted. But for the aforementioned reason I think this is not true. The only point that is arguable is the kind of visual notation that is used. But as the DeMarco was the first to introduce the concept, his notation will be used. Also dataflow diagram were used because of their clear semantics and the concise amount of used visual components. *Business Process Model and Notation* (BPMN) could have been also used to design the reference processes, but its notation contains too many visual components and is too verbose for the purpose of the reference processes. This modelling language could be used to design the concrete process for a concrete system. There its feature richness is beneficial.

Naming conventions

Event classes always carry the suffix *Event* and are written with upper-case, because this highlights their character as a proper concept. The event properties are written in camel case as one word following general UML standard. They are marked as public, in order to avoid the object-oriented modelling

[12] The notation is introduced when the process models are described.

specific need for getters and setters that do not provide any additional benefit in this context. Abstract classes are enclosed by "«. . . »" and all entities that are not an event class have a transparent background, in order to be easily distinguishable from event classes.

The level of granularity for the event reference model and the process models is mostly coarse-grained, in order to preserve flexibility, usability and understandability. The goal is to show the *what*, not the *how*[13]. Nevertheless, in occasions where it seemed appropriate, the granularity of the models is finer. This variation between coarse and fine-grained description is based on (Rolland, 1998, p. 12) who used this scale for the definition of process models.

Event type vs. event class

In terms of the nomenclature used in the reference model, the difference between *event type* and *event class* must be highlighted. An *event type* is used as defined in section 2.2.3, i.e. it groups events of the same functional category (not semantic meaning!) and the same structure. An *event class* in contrast represents a distinct and unambiguous semantic concept, i.e. an event class models the meaning for certain aspects. The event classes belong to one or more event types and inherit their syntactic structure and intent. The term *entity* is sometimes used instead of event class, in order to avoid repetition.

Finally, the names of the event classes are based on their semantics. This is in contrast to business process events, where event classes are used to describe the progress of or the activities within a business process and thus use naming patterns where the event class denominates the action together with a verb in past tense, e.g. *Account Created Event* or *Customer Notified Event*. This naming is not useful in this context of a reference model, because of the different context and the distinct goals (reference model vs. process description).

4.2.3 Information gathering

According to (Matook & Indulska, 2009, p. 62) *information gathering* entails the "relation and also rating of information sources". This aspect has been addressed by the introduction chapters 2 and 3, as well as section 4.1 in this chapter that discussed and presented a wide range of relevant information sources relevant for the design of the reference model. Relevant aspects and concepts of information retrieval, information filtering, text stream analysis, event processing and reference modelling were also discussed.

[13] This is covered in more detail in the reference architecture chapter.

4.2.4 Construction

In this section the actual reference model is constructed. This phase is structured as follows: First, the overall layout of the reference model is introduced. Second, the various components and their constituting event types are introduced and discussed. This also includes defining their most relevant properties and their semantic relationships. The goal is to provide an overview of all event types that are required to address the reference processes which are defined in section 4.2.1[14].

Overview In order to operationalise text for information filtering using event processing, a domain-specific entity reference model and essential reference processes were designed. The entity reference model was designed along four dimensions that were used to structure the concept space. The visual layout was inspired by (Renners et al., 2012, p. 33), but the proposed dimensions itself as well as their content were developed in the course of this dissertation, and thus represent new research aspects.

New aspects

The layout also helps to classify distinct event classes in to homogeneous groups. This grouping is beneficial as soon as the model is about to be extended, because it provides a high-level structure that allows for easier navigation within the model. Along those four dimensions different event classes were defined and placed within each group. The four groups, which are now described in more detail, are referred to as *domains*; their design was evaluated using a questionnaire. They also formed the based for the prototype that was used for the evaluation sections of this dissertation.

- General event domain
- Text event domain
- Meta event domain
- Pattern event domain

Figure 4.5 shows the complete reference model for text processing and information filtering. This figure only shows the high-level concepts, in order to keep the structure clean and concise. Each dimension is discussed in more detail in the forthcoming sections and is then enriched with fine grained event classes.

[14]It should be noted that the application of the reference model requires domain knowledge in terms of event processing, information retrieval and linguistics. The foundations for both have been discussed in chapter 2 and 3 and are not repeated herein.

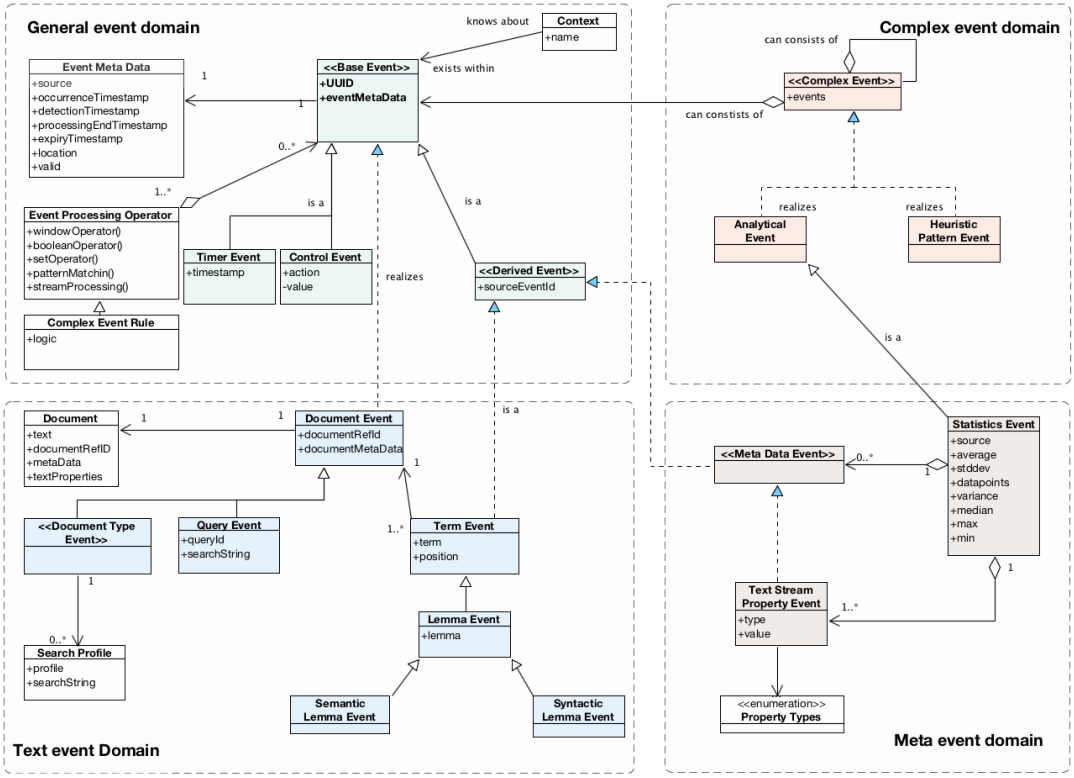

Figure 4.5: High-level view on the reference model.

4.2.4.1 General event domain

The *general event domain* contains the essential concepts that must be present, in order to be able to run event processing. They were derived based on the stipulated characteristics of an event in section 2.2.2, *viz.* temporal unambiguousness, semantic unambiguousness, finality and referenceability.

Base Event

The *Base Event* is an abstract root entity, that is never instantiated, because it does not provide any specific semantic mean. Nevertheless, precise semantic meaning has been previously introduced as one of the reason why a reference model is required. Thus, this entity serves only as a root node in the event hierarchy. Each Base Event has one globally unique id[15] and an entity of type *Event Meta Data* assigned. The characteristics of the Event Meta Data, which are described now are based on (Etzion & Niblett, 2011, p. 66ff), and are used to address two aspects.

First, the temporal aspects of an event, i.e. it contains an *occurrenceTimestamp* and a *detectionTimestamp* to provide means for modelling scenarios where one or both aspects are required (cf. (Etzion & Niblett, 2011, p. 284f)). The oc-

[15]The id is really meant to be unique world wide. This can be implemented for instance using namespaces as proposed in (Rozsnyai et al., 2007, p. 65)

currence timestamp represents the occurrence of the event in the real world, i.e. the kick off at the last soccer world championship, and thus is relevant for relating events during the same time horizon from a semantic point of view. The detection timestamp indicates when the event processing system became aware of the event. This timestamp is used to model technical aspects of an event, e.g. how long it took to process the event. The separation is made, as both do not necessarily coincide. The *processingEndTimestamp* indicates when the event processing system has finished processing the event. Together with the *detectionTimestamp* the technical lifespan of the event can be traced and logged. The *expiryTimestamp* indicates the semantic validity of an event. This means by setting this timestamp an event can be marked as not relevant anymore even though its context, the window it exists in, might still be valid. Second, the geospatial aspects are covered by the *location* property. Depending on the purpose and the available meta data from the raw data source, these can be *Global Positioning System* (GPS) coordinates, or also city names along states or geographic regions. Finally, a valid property is introduced, that can be used to restrict the validity of the event based on some business rules. In terms of the information filtering context, filter policies can invalidate Document Events, so that they are no longer considered within the filtering process.

Further properties

A Base Event can have additional properties in the context of event processing. For the purpose of the given domain, the aforementioned suffice. Nevertheless, for more sophisticated system integration, i.e. an event-driven text analysis system to be integrated into a larger, yet existing event-driven business context, further properties could be relevant. (Becker et al., 2011) presented an evaluation of different event formats for event-driven business process management. They were also unable to identify a single, accepted approach for modelling events, thus the ideas of event formats like *Business Process Analytics Format*, *Extensible Event Stream* or *Common Base Event* (p. 435) can be beneficial, but in the end depends on the scope and the existing prerequisites of the target system.

Timer Event

The *Timer Event* represents the clock within an event processing system. An event processing system is a highly temporal construct. Therefore the progress of the clock used within the event processing network must be represented. The Timer Event can represent a natural clock signal, like those emitted by atomic clocks, or it represents an artificial one whose emittance frequency is not based on natural laws but on some kind of artificial logic. Thereby it is possible to replay or forward within the processing of already stored events. It is crucial to coordinate and synchronize the clocks if there are Timer Event sources, in order to maintain comprehensible and consistent results of the

event processing system (Etzion & Niblett, 2011, p. 291f).

Control Events

Control Events are required to configure the event processing network by adjusting its properties. Such Control Events can be used to start or stop certain event processing agents or they can be used to adjust for instance the size of the sliding window all event processing agents should use.

Derived Event

The *Derived Event* is an essential event type[16]. It also represents an abstract entity that is not instantiated, because it has no dedicated semantic meaning (same as a Base Event) and is only used to indicate that one event was derived from another one. (Etzion & Niblett, 2011, p. 42) defines a derived event as "an event that is generated as a result of event processing that takes place inside an event processing system". It has only one property, *sourceEventId*. This property stores the globally unique id of the event from which an event was derived from. A Derived Event can also be derived from another Derived Event. It does not need to be derived from a Base Event and by using this construction the original event can be recursively re-constructed.

Event processing operator

In addition, there are two more concepts required that were also introduced in (Zang & Fan, 2007) and (Etzion & Niblett, 2011). First, an *event processing operator* represents all event processing related actions that can be executed using an event processing engine. This entails simple pattern matching, windowing mechanisms, and filter and stream calculation. The relationship between operator and event is modelled as a uni-directional aggregation, i.e. the existence of the events does not depend on the existence of the operator, but the operator *aggregates* events. The event processing operator relates multiple events and derives new knowledge based on the chosen operator function. The available functions depend on the chosen event processing engine. Therefore the possibly available operator functions are not discussed, because this can be decided during the design phase of the concrete system. Adhering to the principle of flexibility no restrictions are made here. For reasons of understandability only a selection of operator functions is listed. The multiplicity on the operators side shows that at least one or more operators must exist. This is logical otherwise nothing would happen to the incoming events. The multiplicity on the Base Event side indicates that if an event processing operator exists, it aggregates or relates zero or more base events. A special kind of event processing operator is also shown, *Complex Event Rule*. This subclass represents every more sophisticated approach to event processing like machine learning or complex pattern matching[17].

Context

The *context* defines the "circumstances" in which the event is processed. These

[16] In this case the term *event type* is used as proposed in section 2.2.3

[17] Simple pattern matching refers to detecting of "obvious" event sequences like $ABAC$. In contrast complex pattern matching refers to detecting unknown patterns or trends.

can be temporal, semantic or spatial limits that restrict the scope of an event. This means the context can determine e.g. if an event is processed or if it is completely or partially ignored. (Etzion & Niblett, 2011, p. 145) describes a context in terms of event processing as follows:

> A context is a named specification of conditions that groups event instances so that they can be processed in a related way. It assigns each event instance to one or more context partitions. A context may have one or more context dimensions and can give rise to one or more context partitions.

Each event can belong to various contexts. Thus, each event has a set of context identifiers associated by which each event can be assigned to the correct context.

Figure 4.6 shows the concepts that have been introduced in this section in detail.

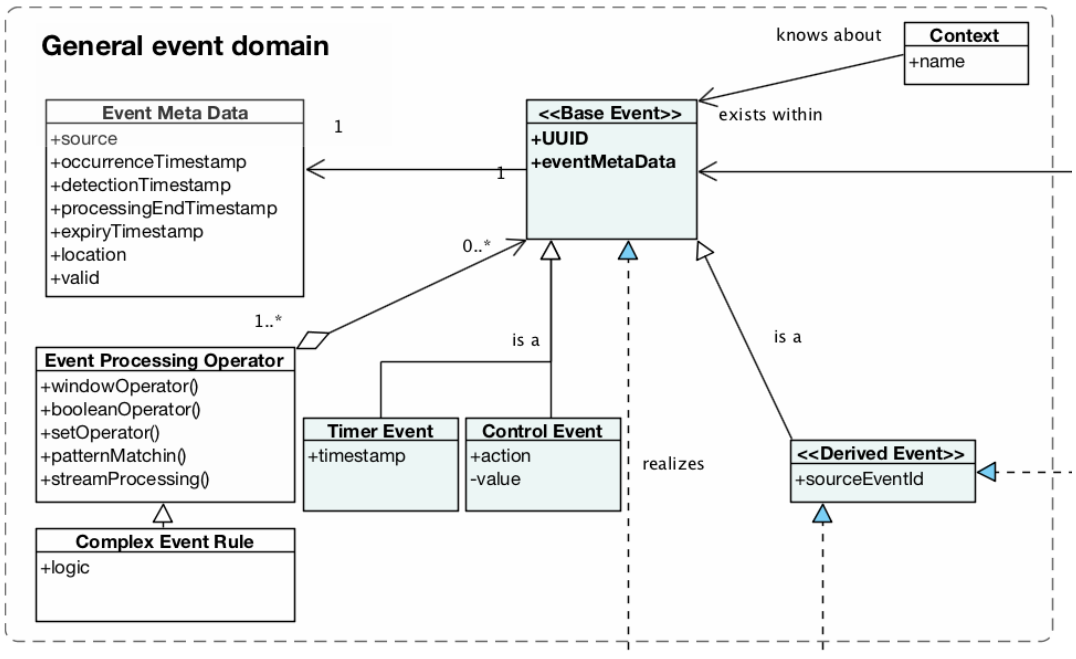

Figure 4.6: General event domain concepts.

In the next step, the relevant event classes for text processing are designed based on the just introduced concepts.

4.2.4.2 Text event domain

The *text event domain* contains all event classes that are required to model aspect from information filtering and text processing as described in section 3.1, i.e. event types are introduced that can be used for linguistic analyses, score calculations, term weighting, relevance feedback and query expansion. A high-level overview of the discussed concepts is shown in figure 4.7.

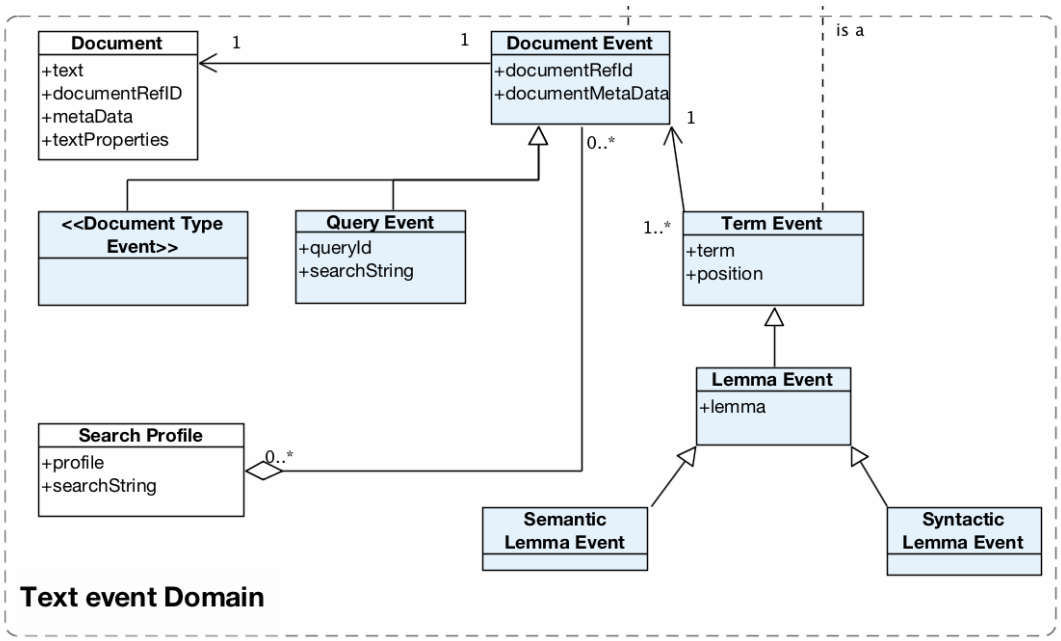

Figure 4.7: Text Event class hierarchy.

Document Event

The central entity for text processing and information filtering is a *document*. Thus, the first event class that is introduced is a *Document Event*. It represents the unprocessed document as it is received by the event processing system. It contains the properties it inherits from the Base Event class together with a *documentRefId* property. The documentRefId represents a unique id of the original document that is associated with the raw document as it is perceived by the system. Thereby, the original text can be retrieved from the document storage whenever required. The document text itself is not an attribute of the Document Event, because otherwise it would be sent around within the event processing network along each Document Event, which would increase the payload within the event processing network unnecessarily.

Dynamic corpora

The Document Event serves as the central class for all further event classes in the text event domain that are derived from it. Additionally, Document Events are also the base for the *dynamic corpora* that are a central subject of investigation in this dissertation. A dynamic corpus is similar to a static or in-

cremental corpus as it contains a collection of documents. But rather than being fixed in sized (static) or continuously adding new documents (incremental), a dynamic corpus only contains as many documents as defined by an expiry policy. In terms of event processing, this means a windowing mechanisms determines how many documents there are within the corpus. As time goes on or new documents arrive, old ones expire and the corpus changes its configuration. In terms of lifetime within the event processing network, after creating all required derived event classes, the Document Event is stored to a data store and only loaded for reference and display purposes. This is because all facets of a document are represented by the events that are derived from the document event class. All derived event class instances carry the unique id of the originator event class, the document event, and thereby the complete document is represented by its constituting parts within the event processing network and could be re-created using only the derived events.

The Document Event does not know about the events that are derived from it. The UML association between the Term Event entity and the Text Property Event entity, the two descendants from this event class, show that it is only navigable from the derived event to the original event. If the Document Event knew about its derived events, it would contain a set of global identifier ids of the derived events. But in this case this is not necessary, as the Document Event only serves as a reference that is not used within the event processing network, but is rather represented by the events that are derived from it. Text length, language, parts-of-speech, stop words, etc., all aspects that represent a document carry the unique id of the Document Event.

Term Event

A document consists of at least one term; thus at least one *Term Event* is always derived from a Document Event. This is indicated by the cardinality between Document and Term Event as depicted in the model. In addition, there is only a unidirectional association and not a composition or aggregation relation, because the Document Event is neither made up of Term Events (aggregation) nor does it depend on their existence (composition). The purpose and semantic meaning of a Term Event is to preserve and represent the raw, inflected form of single tokens within a document along with its position in the text. Thereby the original document information is still available within the event processing network without having to transmit the complete document and the original document can be re-constructed[18]. In terms of the defined tasks, Term Events facilitate the task of document structure representation and they can be used for feature selection, because the raw, inflected term form can be incorporated into metrics that e.g. considers the

[18]Usually the Document Event and the assigned document would be retrieved from the event store.

variety of a dynamic corpus[19] based on the number of different terms.

Lemma Event

A term is the unprocessed occurrence of a word. But words occur in different forms, e.g. nouns in plural or singular, and verbs in the active or passive form. In order to reduce the dimensionality of the event space, terms are processed and reduced by applying stemming algorithms to a basic form called *lemma*. In (Crystal, 2008, p. 273) a *lemma* is defined as an

> ...item which occurs at the beginning of a dictionary entry; more generally referred to as a headword. It is essentially an abstract representation, subsuming all the formal lexical variations which may apply: the verb walk, for example, subsumes walking, walks and walked.

To represent this concept, a *Lemma Event* is introduced which is a subclass of a *Term Event*. It only adds the property *lemma* to the event class and is used to carry the stemmed term information. Lemma events are heavily used within an event processing system for text processing and information filtering, because e.g. term counts, queries or term matching are normally based on lemmata. But a lemma is not capable of covering further syntactic or semantic meaning. Therefore, two Lemma Event subclasses are introduced: *syntactic lemma events* and *semantic lemma events*.

Syntactic Lemma Events

The idea is based on the separation of syntax, semantics and pragmatics as proposed by (Morris, 1946). This means that a Syntactic Lemma Event "disregards meaning in favor of the study of 'purely formal phenomena"' (Allwood, 1981, p. 178). These event classes represent concepts that only describe the formal aspects and the linguistic structure (Chomsky, 2002, p. 1) of a document associated with a Document Event. In this version of the reference model, this subclass entails *Stop Word Events*, *N-Gram Events*, *Parts-of-Speech Events* and *Punctuation Events*. Stop Word Events represent a stop word[20]. An N-Gram Event can be derived if one or more token exists within a document. In fact, a Lemma Event is a *Unigram Event*, but to avoid this rather specific term the more general and understandable term Lemma Event was introduced, because most term weighting schemes and filtering approaches use single lemmata and thus the more commonly used class is used as the superclass; NGram Events are modelled as a special case of Lemma Events. Parts-of-Speech Events define the grammatical class of a Lemma Event, i.e. they carry the basic word category information (noun, verb, prefix, adverb,

[19] The term dynamic corpus is used to describe a steadily changing corpus whose scope is defined using some kind of event window (cf. section 2.3.3)

[20] Stop words are words that are ignored for e.g. score calculation or term matching based on rules like "ignore ten most common words".

etc.) (Crystal, 2008, p. 352f). Nevertheless, the attribution of this event class to syntactic events is debatable, because by enriching a lemma with its parts-of-speech it in fact provides deeper semantic information. The reason to situate this event class in the syntactic hierarchy is that parts-of-speech were categorised as syntax related in one the seminal books on *natural language processing* (Jurafsky & Martin, 2000, p. 308). A Punctuation Event represents special signs within a document like !, ?,. or $ for instance. One could argue that such signs are not a lemmata, but it depends on the purpose of the analysis[21]

All these event classes can be used to describe the linguistic structure and by using events from the *Meta Event Domain* that are described in the next section, interesting analyses of text streams can be conducted in real-time[22]. Depending on the purpose and the used text streams more event classes focused on the structure of the document could be derived like *Caption Event* or *Image Caption Events*. However, due to the defined scope these are not shown here. Figure 4.8 displays the classes and relationships just described.

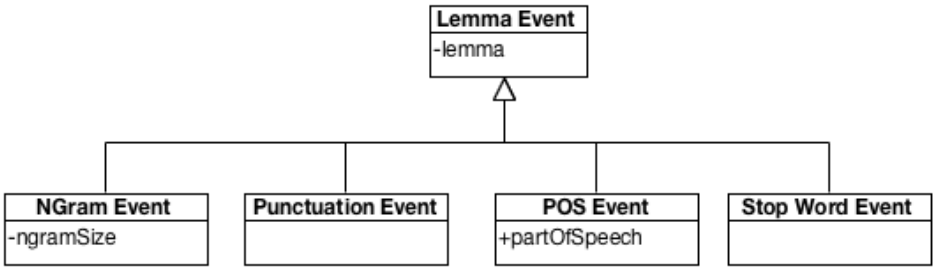

Figure 4.8: Syntactic Lemma Events class hierarchy.

Semantic Lemma Events

Semantic Lemma Events in contrast are centred around the definition of semantics as given in (Allwood, 1981, p. 178) where semantics "...is supposed to be concerned with those aspects of meaning which are situation independent while pragmatics deals with those aspects of meaning which are dependent on situational factors"[23]. There are four defined main semantic lemma event subclasses. The model is extendible, thus it is possible to add new subclasses if required. But for the purpose of this version of the reference model the scope is on information filtering, therefore only the necessary classes are

[21] (Jurafsky & Martin, 2000, p. 192) names for instance using ? as indicators for marking questions, using punctuation for spell checking purposes or incorporating punctuations in n-grams to enrich the context of the n-gram.

[22] Chapter 7 presents such an analysis

[23] *Pragmatics* – to conclude the proposed trichotomy proposed by (Morris, 1946) – represents based on this definition higher-level concepts that are more suited to be situated in the *Pattern Event Domain.*

included in the model.

Sentiment Events represent an emotional expression within a document. There are three subclasses of Sentiment Events: Neutral, Positive and Negative Sentiment Event. As these are subclasses of a Lemma Event they are derived using dictionary based sentiment analysis. This means that lists of words for each of the three sentiment types are used to lookup if a given lemma pertains to a specific group. The problem with dictionary based sentiment analysis approaches is that they cannot cover topics like irony, negations or complex interplay of utterances. But for basic analysis of the sentiments within a data stream this is a reasonable starting point, because the determination process is fast and calculation based on counting Sentiment Events is also efficient. Higher-level Sentiment Events that are derived using more sophisticated methods can be situated in the complex event domain.

The next class of semantic lemma events are *Function Indicating Events*. They represent a subclass of Lemma Events to which a special functionality can be attributed based on some special mark-up This means a simple lemma is enriched with more semantic meaning. The Function Indicating Event class is an abstract class. This is similar to the Base Event and means that this class is not instantiated, but serves only as a superclass to group semantic concepts. *Hashtag Events* are based on the idea to mark a topic with prefixing a lemma with a # sign. A more general case is a *Tag Event* that indicates if a lemma was marked by a user as a keyword for a given document. These two classes are modelled on the same level, due to the popularity of Twitter and its hashtag concept.

Link Events are the superclass for events that indicate any type of "connection", i.e. it represents the linkage between arbitrary objects or persons. The term *link* is not used in the sense of an internet link. This is more precisely covered by the *URL Event* that represents this concept. The *Relationship Event* models a binary relationship between a *target* and a *sender*. This can be used to model *Contact Events*. For instance, in Twitter an @ sign can be used to address a specific other user. By extracting this information from the text using some regular expression and by taking the sender information from the Tweet, this event can be filled and used to model more sophisticated interactions (cf. first example table 4.1).
Named Entity Events on the one hand represent the result of the *NLP* (Natural Language Processing) related task of identifying persons, objects, organisations or quantities from a raw text. On the other hand they can also be directly instantiated from meta data information, for instance from the document meta data information or the event meta data information. Figure 4.9 shows the Semantic Lemma Events class hierarchy in detail.

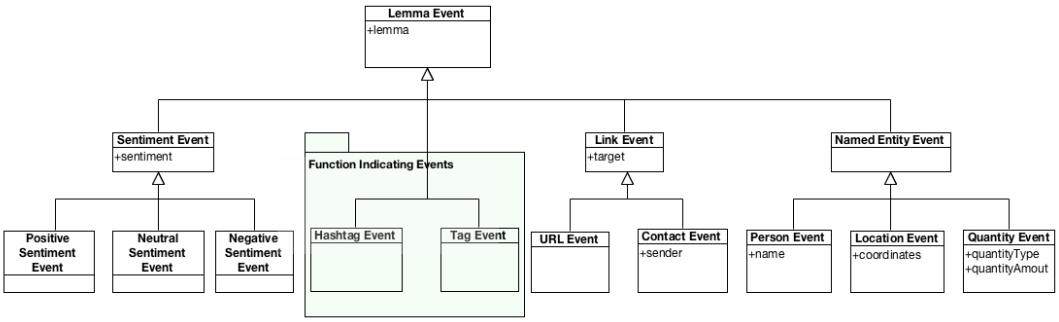

Figure 4.9: Semantic Lemma Events class hierarchy.

All this syntactic and semantic information could also be kept as an attribute of a Lemma Event, but as argued above, it is preferable to map each semantic base event onto distinct and semantically specific event classes, because it facilitates better understanding and easier implementation[24]. Statistics calculated for each event class can be related by using the lemma property as a primary key. For instance, counts of a Lemma Event "Obama" can be joined with the counts of a Hashtag Event "Obama". Compared with other lemmata and hashtags one can make statements about the importance of a hashtag.

Document Type Event

Next, the *Document Type Event* class is introduced. It is used to represent a specific semantic meaning that is applicable to a whole document. Such semantic meanings can be general categories like *Comment Event* or event classes derived from popular social media applications. For instance documents from Twitter can have the type *Tweet Event*, *Retweet Event* or *Favoured Event*, on Pinterest items are pinned and thus a document can be of type *Pinned Event* or *liked* posts on Facebook can have the type *Liked Event*. As argued previously, it is preferable to map distinct semantics to distinct event classes. Nevertheless, it could be also favourable to model this aspects as an attribute of the Document Event class. The reason for this could be induced by the event processing engine that is used and thus must be decided by the architect who designs the concrete implementation.
Also, one might argue that this could be represented using the *source* property from the Event Meta Data, but the source itself is not sufficient to provide clear semantics (e.g. Tweet and Retweet Event). This is because in applications like Facebook, Twitter or Pinterest a whole document (regardless of length) can belong to various semantic classes.

The *Query Event* is a direct subclass of Document Event. It can be used to represent search or filtering intentions of users. By implementing this event class, it is possible to conduct analyses just like for the other document event

[24]When writing event processing language statements such a nomenclature is easier to read

classes, e.g. count and analyse the most common lemmata in query strings.

Search Profile

The last concept that is introduced is the *Search Profile*. It represents the information need expressed by a user and therefore contains the query terms, an identifier and an arbitrary complex profile of the user or the search intention. This means the *profile* property can represent the complete social network of the user who issued the information need or it can contain all positive and negative relevance feedback documents and query expansions terms. Hence this entity is important, because the information stored in the search profile influences information filtering related tasks like scoring, filtering or relevance feedback. The association between the Document Event and the search profile is an aggregation that is navigable in both directions, i.e. a search profile can contain zero or more documents that are relevant (or also not-relevant, depending on the purpose of the document) and a Document Event knows when it is assigned to a search profile. The *zero to many* multiplicity on the side of the Document Event also signifies that it can be relevant to several search profiles.

4.2.4.3 Meta event domain

The third dimension that is introduced is the *Meta Event Domain*. The goal of the events situated in this dimension is to provide a model that allows representing information that is derived from or is about other event classes. Figure 4.10 provides an overview of the concepts that are now introduced.

Various Meta Data Events

The main class is obviously the *Meta Data Event*. It is an abstract class that is used to group derived subclasses semantically. An important aspect of this dimension is to differentiate between the Meta Data Event and the Event Meta Data that were introduced in the General Event Domain as properties of Base Event. The Meta Data Events represent the information from the Event Meta Data as a self-contained event class. The advantage is that thereby Event Meta Data becomes independent of the initial event they were attributed to, and can then be used to execute analysis on specific meta data aspects. For instance, a regular Document Event has an attributed location that matches is point of origin. Thereby it is already possible to filter Document Events solely based on the value of the location property of its eventMetaData. But if you instantiate this location as an self-contained *Location Event* this information becomes independent of the Document Event and can then be used as an autonomous concept within stream calculations or pattern matching. Of course, one could use the property instead of the Meta Data Event. But this would clutter pattern statements, and more importantly, it would violate the goal of clear semantics. Furthermore, Event Meta Data

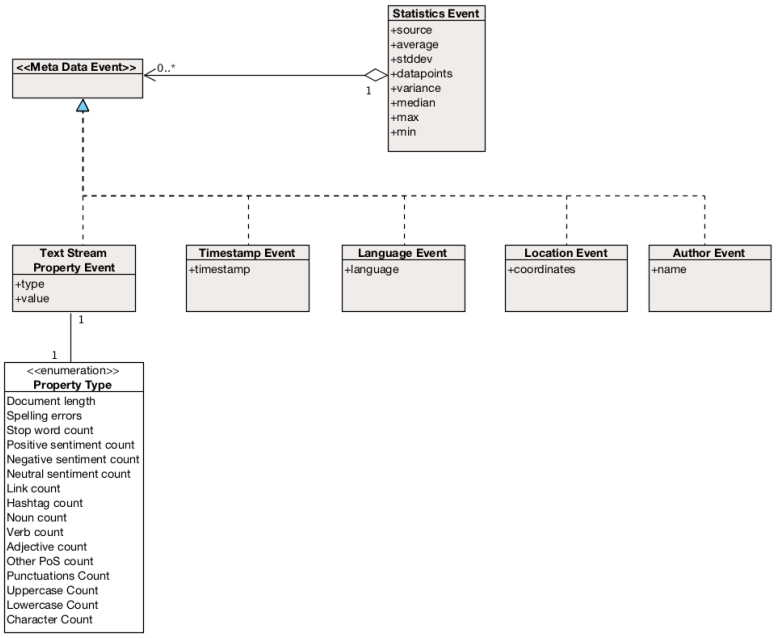

Figure 4.10: Meta Event class hierarchy.

and Meta Events are separated for reasons of simplicity, because the reconstruction of meta data context for an event solely based on the derived Meta Data Events would require costly operations in event stores[25] to retrieve all required events. Thus, also *Author Events*, *Language Events* and *Timestamp Events* are introduced.

Text Stream Property Event

The next event class introduced is the *Text Stream Property Event*. It is used to model any structural aspect of the text contained within a Document Event. This event class has two properties, a *type* and an according *value*. The type refers to an enumeration of text stream properties that can be recorded for a document. The reason to model the various structural aspects as an enumeration property instead of independent, self-contained event classes is that these structural features are often used jointly, e.g. in order to calculate features for learning algorithms all the processed Text Stream Property Events can serve as input. A distinct event subclass for every Syntactic or Semantic Lemma Event would only also clutter the reference model. Nevertheless, where appropriate, i.e. depending on the chosen event processing engine, different subclasses can be easier to implement. Again, this decision must be made by the solution architect. Nevertheless it can be stated that Location Events, Language and Timestamp Events represent different semantic aspects, while Text Stream Property Events are highly related in their purpose

[25] Event stores are used to save raw event data and are explained in more detail in Chapter 5.

– describing syntactic and semantic aspects of a document. In general Text Stream Property Events are used to convey quantitative information about a document in form of counts of specific structural features, therefore the same calculation procedures apply and thus they should preferably be modelled using the attribute approach. Figure 4.11 shows a number of possible features. This can be extended as needed, but in this dissertation only the ones shown were used.

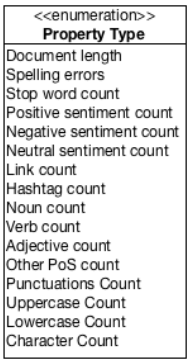

Figure 4.11: Possible property values for the Text Stream Property Event.

Statistics Event

Finally, the *Statistics Event* is introduced. This is in fact a subclass of the *Analytical Event* class from the *Complex Event Domain* that is introduced in the next section. But the Statistics Event is situated here, because this event class is used to store the summary statistics of the Meta Data Events. The goal is to represent the most relevant statistical modes. This is important, because such statistics can be used throughout the information filtering process. For instance, the *standard deviation* and the *mean* can be used to calculate the *z-score* of a Text Stream Property Event value. The *minimum* and *maximum* values can be used to normalize a count value to a scale of $[0, \ldots, 1]$. *Variance* and *median* are also relevant data to characterize data. On the one hand all these values can be calculated incrementally and fast within data streams. And on the other hand they provide rich semantics for further analysis. Thus this event type is essential for many of the tasks that were introduced in the scope and problem definition section.

4.2.4.4 Complex event domain

The fourth dimension deals with the event type hierarchy of *Complex Events*. The event types described in this section represent a set of event types that can be derived using information from the other three dimensions. As there a many ways to combine, transform or aggregate event types from the other

domains, this domain does not define an absolute, final set of events, but rather demonstrates what kind of Complex Events can be derived. It is illustrated by a few plausible examples.

According to the definition given in (D. Luckham & Schulte, 2011, p. 17) a Complex Event is "[a]n event that summarizes [sic], represents, or denotes a set of other events". Two main subclasses are presented: *Analytical Event* and *Heuristic Pattern Events*. Figure 4.12 gives an overview of the now discussed event classes.

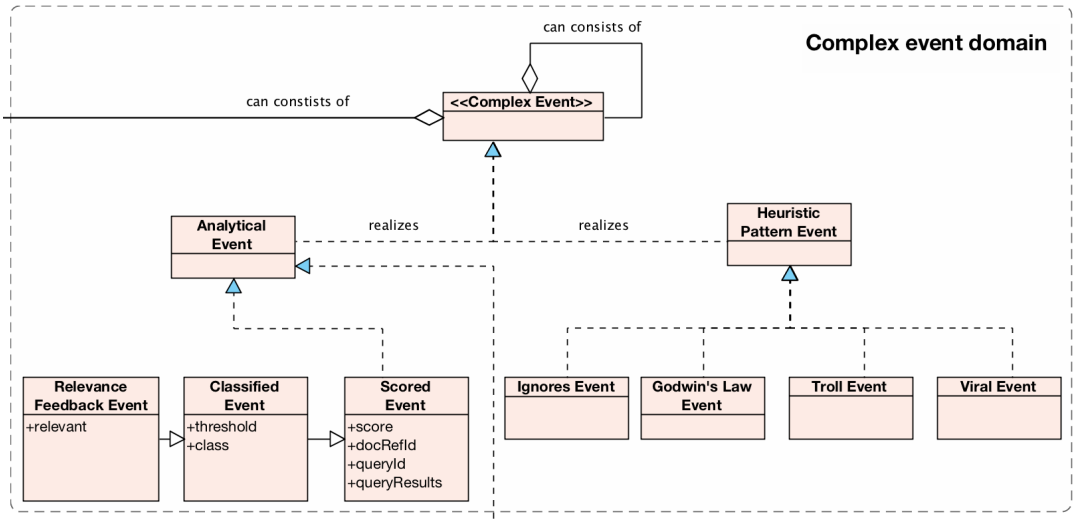

Figure 4.12: Complex Event Domain class hierarchy.

Analytical Events

The Analytical Event subclass represents events that are the result of mathematical calculations, algorithms or some automated pattern detection based on supervised or unsupervised machine learning. The simplest form of an Analytical Event is to calculate any statistical values from a data stream and represent it as a Statistics Event. This class was already introduced in the Meta Event Domain. In the context of this dissertation three relevant subclasses are required and are thus in scope of this reference model. *Scored Events* represent the result of a document scoring process, i.e. based on a term weighting scheme like *tf.idf* a score for a document is calculated and emitted. Such Scored Events can also be used as input for a classifier. They also are the superclass of Classified Events. *Classified Events* represent the results of a classification algorithm. The classification algorithm learns a model using Text Stream Property Events. The model is used for classification purposes and the results are emitted as Classified Events. Finally, a *Relevance Feedback Event* is a Classified that contains information about how the user perceives the classification of the system. The *relevant* property indicates if the user considered the information as really useful and not useful.

Heuristic Pattern Events

In contrast to Analytical Events, *Heuristic Pattern Events* represent events that are generated by matching event patterns that are based on heuristics and were developed by domain experts. Although machine learning is a major enabler to harvest information from data and text streams, for several reasons it can still be beneficial to encode domain-specific interests in human modelled event patterns. First, such heuristic patterns are comprehensible and the results are reproducible for a human being. Second, machine learning works by e.g. minimizing a cost function or detecting hidden structures based on mathematical principles. But it is not necessarily true that the objectives or results of a machine learning algorithm match the interest of a domain expert.

Digital humanities

For instance, an example from the domain of *digital humanities*[26] should illustrate this idea. A sociologist could be interested in using the Twitter stream to look for indicators of riots that are not located in a certain area, but globally. Therefore he could set up a list of term filters that are indicators for unrest or dissatisfaction. He could also define a set of sentiment filters that are related to negative emotions. Additionally, she would say that at least 15 different people within 10 minutes and within an area of one square kilometre must have used the previously defined terms or sentiments in their Tweets. All this requirements could be summarised and modelled as a heuristic event pattern and deployed on the Twitter stream. The results of these pattern matches are Heuristic Pattern Events, which could then be used to further drill down and analyse what has happened in more detail.

An automated machine learning based approach could also generate such event patterns. But the first problem could be that the machine learning algorithm would maybe detect this pattern at a time when no domain expert is interested. The second problem of an automated approach could be that they generate too many or too few patterns. In the first case a human examiner would probably miss interesting patterns as she cannot evaluate all proposed event patterns. In the second case, potentially interesting patterns could not be detected, because they were not presented to the human examiner. Therefore it is important for a domain expert to be able to explicitly model his interests and intentions using heuristic event patterns and to process and analyse the matched Heuristic Pattern Events.

Of course, it is possible to combine both complex event subclasses, in order to leverage the best of the two approaches. But this is not in scope of this dissertation and could be subject to further research. Nevertheless, a few social

Social media event patterns

[26] Digital humanities is a field of research that combines humanities and computer science as a new way to answer humanities related questions using the possibilities offered by modern data processing technologies.

media event patterns are introduced, in order to illustrate the idea how to combine low level events into higher level ones. The first pattern tries to detect *Ignore Events*. An Ignore Event could represent the intentional ignoring of one user by another. The idea is to monitor several contact trials from a user A to user B within a given time period where no response from user B is detected. Another event class that could be derived using social media event patterns are *Riot Events*. The pattern to detect these events could be based on monitoring a substream A that contains lemmata that represent anger or unrest, a substram B that monitors negative sentiment events, a substream C that relates substreams A and B and the final substream $result$ that takes the results from substream C, combines them with an influence score of authors of events detected in substream C and counts how many of these events occur within a given time period and within a certain area. A *Troll Event* could be spotted if a certain number of insulting lemmata by a single author are monitored within a given time period for a given topic. Finally, a *Godwin's Law Event* could be detected by monitoring a certain hashtag and waiting for the first appearance of any term related to German fascism or Hitler[27].

Table 4.1 summarises a few examples of such event classes along the corresponding event patterns written in a pseudo pattern code[28].

4.2.5 Reference process models

This section describes basic reference processes that are required for an event based text processing and information filtering system. In section 4.2.1 a set of processes has already been proposed as in scope and these are now presented in more detail. The goal is to provide a basic and general blueprint of information filtering related tasks in the context of event processing. This is not an exhaustive list and further processes could be subject to further research.

Aspects such as the mapping to certain types of event processing agents or infrastructure components as well as the definition of cardinality of event processing agents or the expected event volume, i.e. more software architecture related questions, are cover in the next chapter on reference architecture. This section only discusses the fundamental process aspects.

[27] Godwin's Law states that "[a]s an online discussion grows longer, the probability of a comparison involving Nazis or Hitler approaches one" (Godwin, n.d., p. 1).

[28] The patterns should only illustrate the idea behind the complex events and do not represent working patterns. $\wedge$ means *and*, $\neg$ means *not*, $\{3,\}$ means an event has to occur three times or more and the $.function(parameter)$ notation is borrowed from Java syntax

Event class	Pattern
Ignores Event	$substream_x = filter(RelationshipEvent(sender = A, target = B))$ $substream_y = filter(RelationshipEvent(sender = B, target = A))$ $result = [(substream_x\{3,\})\neg \wedge substream_y].within(10minutes)$
Riot Event	$substream_a = filter(LemmaEvent(lemma \subset corpus(UNREST)))$ $substream_b = filter(SentimentEvent(lemma \subset sentiment(Negative)))$ $substream_c = filter(substream_a.documentRefId = substream_b.documentRefId)$ $result = filter(\text{count distinct}(c.sourceEventId = AuthorEvent.sourceEventId) > threshold)$ $\wedge within(xminutes)$ $\wedge within(nsquarekilometres)$
Troll Event	$result = count(LemmaEvents(lemma \subset corpus(INSULT)).groupby(AuthorEvent.name)$ $result = selectwhereA > threshold$
Godwin's Law Event	$result = filter(count(DocumentEvent.groupBy(Hashtag)))$ $\neg \wedge DocumentEvent.text.contains([\'Nazi\|Hitler\'])) > threshold$ $\rightarrow DocumentEvent.text.contains([\'Nazi\|Hitler\']))$

Table 4.1: Examples of complex events based on heuristic patterns.

4.2.5.1 Notes on modelling aspects

As mentioned in section 4.2.2, data flow diagrams based on DeMarco's notation are used (Structured Analysis Wiki, n.d.). Figure 4.13 shows the main design artefacts that were employed. The rectangular boxes represent external entities. In terms of modelling towards event processing systems where decoupling and asynchronous processing are essential aspects, such external entities can for instance be raw document sources outside of the event processing network, as well as other event processing agents within the same event processing network. This might seem contradictory, but as each event processing agent acts independently it is valid to model event processing agents that way. Arrows indicate the flow of events between the components. Circles describe one or several process steps that are executed within an event processing agent[29] and event stores represent locations where data are at rest and waits for further processing (Structured Analysis Wiki, n.d., sec. 2.2).

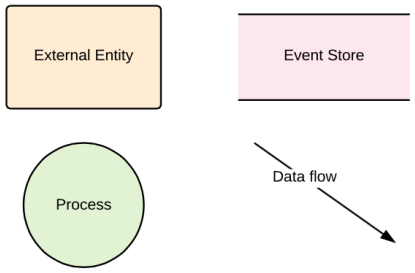

Figure 4.13: Used dataflow diagram artefacts.

4.2.5.2 Document pre-processing

This process describes the essential steps that *always* need to be taken when a new document arrives at an event processing system. This is because with this step text stream data gets converted into the basic event types and can thereby be processed by event-driven means.

The first step is to wrap the document in a Document Event and create the required event meta data. Then, the Document Event is emitted and three further steps happen in parallel execute by specialized EPAs; this is possible due to the asynchronous nature of an event processing system. One step is to save the Document Event to an external event store. As mentioned,

[29] As described in section 2.3, these are the central components of event processing networks in terms of operating on events.

the complete text is no longer required in the event processing network after pre-processing, as it is only needed in few specialized steps like scoring. Thus it can be safely stored until it is required again, in order to avoid network congestion. The other two steps involve the creation of Text Events and Meta Data Events. This means all event classes from the Text Event Domain and Meta Event Domain that are required[30] are derived from the text of the document. Figure 4.14 shows the process.

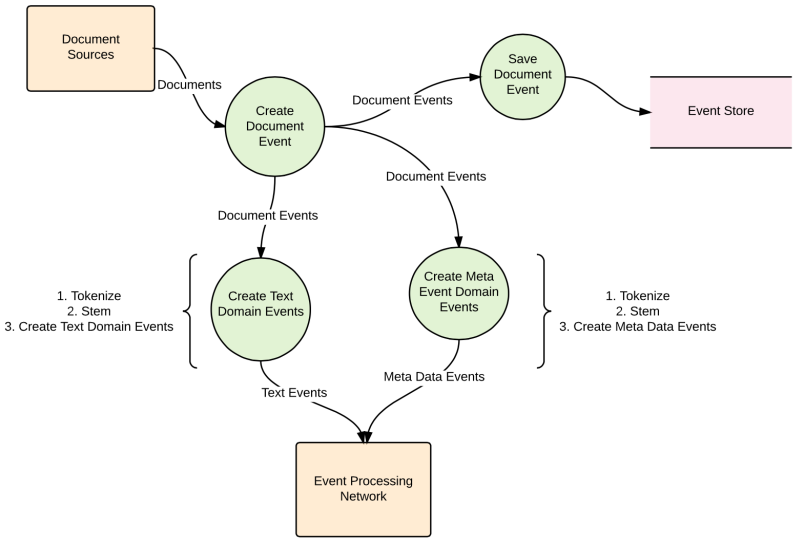

Figure 4.14: Process model: document pre-processing and index maintenance.

4.2.5.3 Stream features modelling

Stream feature modelling addresses the derivation of data that can be used for machine learning or scoring algorithms. As previously mentioned, counts can be calculated well on data streams, and many scoring algorithms and machine learning algorithms rely on counts as the basis for their calculations. Thus, the process is to turn the various event classes into quantities and feed them into the event processing network, where different event processing agents can act upon them as required. One process step takes the counts associated with a Lemma Event and stores them in a temporal index. As all processing is based on sliding windows the temporal index and counts contained within are updated frequently. Another process step is to calculate the summary statistics for the various Meta Data and Text Events within the event processing network. Based on the event reference model

[30]The required events can vary from purpose to purpose. In context of this dissertation the instantiated event classes are listed in the implementation chapter.

each semantic aspect within the event processing network is represented as a distinct event class and thereby each event class forms an independent data stream on which the summary statistics can be calculated without being mixed with aspects from other event streams. The summary statistics mapped onto Statistics Events can then be used by the process for model learning, which as a first step normalizes or standardizes the features from the document. These are subsequently used for modelling learning or scoring. Figure 4.15 summarises the process.

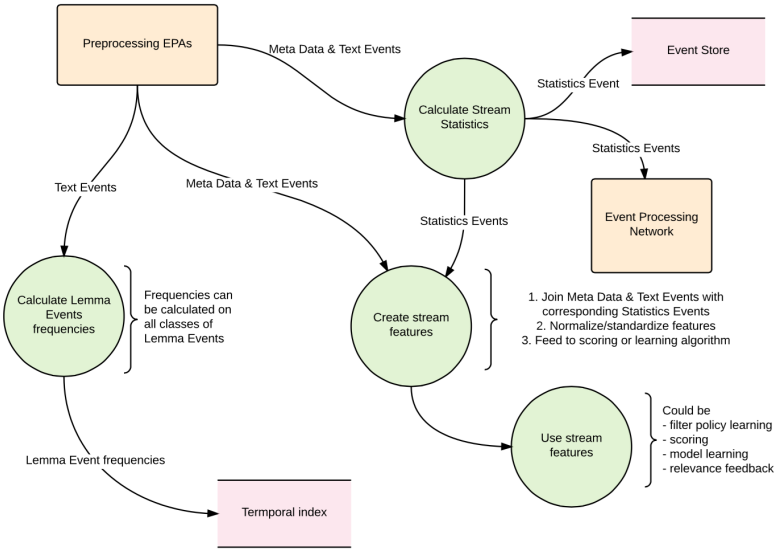

Figure 4.15: Process model: stream features modelling.

4.2.5.4 Scoring, term weighting and classification

Scoring and term weighting are relevant for filtering high volume text streams, because a score, which based on term weights and document properties, is an essential aspect in information filtering and information retrieval, in order to judge the quality of a document. There are several ways to determine the score of a document, but for high-volume text streams, count-based and probability-based approaches are the most natural fit, because counting on data streams works well[31]. Probability-based terms weights require count information about terms, co-occurrences or overall document volumes, in order to calculate their scores. Thus, a temporal index is required that stores this kind of information. When a dynamic sub-corpus approach[32] is used such a temporal index can be kept individually and independently for every

[31] For instance, cf. *Count-Min Sketch* algorithm by (Cormode & Muthukrishnan, 2005).
[32] Dynamic subcorpora are introduced and discussed in section 3.3.1

EPAs due to the continuous removal of out-dated information. Additionally Statistics Events can be incorporated to enrich scoring and weighting algorithms. The scoring algorithms takes the incoming events and emits scored events into the event processing network. Figure 4.16 displays a conceptual description of the process.

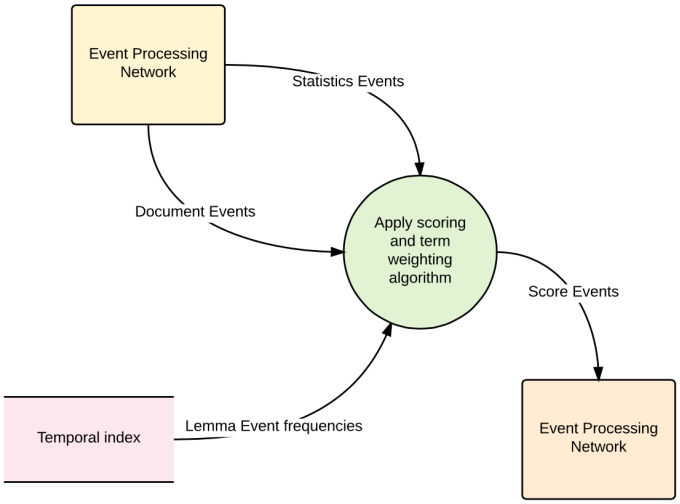

Figure 4.16: Process model: scoring, term weighting and classification.

4.2.5.5 Relevance feedback and query expansion

As described in section 3.3.5, relevance feedback and query expansion are an essential aspect when it comes to readjusting and expanding the initial search profile. Relevance feedback helps to filter better quality documents and query expansion helps to broaden the covered aspects of a search profile.
The process is as follows. The relevance feedback process step takes the Classified Events, adjusts the term weight of the documents pertaining to the Classified Event and hands over these terms to the query expansion steps that updates the search profiles. Figure 4.17 summarises the process.

4.2.5.6 Filter policy generation

Filter policies are a special kind of filters which are comparable to term filters, however they are not based on terms, but rather on other quality aspects of a document. In fact, a term filter is a special kind of filter policy. The goal

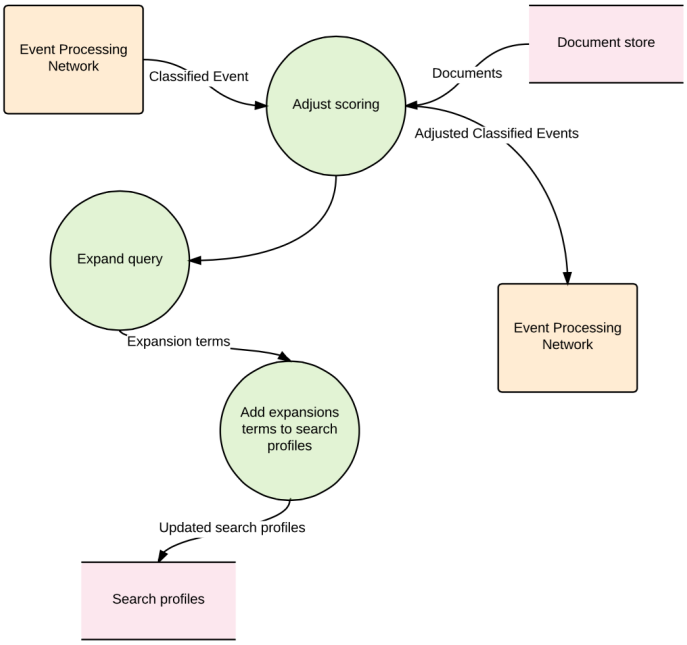

Figure 4.17: Process model: relevance feedback and query expansion.

of a filter policy is to remove as many obviously[33] irrelevant documents as possible, in order to decrease the load on the later processing stages in the event processing network.
In this dissertation three approaches are studied in later chapters, *viz.* classifier based, heuristics based and frequent pattern based. The idea of all of them is to use other features than terms to conduct a high quality preselection of documents. Figure 4.18 depicts the process.

4.2.6 Evaluation

As described in section 4.1.2 there is no single best approach to evaluate a reference model. Among the various discussed approaches, an empirical evaluation approach based on case studies together with a questionnaire seemed to be the best choice in the context of this dissertation due to the means required in terms of time and participants.

Case studies

Therefore, the quality is, first, validated based on the successful implementation of two case studies that are presented in chapter 7 and 8[34]. Without anticipating any of the results, the reference provides sufficient detail to imple-

[33] For instance, if you are only interested in English documents it does not make sense to process any non-English document

[34] In this dissertation a text stream analysis as well as a study of information filtering components were conducted successfully using the proposed model

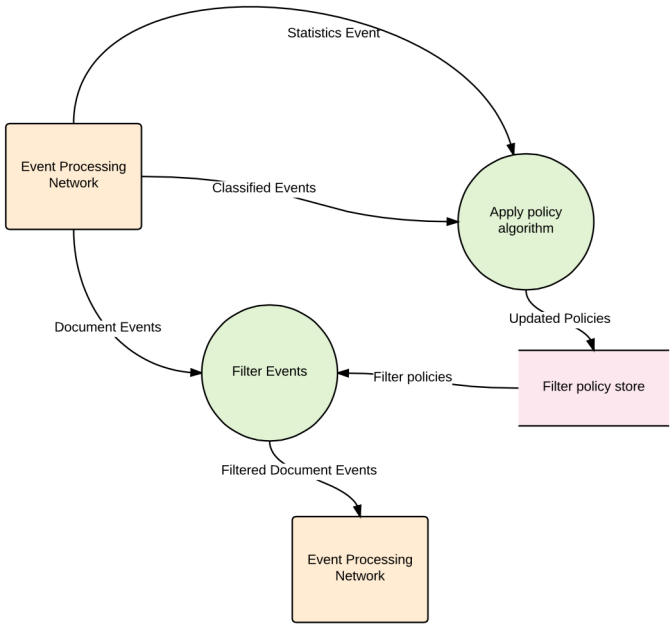

Figure 4.18: Process model: filter policy generation.

ment the required tasks. Furthermore, the acceptance of the reference model at the DEBS 2014 conference[35] can also be considered as confirmation of the quality and the necessity of such a reference model.

Questionnaire

Second, a questionnaire was developed based on the reference model characteristics introduced in section 4.1.1 and using evaluation criteria for reference models presented in (Frank, 2007). The characteristics of reference models included generality, flexibility, completeness, usability, understandability, re-usability and acceptance. These properties were guiding when the questionnaire was designed. This means each aspect should be covered by one of the questions of the questionnaire.

The concrete structure of the questionnaire was derived from findings presented in (Frank, 2007). The details were discussed in section 4.1.2, but in summary the author argued to evaluate a reference model from different views, in order to be able to respect the requirements of different stakeholders. Therefore, 22 questions were derived that tried to address as many of the aforementioned characters as possible. Tables 4.2 and 4.3 show which questions addresses which characteristics of the reference model. These questions were then structured along the four proposed views, economic perspective, deployment perspective, engineer perspective and epistemological perspec-

[35] The paper (Bauer & Wolff, 2014) passed a four reviewer based peer review and belonged to the 16 papers that were accepted out of 174 papers (cf. `https://dl.acm.org/citation.cfm?id=2611286` tab "publication")

ID	Q01	Q02	Q03	Q04	Q05	Q06	Q07	Q08	Q09	Q10	Q11
Generality								x	x	x	
Flexibility				x	x		x				
Completeness				x	x	x					
Usability		x	x	x				x		x	
Understandability	x	x	x	x				x		x	x
Re-usability	x			x					x		
Acceptance	x	x	x	x	x	x			x		x

Table 4.2: Confusion matrix of reference model characteristics and questionnaire questions no. 1 to 11

ID	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
Generality								x	x	x	x
Flexibility			x								
Completeness									x	x	x
Usability			x	x	x	x	x				
Understandability	x	x		x	x	x	x				
Re-usability			x	x	x	x	x			x	
Acceptance	x	x									

Table 4.3: Confusion matrix of reference model characteristics and questionnaire questions no. 12 to 22

tive (Frank, 2007, p. 119ff), in order to provide a precise context for evaluation participants. Table 4.4 shows the questions in detail.

ID	Question
Economic perspective	
Q01	Is the model suited to foster internal (in the range of the defined stakeholders) dissemination of relevant knowledge?
Q02	Do you think the model supports the development of relevant, domain-specific skills of employees?
Q03	Is the model suited to decrease the time that is required to bring to employees up-to-date?
Deployment perspective	

Q04	Is the description comprehensible to be adapted by the defined stakeholders?
Q05	Are enough scenarios (=process models) provided to use the model in real world scenarios?
Q06	Is the scope of the scenarios (=process model) clear?
Q07	Is the model open to be implemented with different / new technology stacks?
Engineering perspective	
Q08	Is it clear to which domains the model could be applied?
Q09	Is the purpose of the model clear?
Q10	Are the model elements (entities and relationships) correctly assigned to the process models?
Q11	Are design decision explained and justified?
Q12	Are compromises and resulting drawbacks of the design discussed?
Q13	Are alternative approaches (e.g. for modelling event classes) discussed?
Q14	Is the level of formalization and extensibility adequate to support the adaptation of the model by solution designers?
Has the model been designed formally correct in terms of	
Q15	the proposed conventions (cf. chapter requirement analysis & conventions setting, e.g. naming conventions)?
Q16	the introduced event classes?
Q17	the use of generalization and specialization?
Q18	the use of modularization and encapsulation?
Epistemological perspective	
Q19	How adequate do you think was the choice of modelling languages (UML and dataflow diagrams)?
Q20	Do you think the model provides enough information to be subject of further research activities?
Q21	How do you rate the level of abstraction of the model?
Q22	How do you rate the level of originality of the model?

Table 4.4: Questionnaire used for the evaluation of the event reference model.

Evaluation

The evaluation process was as follows. The previously described event reference model and the process models were presented to five IT experts with

	Gender	Education	IT Experience
Rater 1	m	Diploma Computer Science	>15
Rater 2	m	Master Information Science	>10
Rater 3	m	PhD Computer Science	>20
Rater 4	m	Master Information Science	>10
Rater 5	m	Master Information Science	>10

Table 4.5: Overview of questionnaire participants.

varying knowledge in terms of event processing, information retrieval and linguistics. All of them had several years of proficient IT experience. After studying the text they were asked to fill in a questionnaire that contained the questions from table 4.4. They were able to rate each question on an uneven *Likert* scale. The scale was from one to five, where *one* meant *very good*, *three neutral/no opinion* and *five* corresponded to *not good at all*. The uneven version was chosen to allow the participants to express a neutral sentiment about the questions. The "neutral" value *three* was used to replace missing values. In total, nine out of 110 answers contained missing and were thus replaced with this value. The reasons was that due to the small number of questionnaires, missing values would have pulled single questions to one or the other extreme of the scale, which would have distorted the evaluation.

Participants

In total five participants filled out the questionnaire. On the one hand this is not a statistically significant number of samples, but on the other hand it provides valuable information on the perception of the model. Table 4.5 shows their qualification and their experience in IT.

Visual questionnaire analysis

The evaluation of the questionnaires was performed using the R package *likert* (Bryer & Speerschneider, 2015), as it offers several convenient methods to analyse such kind of data. Figure 4.19 shows a visual summary of the questionnaire. The id of the question that is visualized is on the left side. The percentage values on the left, in the middle and on the right are the percentiles. The first overall impression shows that the participants rated the reference positively. The majority of the rating is higher than three and the overall tendency is positives. This means the participants rated the reference model as useful.

Discussion of results

Nevertheless, there are also a few questions that do not have the same positive overall rating. Thus a few are discussed in more detail. Question Q03 addressed if a new user can be brought up-to-date efficiently by using the reference model. One of the participants considered the model as bad here,

i.e. more domain knowledge should be introduced. But as with any domain-specific topic, some upfront effort to get to know the domain is always required as well as the basic domain knowledge has been introduced in the previous chapters.
Question Q05 asked, if enough real world scenarios were provided, and a few participants wanted more use cases. Even though the author thinks that the reference processes provide enough illustration this point could be addressed in the next iteration.
Question Q13 was concerned with alternative approaches to designing the reference model. A few of the participants missed that only the UML based class diagram and the data flow approach was discussed. These comments can be addressed as follows. First, section 4.1.4 introduced and discussed other reference models in event processing. Although, they were not compared and there was no evaluation of which approach is better – because this is out of scope of this dissertation –, but they were introduced, and thus consumers of this reference model know where to look for alternatives. Second, many of the approaches used UML as the basis for their modelling, thus the focus on class diagrams for designing the reference model is reasonable. Third, there is no standardized or industry-wide accepted way how to design an event reference model. Although, there are prominent experts in the domain of event processing like (D. C. Luckham & Vera, 1995), (Paschke et al., 2011), (Etzion & Niblett, 2011) or (Springer et al., 2009) who proposed ways of designing reference models, there does not exists a standard that was officially released for instance by OASIS or *Institute of Electrical and Electronics Engineers* (IEEE). Therefore, only standard UML class diagrams and widely accepted data flow diagram – even though the notation is subject to individual taste – were used.

Questionnaire comments

There were also so a few written comments from the participants. In general, they argued that more practical examples and a collection of use cases would be helpful to answers questions 1, 5, 6 and 8. In fact, this was addressed by the introduction of the reference processes in section 4.2.5, the discussion of the function view of the reference architecture in section 5.2.2.1 and the architectural patterns in Chapter 5.2.2.3. Also, the use of DeMarco's data flow notation was considered as uncommon. The author partly agrees with this. There are multiple, and often unstandardised notations for data flow diagrams. The choice of DeMarco's notation was based on two reasons. First, DeMarco was the first to introduced data flow diagrams and, second, the visual representation was the most appealing to the author. Thus, this point can be discussed. But the use of data flow diagrams disregarding the

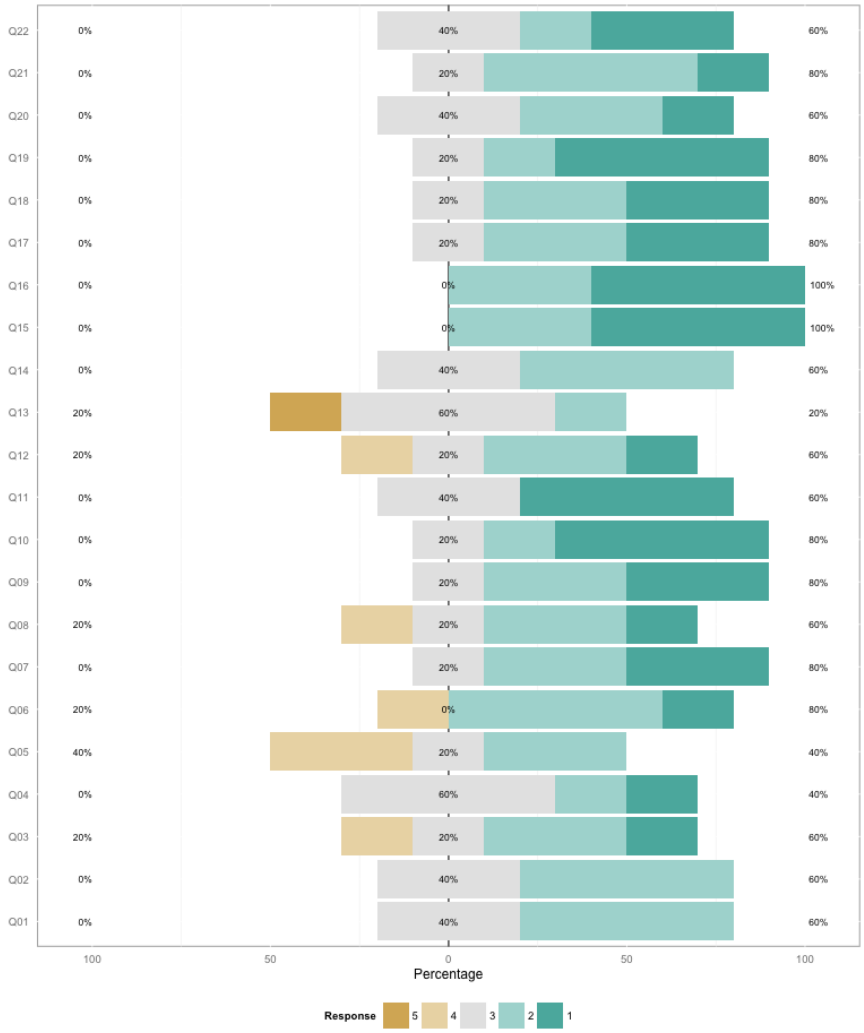

Figure 4.19: Visualization of the Likert-scale based evaluation of the event reference model.

notation seems rather sensible to the author, because the underlying processing mode of event processing components is to communicate via events over some kind of channel. And data flow diagrams are a natural fit for this requirement.

Finally, it must be noted that the design of a reference model is usually an iterative and ongoing process, i.e. each version gets reviewed and reworked. For the purpose of this dissertation and the presented evaluation, the version that evolved over the course of this dissertation and that was presented to the domain experts was described in this chapter, in order to show the design process, the design decisions and the feedback of the reviewers in the same state. The comments and proposals for improvement are listed in the next section, and they were used to derive ideas for the next iteration on the reference model. Thus, the model represents a snapshot before the next re-work phase.

4.3 Summary

This chapter has introduced a reference model for event-driven text stream processing and information filtering. As stated in the introductory chapter, a reference model for these areas is missing, but is required to be able to process text stream data in an event-driven manner. Before the model was introduced, basic principles of reference modelling were discussed and the scope of the reference as well as a design approach for the reference model were defined. Also standards in the domain of event processing modelling were discussed and it was shown that the applied methods for designing the reference model adhere to research standards in this domain. After the theoretical framework had been set, the actual reference model was developed using standard modelling notations. The introduced model unites concepts from event processing, linguistics, information retrieval and information filtering and provides a concise and structure approach, how turn text streams into events. Thereby, the model helps to make text streams processable and manageable using event-driven technologies. Additionally, a set of essential reference processes for the domain of text stream processing and information filtering was described in an event-driven way, i.e. standard information retrieval and filtering tasks like scoring, feature modelling or document preprocessing were described using the tasks and event types that are required to execute those task with event-driven technologies. Finally, the proposed event reference model was evaluated. Therefore, a questionnaire with 22, Likert-scale based questions was developed based on the latest research in reference model evaluation. The questionnaire was answered by several IT

experts and the results were evaluated as well as discussed. The results from the questionnaire showed that the proposed reference model can be considered as a valuable effort to facilitate the application of event-driven technologies to text streams. In the next chapter a reference architecture is devised that structures required components for text stream analysis and information filtering from an architectural point of view. Together with the reference model, these two artefacts allow to design event-driven text stream and information filtering systems.

Chapter 5

A reference architecture for event based text processing and information filtering

In this chapter a reference architecture for event based text processing and information filtering is proposed. The general goal of a reference architecture is to provide guidance and solution templates for instantiating real software architectures within a domain, using the concepts from a related reference model. The last chapter introduced such a reference model including basic concepts and their relationships along a set of essential reference processes for the defined scope of this dissertation. Now, architectural templates, patterns or in general "blueprints" are proposed, in order to facilitate the implementation of real software architectures that enable text processing in an event based manner.

The structure of the chapter is as follows. First, the scope and purpose of a reference architecture are defined. Then, the benefits of reference architectures are introduced, followed by the introduction of a design process for reference architectures. After a brief overview of different reference architecture definitions, which provide a few examples of how to design them, the actual reference architecture is introduced. Using the design process the architecture is developed. The results of the process are a functional view, a process view, a set of basic patterns for event-driven systems, and brief discussion about relevant quality attributes.

5.1 Reference architectures in general

In Chapter 4.1.1 the objectives of a reference model were defined as the description of relevant entities, their relationships and the flow of data between them independent of a specific technology or standard. Now, the goals and properties of a reference architecture in the scope of this dissertation are defined by discussing various definitions of reference architecture from research and industry. Thereby, the difference between both artefacts becomes clearer.

5.1.1 Definitions of reference architectures

Before definitions of reference architectures are discussed, the term *architecture* needs to be defined, as a reference architecture is a special kind of software architecture and, thus, the definition of the generic concept is helpful. According to *ANSI/IEEE Std 1471-2000* (cited after (The Open Group, 2011, Chapter 2.1)) an architecture is the

> fundamental organisation[sic] of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.

The Open Group Architecture Framework (TOGAF) uses two interpretation of this definition (The Open Group, 2011, Chapter 2.1).

> 1. A formal description of a system, or a detailed plan of the system at component level to guide its implementation
> 2. The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time

From these two definitions, three essential aspects can be synthesized that are required by any architecture: a component description, a description of their relationship and guidelines that help during the design process.

For there reference architecture proposed here, the first definition is used, because the second requires modelling and describing the internal structure of each component in detail. But the goal of the reference architecture is only to provide guidance and patterns in an abstract and generic way (cf. next section).

A *reference architecture* – to be more precise it is a *software reference architecture* –represents a special kind of architecture. In its basic sense a reference architecture is an architecture to which a stakeholder *refers* to find hints, best practices or advice when it is about to design a concrete software architecture. There are several definitions in the literature that help to shape the term more clearly.

(Bass et al., 2003, p. 48) provided a concise definition of a reference architecture that also addressed the relationship between reference model and reference architecture at the same time. They argued that

> [a] reference architecture is a reference model mapped onto software elements (that cooperatively implement the logicality defined in the reference model) and the data flows between them. Whereas a reference model divides the functionality, a reference architecture is the mapping of that functionality onto a system decomposition. The mapping may be, but by no means necessarily is, one to one. A software element may implement part of a function or several functions.

This definition highlighted the close relationship between reference model and reference architecture by saying that a reference architecture was in fact a reference model, but concerned with fulfilment of the requirements of software architectures and system components. This means a reference architecture describes the entities, their relationships and the dataflow between them from a software architecture point of view.

DoD definition

The *Department of Defense of the United States of America* (DoD) defined reference architecture as " ...an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions" (Office of the DoD CIO, 2010, p. 3). This definition adds two relevant aspects. First, it defined a reference architecture as an authoritative source of information, i.e. within its scope, which is reasonably defined by the reference model, the reference architecture should be considered as the main source of information in terms of architectural design and decisions. Second, they emphasizes that a reference architecture guides *and* constrains the instantiation of concrete software architectures. This means it does not only provide how to design a concrete architecture, but also how *not* to design it.

OASIS definition

In (OASIS - Organization for the Advancement of Structured Information Standards, 2013, p. 9) the purpose of a reference architecture was defined to model "...the abstract architectural elements in the domain of interest in-

dependent of the technologies, protocols, and products that are used to implement a specific solution for the domain". From this definition two more aspects can be derived that are important to reference architectures. First, they stressed the aspect of *abstraction*. This means, a reference architecture does not prescribe how an architectural component is implemented, but similar to the previous definition of the DoD it should only provide guidance and constraints. Second, they underlined that a reference architecture must be technology agnostic, i.e. the reference architecture should defined in such a way that different technology stacks can be used to implement it. This is important, because in different situations different technology stacks can or must be used. For instance, if a company wants to adapt a reference architecture, but the technology stack is already given due to licence agreements or organisation wide blueprints, this should not affect the application of the reference architecture.

More properties

(Buchgeher & Weinreich, 2013, p. 332f) presented a meta-study of various other definitions of reference architectures, which are not described here, and synthesized three more characteristics. First, reference architectures are intended to foster re-use of architecture design decisions. Second, they encode these architectural design decisions. This means they provide patterns and blueprints for the design of components that are required to implement concept from the domain-specific reference model. This aspect corresponds to the concept of *architectural patterns* as presented in (Bass et al., 2003, p. 48) that are described as sets of usage relationships and constraints together with quality attributes for an architecture component. Third, they identify that reference architecture takes over different roles during the software architecture design process. Besides the authoritative character as already mentioned in (Office of the DoD CIO, 2010), they also see reference architecture as sources of information about architectural knowledge and as regulative in terms of "restricting the design space of systems in development" (p. 333). A reference architecture can also be used to control the quality of a concrete software architecture during the development phase (p. 332).

(Bass et al., 2003, p. 48) also presented a visualization that concisely illustrates the relationship of reference model, reference architecture, architectural patterns and concrete architecture (cf. figure).

5.1.2 Goals, properties and benefits of reference architectures

Using the definitions from the previous section a reference architecture for the context of this dissertation is defined as follows:

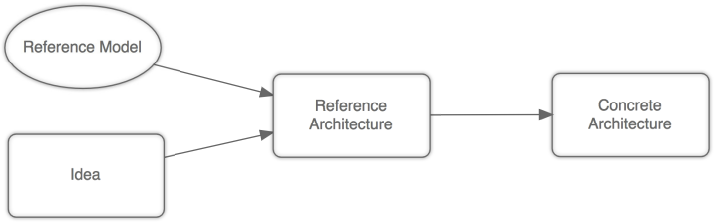

Figure 5.1: Relationship of reference model, reference architecture, architectural patterns and concrete architecture according to (Bass et al., 2003, p. 48)

> A reference architecture is a mapping of concepts from a reference model to software architecture components. It guides, instructs and constrains the design of concrete software architectures. A reference architecture must be abstract, i.e. it does not require specific implementations. It must be generic and technology agnostic, in order to be applicable in various situations without limiting the choice of technologies, and it provide guidelines for designing concrete software architecture.

Properties

This definition introduces three properties that need to be discussed in more detail. First, *abstract* means that a reference architecture should describe its components on a coarse-grained level, i.e. the architecture should only describe the basic aspects without going into too much details. This would limit the applicability of the architecture.
Second, a reference architecture should be generic. This means the architecture should applicable to different situations that are concerned with event based information filtering or text processing without further adaptation or modification. The third aspect, that supports the applicability of a reference architecture is that it should be technology agnostic. The prescription of a certain technology stack would also limit the flexibility of the reference architecture.
Third, a reference architecture should contain guidelines. (Buchgeher & Weinreich, 2013, p. 332) addressed this by stating that a reference architecture should be instructive, informative and regulative. Instructive means that it should provide guidance during the design of a concrete architecture. Informative means that knowledge can be shared and communication is fostered, and regulative means that it guide the design process by "restricting the design space of systems in development" (p. 333).

Based on the previous definition as well as the ones presented in section 5.1.1 the following goals of a reference architecture can be derived:

- Provide knowledge about a domain architecture to stakeholders that have not yet worked within the given domain.
- Foster re-use of architectural decision and patterns.
- Provide guidance and quality control during the process of designing concrete software architectures.
- Reduce complexity during the system design process by providing patterns.
- Facilitate communication and discussion about the subject matter in terms of software architecture.

Benefits and drawbacks

Finally, the design of a reference architecture is not just an end in itself, but is also intended to provide real benefit. This aspect is also important in terms of the Design Science approach that is used in this dissertation, where solutions to problems – in this case the design of high volume text filtering system using event processing means – are required.
(Martínez-Fernández, Ayala, Franch, & Marques, 2013) conducted a study where they investigated the use of reference architectures within nine real world software projects. In addition to the benefits, they also identified several drawbacks of reference architectures. These are also important to mention, because only a covering positive *and* negative aspects of a subject matter allows studying it in its completeness. Based on their study (Martínez-Fernández, Ayala, Franch, & Marques, 2013, p. 308f) defined e.g. the following benefits:

- Reduced development costs due to re-use and speed up process due to the available best practice
- Reduced maintenance costs, because of proven architecture decisions that contain less errors. Also improves a reference architecture the quality of the maintenance documentation
- Facilitated development and increased productivity of software engineers
- Incorporation of latest technologies due to the technology agnostic design of reference architectures
- Risk minimization, because a proven a blueprint for new concrete software architectures already exists

But reference architectures do also have drawbacks. According to (Martínez-Fernández, Ayala, Franch, & Marques, 2013, p. 309)) a few of them are:

- Initial overhead to create the model

- Steep learning curve
- Depending on the volume and scope of a reference architecture the complexity of the architecture prevents its application.

Although a few drawbacks can be identified, due to its benefits a reference architecture is a pivotal component during the design of concrete software architectures. Foremost, in terms of this dissertation, the introduction of a reference architecture is reasonable, because event based text processing systems have not yet been studied in detail and therefore a reference architecture can provide a good starting point to deepen this area of research.

5.1.3 Scope definitions and examples of reference architectures

Scope

Up to here, goals, characteristics and benefits of reference architecture have been defined. Still missing is the definition of the scope and the concrete artefacts of a reference architecture. As there is no a universally valid definition for the scope of a reference architecture, the contents of the reference architecture in this dissertation are derived based on the scope of already existing reference architectures.

DoD scope

(Office of the DoD CIO, 2010) defined that their reference architecture should provide strategic purpose, principles, technical position, patterns and vocabulary. Strategic purpose was supposed to reflect the *mission statement* of the reference architecture, i.e. the goals and objectives of a reference architecture. This has been addressed in the previous paragraph. Principles are supposed to represent "foundational statements of rules, culture, and values that drive technical positions and patterns" (p. 5). This aspect is not covered by this reference architecture due to the lacking organisational foundation. Technical positions reflect predefined technical settings within an organisation. But such organisational constraints are not relevant, because they do not exist in the context of this dissertation and, thus, technical positions are also neglected. Patterns are defined as "[g]eneralized architecture representations (viewpoints, graphical/textual models, diagrams, etc.) that show relationships between elements and artefacts specified by the technical positions" (p. 6). This is in fact a broad definition of the pattern concept[1], but comprises the scope of the proposed reference architecture that is discussed further in the next sections. Finally, a vocabulary should be provided that contains terms and definitions. This aspect has been addressed by the reference model

[1] This dissertation uses the term pattern in the more concise sense of a solution template to a specific problem.

in the previous chapter. In summary this reference architecture contains for the domain of event-driven text stream processing system[2]:

1. A functional view that describes architectural units and structures them along a set of event processing layers.
2. A process view that depicts the interaction of architecture components for information filtering.
3. A set of architectural patterns that are helpful for designing concrete systems.

(OASIS - Organization for the Advancement of Structured Information Standards, 2013) did not define a fixed scope of reference architectures, either. The scope of their reference architecture for service oriented architectures was defined as follows (p. 11): OASIS scope

> The Reference Architecture Foundation is not a complete blueprint for realizing SOA-based systems. Nor is it a technology map identifying all the technologies needed to realize SOA-based systems. It does identify many of the key aspects and components that will be present in any well designed SOA-based system. In order to actually use, construct and manage SOA-based systems, many additional design decisions and technology choices will need to be made.

Additionally, a few examples of reference architectures are presented, in order to provide an idea of what a reference architecture can comprise. Figure 5.2 shows the reference architecture for Web 2.0 applications presented by (Governor et al., 2009, p. 87). This architecture contains the main building blocks common to Web 2.0 applications like *Controller*, *Communication Services* or *Databases*. All the various components are arranged in layers that structure the reference architecture and group components with similar objectives. The model also includes a basic flow of data, in order to show – in this case – how to connect *service tier* and *client tier*. Visual examples

Similar types of reference architectures, i.e. reference architectures whose visual representation is built around a layered grouping of domain-specific components including basic information flows, can also be found in e.g. (Paschke, Vincent, Moxey, et al., 2012)[3], (Le et al., 2006), (Microsoft, 2012) and (Stifani et

[2]Section 3.1 defined text stream processing as the generic concept for information filtering, information retrieval and text processing in a data stream context.

[3]This reference architecture is used as a foundation for proposed reference architecture

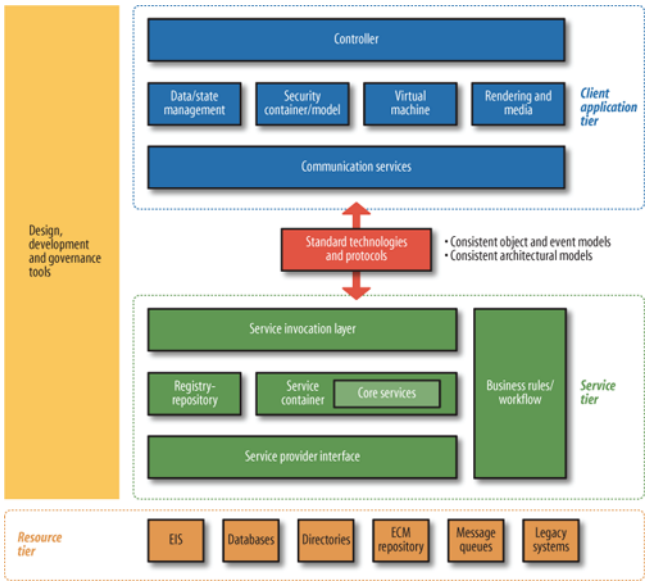

Figure 5.2: A reference for Web 2.0 applications by (Governor et al., 2009, p. 87)

al., 2012). These examples illustrate which level of abstraction and granularity are used for reference architectures and can be used as an indicator for the design of such. Finally, an example of an event processing related reference architecture was presented in (Bruns & Dunkel, 2010, p. 3). They described the basic principles of event processing systems and also structured event processing capabilities along a layered approach that goes from the *monitoring* over *event processing* to the final stage *event handling*. Figure 5.3 shows how they designed their reference architecture.

These examples gave an impression of the broad variety of scopes of reference architectures. It can be summarised that layered approach with simplified architectural components can be considered as an acknowledged approach for reference architecture visualizations, and thus this approach is also adapted for the proposed reference architecture.

In order to develop a high-quality reference architecture a structured design process is required. Therefore, the next section discusses a structured design approach that applied for the design of the proposed reference architecture.

5.1.4 Designing reference architectures

Reference architectures are also a kind of software architecture. Therefore, it is reasonable to apply methods from software architecture design to develop a reference architecture (Angelov et al., 2013, p. 17).
There are a multitude of approaches for designing software architectures.

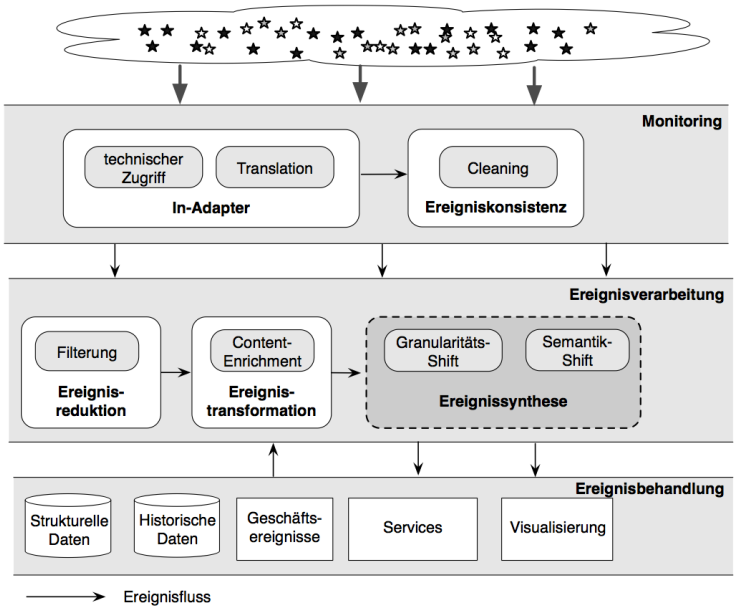

Figure 5.3: Common reference architecture for event-driven system according to (Bruns & Dunkel, 2010, p. 3)

In (Hofmeister et al., 2005) a study of five renowned industrial architecture design methods was presented. They studied among others the *Software Engineering Institute's* (SEI) *Attribute-Driven design* approach, *Siemens' 4 views* or Kruchten's *"4+1" Views* and identified three principal design phases that all five approaches had in common (p. 4). Due to the widespread use of the phases, it was plausible to use this as a blueprint for the design process of the proposed reference architecture. The following phases are used:

Process phases

1. **Architectural analysis:** Defines the problems the architecture should solve together with the scope. This includes the different views the architecture contains and the patterns it provides.
2. **Architectural synthesis:** The core of the design process. Represents the solutions to the questions posed during the analysis phase.
3. **Architectural evaluation:** Validates the proposed architecture by e.g. the evaluation of an implemented solutions based on the architecture or the check of the fulfilment of different quality attributes like performance or scalability.

But this process is a rather coarse-grained description. Thus, it only provides the phases that were used during the design process of the proposed reference architecture. A more concrete outline of what a software architecture is supposed to cover, was presented in the *IEEE: ISO/IEC/IEEE 42010: 2011:*

Process aspects

Systems and Software Engineering, Architecture Description[4] (ISO, 2011). This approach was also used in (OASIS - Organization for the Advancement of Structured Information Standards, 2013), which is an often cited reference architecture proposed by a renowned IT standardisation organisation. The proposed reference architecture was designed using these guidelines combined with the defined previously phases.

According to (ISO, 2011, p. 11) an architecture description should cover the following aspects:

1. Common information about how to identify the architecture and a basic description.
2. The stakeholders and their concerns.
3. A definition of the used architecture viewpoints[5].
4. A description of the content of the used views.
5. A list of known inconsistencies.
6. Rationales and explanations for architecture decisions.

Items 1 to 3 can be assigned to the architectural analysis phase, while aspects 4 to 6 are considered during the architectural synthesis phase. The architectural evaluation phase was not explicitly covered in (ISO, 2011) and was defined as subject to future standardisation efforts (p. 19). Nevertheless guidelines for the evaluation of the reference architecture are required, otherwise the proposed reference architecture could be perceived as arbitrary.

Evaluation

There are several approaches to the evaluation of the reference architecture. (Angelov et al., 2008) introduced an evaluation process based on the software architecture evaluation method *Architecture Trade-off Analysis Method* presented in (Kazman et al., 2000). (Martínez-Fernández, Ayala, Franch, Marques, & Ameller, 2013) also proposed to conduct extensive review meetings, workshops and survey. Both approaches require multiple workshops with various stakeholders[6].

Lightweight evaluation

In the scope and context of the dissertation this approach was not implementable, thus more lightweight approaches were required. (Galster & Avgeriou, 2011) proposed to test a reference architecture by using it in a specific project situation. This seemed more feasible for the purpose of this dissertation and hence the implementation of *StreamFI* was set as the required *specific*

[4] IEEE = Institute of Electrical and Electronics Engineers.

[5] This corresponds to the actual scope of artefacts that are presented in the reference architecture

[6] citep[p. 415]Bass:2012ut stated that a thorough ATAM evaluation can take 20 to 30 person days.

project situation and used as an indicator for the validity of the reference architecture. (Hofmeister et al., 2007, p. 116) argued that the empirical study of implemented architecture prototypes is valid approach for reference architecture evaluation. (Zhu, 2005, p. 225ff) proposed a lightweight approach that was based on the evaluation of defined quality attributes. In summary, the evaluation of the reference architecture is conducted by providing feedback from the implementation of *StreamFI* and by checking the architecture description against the previously introduced goals and properties.

Architecture style

Next, the architectural style must be defined. Examples of architectural styles are *client/server*, *object oriented*, *peer-2-peer* or *event-driven* (Bass et al., 2012, p. 203ff). The architectural style also defines basic principles that are relevant for the design of the reference architecture. In this case, the architectural style is obviously event-driven and thus a set of components like event producers, event consumers or event processing agents are set as default.

Additionally, it is important to notice that many software architecture design approaches are iterative and view based (Hofmeister et al., 2007, p. 115f). Consequently, the reference architecture design process is not a single top-down step that defines every aspect upfront, but rather the process is iterated several times and different aspects are defined, implemented and evaluated. The proposed reference architecture also represents the results of the iterative implementation and improvement of *StreamFI*.

Views

Another essential aspects of software architecture design is the use of views. It is widely accepted in the domain of software and reference architecture design to describe an architecture using different views (Bass et al., 2012, p. 8ff), (P. B. Kruchten, 1995), (Zhu, 2005, p. 94), (Starke, 2014, p. 77f). (Bass et al., 2012, p. 10) defined a view as "...a representation of a coherent set of architectural elements, as written by and read by system stakeholders. It consists of a representation of a set of elements and the relations among them". One of the first to make the concept of views popular in software architecture was (P. B. Kruchten, 1995) who proposed to use "several concurrent views, each one addressing one specific set of concerns" (p. 42). This adds interesting aspects to the definition of view (Bass et al., 2012). First, that various views can be used to describe an architecture, i.e. there is no single view that covers all aspects that are relevant to depict. This also means that different views on the architecture are supposed to provide each of the involved stakeholder groups with the required information. Second, that they should be used in parallel, i.e. for each set of concern or stakeholders, different views apply. These aspects are also emphasized in (ISO, 2011, p. 6).
In software architecture, research various frameworks proposed different views. In order to get an idea of the variety of different views, and in order be able

to assess the scope of the views of the proposed reference architecture, the most relevant views are briefly summarised in table 5.2

View	Authors	Description
Logical view	(P. Kruchten, 2003, p. 542)	Is used to facilitate the implementation of functional requirements by providing an object model or an entity relationship model of domain concepts. This view is often modelled using UML class diagrams. This view could also be considered as a reference model if the view is a applicable to a certain domain.
Process view	(P. Kruchten, 2003, p. 543f)	Addresses quality attributes like scalability or performance. In fact, this view is used to group associated functionality within a process group that can be *tactically* controlled together (in terms of start, restart, fail-over or stopping the process)
Development view	(P. Kruchten, 2003, p. 542)	Is focussed on concrete deployable software modules. In this terms it is similar to the module view introduced below.
Physical view	(P. Kruchten, 2003, p. 542)	This view maps the software to the hardware and is thus similar to the execution view (cf. below), e.g. this view includes the modelling of test, integration and production servers along their physical capabilities and communication protocols.
Conceptual view	(Hofmeister et al., 2000, p. 13)	Consists of conceptual components and connectors. Conceptual components contain the system's functionality. The connectors are responsible for data flows and the component coordination. This view employs domain concepts and is in general technology agnostic.

View	Authors	Description
Execution view	(Hofmeister et al., 2000, p. 13f)	Here the physical deployment of modules on runtime entities (e.g. servers) is described. This means physical deployment on real or virtualised hardware along with the requirements and technical details of the runtime entities.
Module view	(Hofmeister et al., 2000, p. 13)	This view is derived from the conceptual view and is used to form modules and subsystems. In this view the architect addresses how components can be realized with current technologies.
Code view	(Hofmeister et al., 2000, p. 14)	This view contains the mapping of software functionality to deployment components that can be deployed on the runtime entities. Thus this view is concerned with software engineering based structure of executable code.
Conceptual view	(Muller, 2004, p. 25,p. 106)	In contrast to definition above, this conceptual view is supposed to reflect *how* the needs of the stakeholders should be realized. This contains e.g. system and function decomposition.
Functional view	(Muller, 2004, p. 23,p. 88)	This view describes *what* the stakeholders wants to accomplish and uses methods like use case or service decomposition.

Table 5.2: Overview of various kinds of *views* in software architecture.

While several frameworks (cf. (P. Kruchten, 2003), (Hofmeister et al., 2000) or (Muller, 2004)) define a fixed set of views, (ISO, 2011) and (Bass et al., 2012) argue that an architecture description should contain enough information and views to adequately address the needs of the stakeholders. This eclectic approach is also used here, i.e. two architectural views (process view and functional view) were chosen to illustrate the relevant architectural aspects to the group of stakeholders of this reference architecture.

Guidelines

Finally, some guidelines are introduced that helped to shape the reference architecture. (Angelov et al., 2009, p. 143) presented a framework of questions that can be used to adequately describe the previously introduced aspects, *viz.* these questions were:

- What is described?
- How detailed is the reference architecture described?
- How concretely is it described?
- Which graphical or notational representations are used?

In order to avoid that the answer to these questions and architectural design questions in general appear arbitrary, (Greefhorst et al., 2006, p. 109) presented a set of architectural dimension scales that allow clearly categorising each aspects of the reference architecture. Thereby a normed description of reference architecture can be derived that is comparable and understandable due to predefined set of scales. These dimensions were taken into account during the design process where applicable. Table 5.3 provides an overview of the defined dimensions.

5.2 A reference architecture for event based text processing and information filtering

In this section the reference architecture for event based text processing and information filtering is introduced. Based on the process steps that were outlined in the previous section the reference model is developed. The design process was closely related to the implementation phase of *StreamFI* and both phases influenced each other; both were developed iteratively.
Along the three main process phases – architectural analysis, architectural synthesis and architectural evaluation – the reference architecture is described.

Dimension	Description
Type of information	The topic of the information (business, organisation, technical)
Scope	The extent of the information covered (industry sector, organisation, domain, system family, system, component)
Detail level	The amount of detail (high, medium, low)
Stakeholder	The target audience (client, end-user, architect, analyst, developer)
Transformation	The transformation phases that the architecture needs to cover (current situation, short-term, medium-term, long-term)[7]
Quality attribute	The quality attribute that is being addressed (functionality, reliability, usability, efficiency, maintainability, portability)[8]
Meta level	The amount of abstraction (instance, model, meta-model, meta-meta-model, meta-meta-meta-model)
Nature	The nature of the information (policy, principle, guideline, description or standard)
Representation	The way architectural information is represented (formal, semi-formal, informal)[9]

Table 5.3: Architectural dimension scales according to (Greefhorst et al., 2006, p. 9).

5.2.1 Architectural analysis

The first phase is the architectural analysis phase. In this phase, the goals and purpose of the reference architecture are defined as well as which problems it tries to solve. Thereby the user of the reference architecture knows what to expect and who the addressees are. Additionally, the previously introduced process aspects *architecture identification*, *basic overview*, *stakeholders* and *viewpoints* are discussed (Hofmeister et al., 2005, p. 4).

5.2.1.1 Concrete goals and purpose of proposed reference architecture

In section 5.1.2 the goals of a reference architecture were already defined. But to recapitulate, the goal and purpose of this reference architecture is to provide knowledge about the information filtering and text processing domain in the context of text streams and event processing to stakeholders that are not proficient in these domains under given context. As previously argued, event processing, text processing and information filtering together have not been studied extensively up to now. The reference architecture tries to address the following problems:

- What components are required to event-driven text stream processing?
- How do components from text stream processing relate to architecture layers of event processing systems?
- What kind and volume of events are the components supposed to process?
- Which architectural guidelines and pitfalls exist that should be considered when designing a concrete system?

Thus, both the reference model and the reference architecture are intended to provide enough knowledge to build this kind of systems.

5.2.1.2 Common information and basic overview

The aspect of providing a common information on how to identify the reference architecture is not addressed, because this is only relevant in cases where a standardised architecture is proposed. According to (ISO, 2011, p. 12) in this step information about the version history, the issuing organisation, the version control information or approving authorities are given. None of these is relevant in this context. A short summary or brief scope description is also omitted, as this is also only a formal aspect of the document

structure of standardised reference architecture.

5.2.1.3 Stakeholders

(Garlan et al., 2010, p. 61) argued that a reasonable architecture documentation can only be derived if the needs of the stakeholders are clear. Based on the (Greefhorst et al., 2006), the proposed reference architecture is intended for the use of architects and developers.

Software architects are concerned with the structures of the software system, identifying the relationships and the interactions of involved components. They also take into account quality attribute aspects and functional requirements during the design phase. Therefore, they require an overview of text processing and information filtering tasks, and how they are related to enabling components and event classes. The functional view, the process view as well as the architecture patterns that are presented in the forthcoming sections help to address their needs.

But developers can also benefit from the reference architecture, as they are required to implement the ideas from software architects and this task becomes easier if the developer in charge understands the design decisions of the architect. Thus the reference architecture represents guidance through the implementation process as well. Furthermore, software development has become agile and iterative over the past years and thus findings from the development are integrated into the architecture. Therefore, interaction, communication and discussion between the architect and the developer are required and can be fostered by this reference architecture.

5.2.1.4 Used views and scope

Contents of the RA

This reference architecture contains a functional view, a process view, a set of architecture patterns and a discussion on relevant quality attributes. In detail the following artefacts are included:

1. Definition of a functional view as proposed in (Muller, 2004, p. 23,p. 88). This view comprises functionalities that are relevant in the context of information filtering and text processing. They are arranged in a layered representation along the event processing tiers presented in (Paschke, Vincent, Moxey, et al., 2012). Additionally, facilitating functional and execution units are introduced and discussed.

Dimension	Functional view	Process view	Quality attributes
Type of information	Technical	Technical	Technical
Scope	Domain	Component	Component
Detail level	Low	Medium	Medium
Stakeholder	Architect, developer	Architect, developer	Architect, developer
Nature	Guideline	Guideline	Guidelines
Representation[10]	Informal	Informal	Informal

Table 5.4: Overview of dimensions addressed by the reference architecture's scope

2. A process view of enabling building blocks according to (P. Kruchten, 2003).
   In section 2.3 the basic building blocks of event processing systems were introduced. This logical view is intended to describe particularities that have to be taken into account when using the components in a text processing and information filtering context.
3. Discussion of relevant quality attributes.
   (Bass et al., 2012, p. 63) defines a quality attribute as "a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders". In a high-volume text stream setting foremost the aspects of scalability and performance are foremost of importance and thus ideas that help to address these attributes area discussed.

Views like development view, code view or execution view already contain by definition much concrete information about the software architecture. But the scope is a reference architecture, which is by definition more abstract and thus these views are not in scope.

Table 5.4 summarises the concrete dimensions of the three components that are covered by the reference architecture. The classification is based on (Greefhorst et al., 2006, p. 109).

5.2.2 Architectural synthesis

The second phase of the design process is the architectural synthesis phase. In this phase, the concrete reference architecture is described according to the scope that was defined in the previous section.

First, a functional view of essential components is presented. For this the main information filtering and text stream processing tasks are decomposed and arranged along the tiers presented in the functional view of the Event Processing Technical Society's reference architecture (Paschke, Vincent, Alves, & Moxey, 2012, p. 326). The goal is to provide an overview of relevant concepts and functionalities and their semantic relationship by grouping them into layers, the required functional units and execution units. The adaptation of this layered approach seemed appropriate as this is an often used approach for the visual description of reference architectures (cf. 5.1.3).

Second, a process view is introduced that shows how to group related tasks, in order to form an executable unit for certain domain-specific tasks. The goal is to provide a blueprint for how to structure various components from the functional view. Based on the functional view, the process view and the experiences from implementing the research system *StreamFI* a basic set of architectural patterns was derived that summarises the main findings from the design process. Finally, quality attributes are discussed that are relevant in the context of text processing and information filtering.

5.2.2.1 Proposed functional view

Figure 5.4 provides a high-level overview of the complete functional view. Each layer and all components are discussed in more detail in this section.

View structure

The functional view is structured as follows. The innermost area describes the various event processing tiers. Essential domain elements[11] are introduced and their semantic relatedness is modelled using the functional view of the *Event Processing Technical Society* reference architecture (Paschke, Vincent, Moxey, et al., 2012). This layering does not imply that the tasks within a layer are necessarily implemented within the same event processing agent. It rather means that the functionality within a layer has the same degree of quality or complexity of the events processed within the layer. In fact, each of the shown tasks is usually implemented as a distinct event processing agent. Thus, the layering helps to structure event processing agents by visualizing the kinds of complexities that are involved and which tasks must be discerned.

For instance, *stop word handling* in the *event preparation* tier only requires Document Events and emits only Stop Word Events. The process to generate such events is straightforward using a stop word list and is computationally inexpensive. In contrast, *event pattern resolution* is, depending on the pattern,

[11] Domain elements are usually domain relevant tasks.

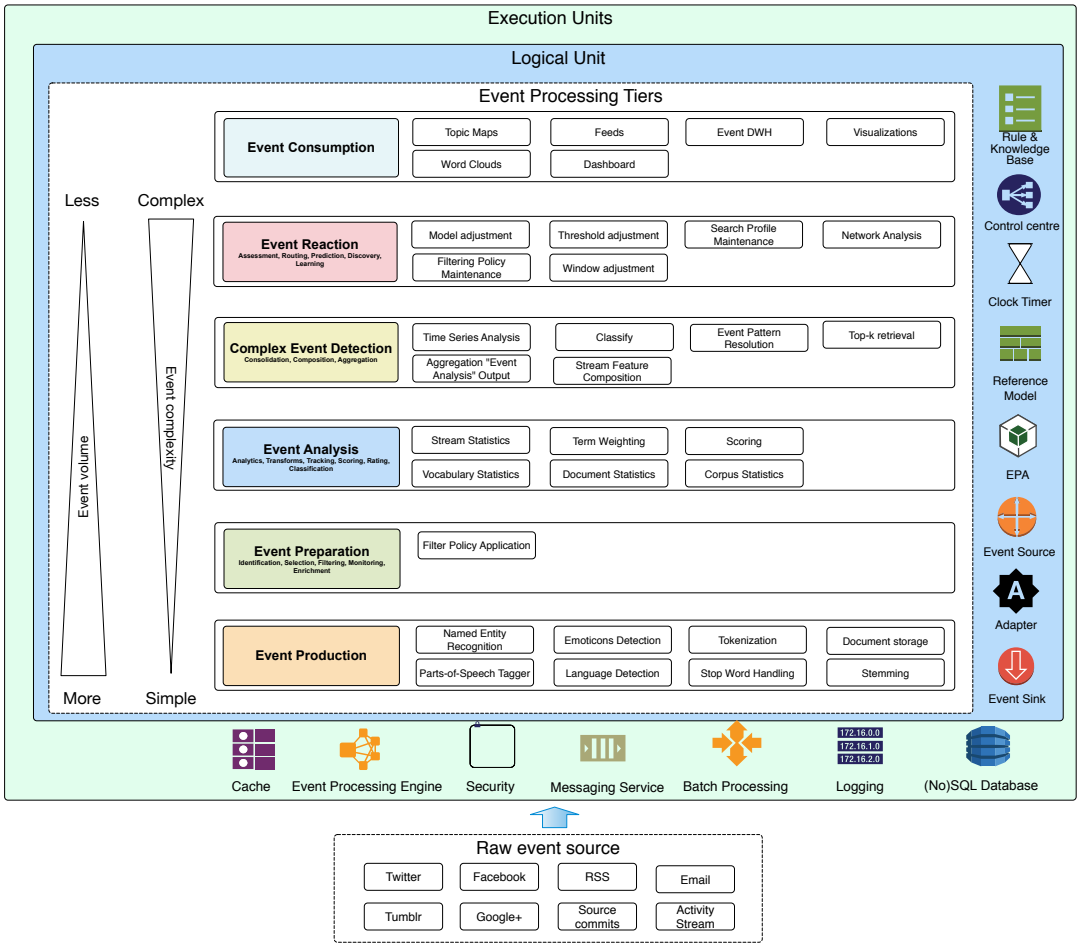

Figure 5.4: Functional view of the proposed reference architecture for event based text processing and information filtering

quite resource expensive and requires several event classes from different layers[12]. Also the layers do not impose a sequence on how events flow, i.e. events can in fact flow between all involved tiers.

The different layers are "encapsulated" by a set of *logical units*. They represent the event processing building blocks as described in section 2.3. The final, outer layer contains the *execution units*. They are required to implement the functional units and the domain elements contained in the event processing layers. The graphical representation of this reference architecture is similar to the one presented in (Governor et al., 2009, p. 83ff).

The elements of the functional view were derived from the most relevant key areas of information filtering and text stream processing that have already been introduced in section 4.2.1:

[12]Cf. 4.2.4.4 for an example of complex heuristic event patterns involving various event types.

- Document pre-processing and index maintenance
- Stream feature modelling
- Scoring and term weighting
- Relevance feedback and query expansion
- Classify documents
- Filter policy generation

Additionally, several other domain related tasks were added and assigned to the different layers, in order to provide a broad view on different tasks in text processing and information filtering.

The next sections describe each layer in more detail using the following structure. First, the tier itself and the associated domain tasks are introduced. Second, the relevant event classes from the reference model are enumerated along a hint on the expected event volume[13]. But before the single layers are discussed, the execution and the logical units are introduced.

Execution units in general Execution units represent the technological building blocks of this reference architecture. They are required to implement the functionality of an event processing system. This section discusses the basic set of execution units that come in helpful when an event based text processing and information filtering system is implemented.

- **Event processing engine**
  An event processing engine is the central component of every event processing agent. It contains the event processing logic written in custom code or in a domain-specific event language. In the context of text processing and information filtering the event processing engine contains the basic set of calculation logic to derive statistics about text streams[14]. There exist numerous event processing and stream processing engines, e.g. *Esper*, *S4*, *Storm*, *Spark*, *HStreaming* or commercial products like IBM *InfoSphere*. The use-case specific functional and non-functional requirements are responsible for the selection of an appropriate event processing engine. An empiric evaluation is required to find out which event processing engine fits best depending on the specific tasks (functional requirements) and non-functional requirements.

[13]The event volume is grouped into *low, medium, high*. This is a relative figure, the absolute amount of events depends on the amount of raw documents flowing into the event processing system

[14]This embraces e.g. statistics about term frequencies, etc. More details on these are provided in the process view.

- **Cache**
  A cache is an in-memory storage that provides fast access to lookup data. This kind of execution unit is very similar to the NoSQL database that is discussed later on, but is presented as a separate unit, in order to emphasize the need for fast lookup capabilities. Caches are required internally as well as externally. For instance, the dynamic corpus that is maintained by an event processing agent requires the capability to update and lookup term frequencies within the event processing agent. An external cache is required e.g. to store search profiles that are accessed by several filtering EPAs.
- **Security**
  This comprises aspects like authorisation and authentication when access from or to the event processing network and the involved raw text sources is required. But this aspect can also be extended to single data streams within the event processing network, mostly if various parties want to access data streams or single events within the data stream[15].
- **Messaging services**
  They form the backbone of every event processing system that relies on inter-node messages. This means, if event processing buildings blocks like sources, agents or sinks do not reside within the same execution unit and thus cannot use shared memory or other in-process exchange methods, message based systems are used to facilitate the communication of buildings blocks.[16] Inter-node communication can also be referred to inter-processes communication.
- **Batch processing**
  This refers to offline processing of data that is collected from the data streams. *Apache Hadoop*[17] is a major enabler in this field of large sized batch processing. But with the decrease of prices for memory and fast solid state disks, the borders between online, real-time processing and batched, off-line processing have started to blur and will probably become obsolete in the near future of data processing.
- **Logging**
  Logging is essential when it comes to tracing errors. Not only the standard error logging is subsumed under this execution unit, but also the

[15] For an example of access policy enforcement in event processing systems cf. (Schilling et al., 2013)

[16] In contrast to this, intra-node communication takes place within the same execution unit and have access to the same allocated memory. Recent Big Data platforms like *Apache Spark* use such approaches. The terms inter-node and intra-node communication are analogous the concept of multi-agent communication and the concepts *inter-agent* and *intra-agent* communication as discussed in (Bocchi & Ciancarini, 2003, p. 150).

[17] `http://hadoop.apache.org`

logging of every single event within the event processing network is subsumed. This means an event processing system should if possible, log every event. Thereby result from the event processing network can be traced back to the causing events. In case of failure, events can be replayed from the event log[18]

- **(No)SQL database**
  In addition to the plain storage of event as logs or in caches, other use cases for data processing can occur. NoSQL databases, as a superclass for technologies like key-value-storages, column-oriented databases or document oriented databases, offer special capabilities than can be helpful in a text processing scenario. A document oriented database can be used to store and retrieve the complete document that formed the origin of all derived text domain events or key-value storages can for instance be used to form a shared index structures within the event processing network.

Proposed logical units On the right hand side of the architecture overview the logical units are placed. They are called *logical*, because they represent general concepts and not concrete module implementations. In fact, they correspond to the essential building blocks of event processing systems as introduced in section 2.3. This includes established concepts like event processing agents, sinks and sources, but also components like a *Clock Timer Component*, a *Control Centre*, *Adapters* as well as the event reference model. The following list summarises the purpose of the logical units.

- The ***Reference Model component*** holds the information about the reference model, i.e. it contains common data types used in every event processing agent as well as the event ontology. Every module that is intended to take part in the event processing network in form of a source, sink or event processing agent incorporates and uses the reference model. Thus all components of the EPN "speak the same language" and can communicate in a semantically well defined manner.
- ***Adapters*** work as connections to the world outside of the event processing network. They handle the access to external sources or sinks. For instance, a Twitter adapter handles the authentication and authorisation issues, receives the raw json Tweet information, and converts it into a representation that can be used by the event source. Adapters are also used within sinks where the event classes are mapped to the

[18] *Apache Kafka* has adopted this strategy to form a high-performing, solid messaging infrastructure. TraditionalDBMSs for decades also have used transaction logs to perform failure-safe database actions.

required format of the consumers outside of the event processing network. The adapter takes over the communication and exchange of the data.

- The ***Event Source*** processes the raw text stream from the adapter, maps them onto the entities from the event reference model and feeds them into the event processing network. It uses adapters to access the raw document sources.
- An ***Event Sink*** works like the Event Source in reverse. It receives events from the event processing network, executes, if necessary, the final processing steps within the event processing network and hands over the results to the adapter that transform the event classes to a representation that can be understood by the consumers of the adapter.
- An ***Event Processing Agent*** is a hybrid of *Source Component* and *Sink Component*. It contains an event processing engine as well as custom code. This separation was made because currently event processing logic and custom code are usually expressed in different languages, e.g. the event processing statements are encoded in *Esper EPL* and the custom code is written in *Java*[19]. In a text processing and information filtering scenario, each event processing agents monitors various text stream properties and keeps an internal frequency corpus, in order to be able to process incoming events.
- The ***Control Centre*** is a component external to the event processing network that is in charge of the coordination and configuration of the event processing network. It can be used to register event processing agents, sources or sinks using the event classes they can handled. Thereby, the complete topology of the event processing network can be automatically derived by querying the event registry. This information can also be used for load balancing by discovering automatically new event processing agents or by instantiating new ones using this information. Furthermore it can be used to start, stop or reconfigure the component of the event processing network[20].
- The ***Central Clock Timer*** is necessary to synchronize the clocks of the nodes by periodically emitting a *Timer Event*. Timing is a crucial aspect in distributed system like event processing networks and thus the requirements of the clock must be agreed on during the system design, especially if patterns based on temporal logic are used[21].

[19] This separation is vanishing, cf. the integration of stream processing capabilities of Microsoft's *StreamInsight* into *C#*.

[20] More details are provided by 5.2.2.3

[21] A detailed discussion of the requirements and pitfalls of time and clocks of distributed systems is out of scope. A seminal paper in this area is (Lamport, 1978) and provides a good

- The *Rule & Knowledge Base* components represents contains rules and knowledge that is required to execute pattern matching or to augment the context. This means that in this component the pattern descriptions are stored and external knowledge, e.g. provided by facilities like *DBPedia*[22] or *WordNet*[23] can be queried to provide an event processing agent with external *world knowledge*, in order to enrich the context and enhance the semantic processing capabilities of the event processing network.

starting point.

[22] `https://http://dbpedia.org`

[23] `https://wordnet.princeton.edu`

Event processing tiers In this section the various layers of the functional view are discussed in more detail. Here, the mapping of the information filtering and text stream processing tasks to the layers of the EPTS reference architecture is made. Thereby, event processing concepts and text stream processing are brought together. This was stated as one of the central goals of this dissertation.

This section is structured as follows: Each layer of the ETPS is discussed in detail and for each layer the relevant text stream processing tasks are introduced, and advice on the expected event types and volumes is given. This information facilitates the design and implementation of concrete software architectures.

A. Event production tier

This event processing tier is exclusively concerned with the reception of raw documents from different source and the mapping to event classes that were defined in the reference model.

Domain tasks

The *Event Production* tier is located at the bottom of the reference architecture. It represents the entry point to the event processing network, and continuously feeds raw text stream data into the event processing network. The goal of this tier is to receive raw documents from text streams via an adapter, map them onto Document Events and emit them via an event source into the event processing network. All Text Domain Events that form the basis for the later analysis are generated in this tier. The event processing agents in this layer accept Document Events that are received from the document sources via a suitable adapter. They split up the documents into single tokens and apply common text processing steps such as stemming, stop word removal, word sense disambiguation, part-of-speech tagging, preliminary sentiment detection (e.g. by taking into account emoticons in microblog text streams) or language detection. The outcomes of these steps are the various Text Domains Events as presented in the reference model.

The implementation of the domain tasks in this layer has to be planned thoroughly, because this has large impact the quality of the text analysis system as a whole. For instance, if too many stop words are eliminated, the performance in terms of filtering quality of the system might decrease, because relevant terms do not make their way into the event processing network. On the other hand words with less than three characters could be filtered, oth-

erwise the event processing network gets flooded by a plethora of useless words that are often only typos or meaningless abbreviations[24]. Hence, it is important to maintain a balance between the number of events that are emitted and the number of events the network can handle. Large scaled event processing networks with large amounts of memory and fast CPUs can of course process more fine-grained events. It always depends on the use case and the research goal for which the event processing network is used.

Parts-of-speech tagging and named entity recognition are resource and computationally intensive tasks. Thus, it is important to thoroughly analyse the processing times of the event processing agents that emit these types of events. Also, the suitability of the tagger or the named entity recognizer for the data source must be verified. Otherwise tagging could not be adequate and the derived events would not reflect the true meaning within the document. This also affects the overall performance of the algorithms that rely on those features.

Event types and volume

This tier emits all kinds of Text Domain events. The event volume depends exclusively on the emission frequency of the raw document sources. In fact, the arrival frequency of the document sources determines the overall event volume within the event processing network; it is also the base for every scaling and performance planning. It must be thoroughly checked if the later tiers and the event processing network in general can handle the event volume. Depending on which event classes are required, each new incoming Document Event can spawn 20 to 30 new derived events. This means, if, for instance, the Twitter fire hose was available, in average 6,000 tweets per second would arrive and this would spawn at least 120,000 to 180,000 events per second only in this tier.

Figure 5.5 summarises the findings.

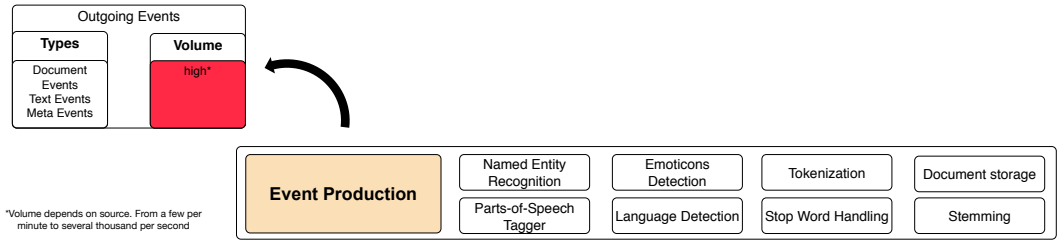

Figure 5.5: Event production tier in detail

[24] If words with less than three characters are filtered there should be a positive list in place that filters popular abbreviations like US, UK or UN

B. Event Preparation Tier

The event preparation tier is according to (Paschke, Vincent, Alves, & Moxey, 2012, p. 327) the first tier to apply actions on incoming events. This entails identification, selection, filtering, monitoring or enrichment. For instance, events are filtered out, because their meta data is corrupt or events are assigned and emitted to a specific event processing network based on their meta data or content.

Domain tasks

In terms of text processing and information filtering the only task that is placed in this tier is *Filter Policy Application*. A filter policy represents a set of rules that filter out obviously irrelevant documents, in order to reduce the load on later tasks like scoring or model application. Filter policies should be chained in order of increasing complexity. This means that the computationally fastest filter policies that remove the most events should be placed before the policies that require more resources.

Filter policies

A filter policy can based on heuristics that are defined by human beings. Such heuristics are straightforward and filter out unwanted documents based on their meta data. For instance, if the system is supposed to process documents in Spanish only, all documents whose language is not Spanish are discarded. The same applies for locations or timestamps. If the system should only process documents that arrive between 1 and 3 am, then documents outside of this time slot are ignored as well. But filter policies can also be more sophisticated and use more complex rules that can be derived using machine learning algorithms. The development of such policies is subject to a tier presented later, the *Event Reaction* tier.

Filter policy application also comprises the *term filter policies*, which are among the most relevant types when it comes to information filtering. This policy validates a Document Event as potentially relevant if one of the search profile terms is contained within a document. For instance, if the search profile string is “soccer world cup 2022” then any document containing one of these terms is considered as potentially relevant and is passed on to the scoring and classification tasks if the other filter policies also apply.
Heuristic filters and term based filters are very efficient means of reducing the load on the event processing remarkably.

Event types and volume

In general, filtering or selecting can be applied to any event class, because every event class has assigned meta data and payload. In the current version, where only filter policy application is relevant, this tier operates only on Document Events.

The main functionalities and data flows are summarised in figure 5.6.

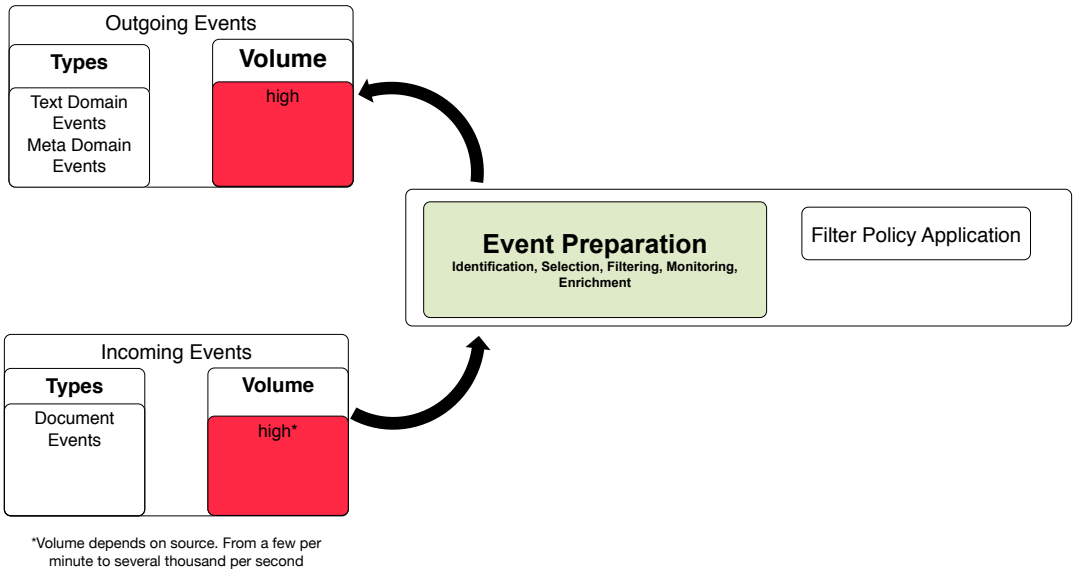

Figure 5.6: Event preparation tier

C. Event Analysis Tier

This tier is concerned with computing values based on the incoming events. This means scores are calculated or the payload of the event is adjusted to comply with requirements of the event processing network (Paschke, Vincent, Alves, & Moxey, 2012, p. 327). In summary, the *Event Analysis* layer covers the "the process of analysing suitably prepared events and their payloads and meta data for useful information" (Paschke, Vincent, Moxey, et al., 2012, p. 91).

Domain tasks

This tier is particularly important n terms of text processing and information filtering. Here counts and statistics about documents and their meta data are calculated. In fact, such values can be calculated for every event class. This is possible, as event processing engines are optimized to operate on data streams and derivation of counts, and statistical moments like average or standard deviation are well operationalisable processes. They are computationally easy to calculate, because they can be either calculated incrementally, for instance average or standard deviation, or approximated within a given error bound (cf. *Count-Min Sketch* algorithm by (Cormode & Muthukrishnan, 2005)).

Statistics Events

Event processing agents pertaining to this tier are supposed to receive a certain type of event, calculate statistics and emit these into the event processing network via a Statistics Event. In general, such statistical values should be calculate for all terms of Meta Events as well as all Text Events. And as mentioned due to the efficient computation of this values, every event processing agent can calculate the required values independently on its own.

Corpus statistics, e.g. relating to the total number of documents or the distinct lemmata within a defined sliding window[25], are required for the calculation of term weights. Different term weighting approaches require different measures to calculate a weight for a given lemma[26]. The scoring functionality uses these term weighting approaches to calculate a score for a given document. The Scored Event is then emitted into the event processing network.

Event types and volume

Incoming events can arrive from almost every other tier and event domain. Outgoing events are foremost Statistics Events and Scored Events. This means,

[25]The definition of the size of sliding windows is subject to the *Event Reaction* tier.
[26]Term weighting is studied in 8.3

a number of statistical values like mean, average, etc. are calculated based on the various Text Domain events and Meta Domain events. In fact, this tier emits such events about every aspect the system should cover. The Statistic Events can then be used in the *Complex Event Processing* tier, where a feature vector for a document can be composed using these statistics events.
Also, scoring takes place in this tier. Therefore, this tier also emits Scored events. Those Scored Events are then processed in the Complex Event Processing tier and filtered out using machine learning algorithms or threshold methodologies[27].

Figure 5.7 summarises this layer.

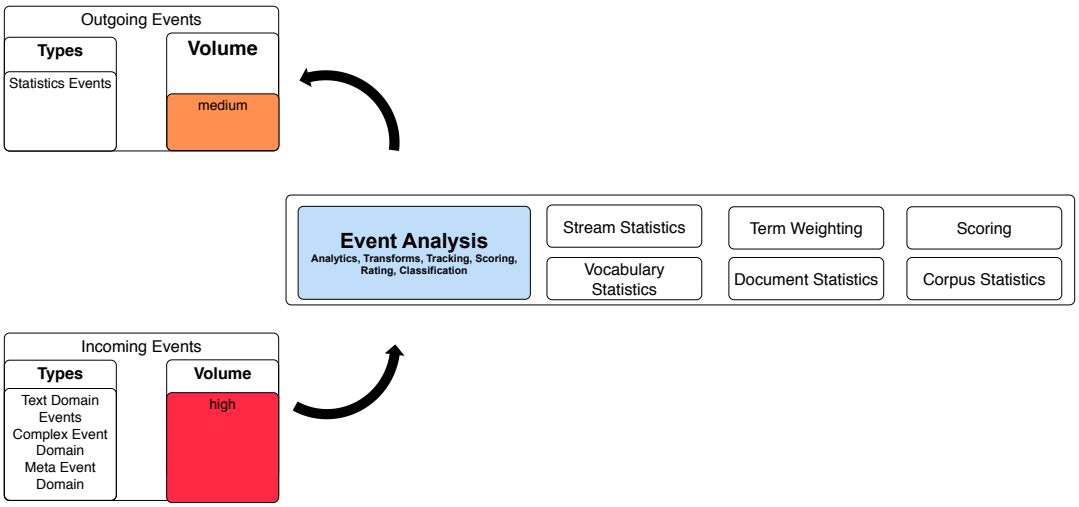

Figure 5.7: Event analysis tier

[27] These aspects are studied in 8.2

D. Complex event detection tier

The next layer of the reference architecture contains the functionality that is used to create *complex events*. According to (D. Luckham & Schulte, 2011, p. 8) a complex event is "[a]n event that summarises, represents, or denotes a set of other events". (Paschke, Vincent, Alves, & Moxey, 2012, p.) describe the purpose of this tier is being concerned with aggregation, enriching and consolidating events as well as with pattern matching.

Domain tasks

Thresholds

The functionality in this tier receives various event classes from the lower levels and tries to derive high quality knowledge from the incoming events. In this tier event processing agents can derive Analytical Events like Classified Events. This means, the Scored Events from the Event Analysis tier are inspected, and based on the defined rules it is decided whether the Document Event is relevant for a given search profile or not. These rules can be either based on a threshold strategy or it can be based on machine learning algorithms. In case of the latter, feature vectors and relevance feedback are required to build and adjust a model that can be used for classification. But in this tier only the model application is situated. The model building process is assigned to the *Event Reaction* tier.

Pattern matching

Furthermore, Heuristic Pattern Events are derived by event processing agents in this tier. This means that a human domain expert has defined an event pattern that the event processing engine, which is associated with an event processing agent, must match in the incoming data stream. It is important to note that such Heuristic Patterns should also include a probabilistic matching aspect. This means that the patterns defined by the domain expert are hardly always able to represent the pattern goal. And without a probabilistic aspect in the patterns, all patterns operate in an *all or nothing* mode. This means that an event sequence is only matched if it fits the given pattern exactly. But as text is usually unstructured, exact matches would discard too many interesting event patterns. Thus, it is important to also take the probabilistic aspect into account that presents pattern matches, even if the pattern is only partially matched[28].

[28]The system *EchoPAT* presented in(Sen et al., 2010) provides capabilities of partial event pattern matches that could be used in this scenario.

Event types and volume

This tier usually receives Statistics Events, Meta Event and Document Events. Text Events are usually not required, because in this tier the raw document and its properties are fetched from the document store and used directly[29]. The classification and thresholding algorithms that are implemented in the event processing agents of this tier usually emit Classified Events.

Figure 5.8 summarises the findings.

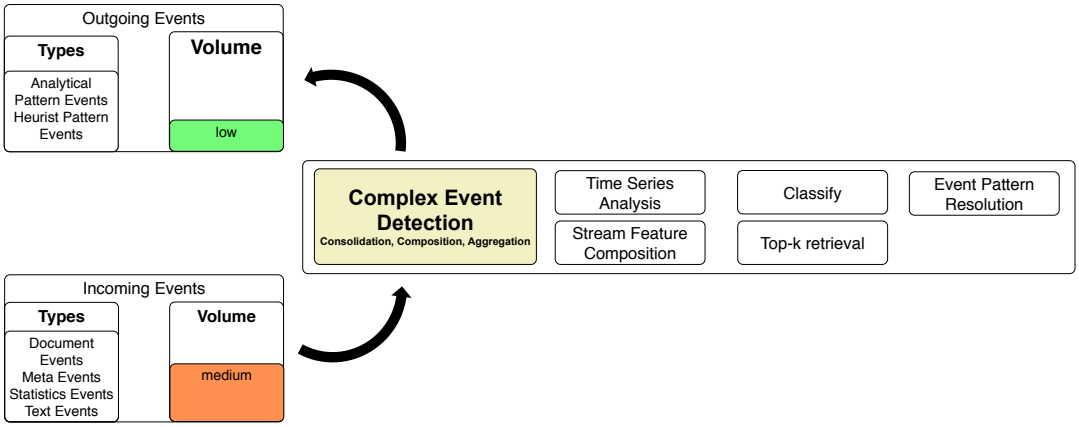

Figure 5.8: Complex event detection tier

[29] More details on the process are provided in section 5.2.2.2

E. Event Reaction Tier

This tier comprises of tasks like event assessment, routing or learning based on event from the previous tiers (Paschke, Vincent, Alves, & Moxey, 2012). The EPAs trigger a certain reaction.

Domain tasks

One of the tasks situated in this tier is the model building and model adjustment process. Machine learning algorithms can be used to learn from the data stream, in order to filter relevant documents from the text stream. As previously mentioned, the model application is situated in the Complex Event Detection tier. But here, the actual model is built using machine learning algorithm.

If a supervised machine learning algorithm is used knowledge about positive and negative examples is required. This information can be provided by Relevance Feedback Events, which belong to the Complex Event Domain. Together with Statistic Events and the document retrieved from the document storage a feature vector can be built that is fed into the machine learning algorithms.

Machine learning

Unsupervised machine learning models, like clustering for instance only require the Statistics Events and the document information from the document storage to build a feature vector. Building a good machine learning model can be computationally expensive. Furthermore, this process still requires interaction with a human machine learning domain expert, in order to refine the model. Thus, model application and model building are situated in different tiers. Nevertheless, the model from the Event Reaction tier must be made available to the Complex Event Detection tier. This could be done by messaging where the model is serialized, sent over a communication model and de-serialized, or if they share the same memory the model can be made available as a *shared variable*[30].

Other tasks in this tier are threshold adjustment, network analysis, window adjustment or search profile maintenance. Search profile maintenance is the process of incorporating relevance feedback information into the initial search profile. A search profile only reflects is in its simplest form just a query string without further search operators like wild cards, etc. But as the initial query string usually does not reflect all the aspects that should be covered by the query, the search profile needs to evolve and for this relevance feedback is

[30]For instance *Apache Spark* provides *broadcast variables*, which represents read-only variable available to involved nodes (Apache Spark, 2015a).

used. This includes the tracking of documents and terms of relevant and non-relevant documents. Both kinds of information are used to calculate query expansion terms that can be used as additional term filter policies (cf. section 5.2.2.1). Statistics Event can also be used to adjust thresholds. For instance, the average score per non-relevant document could be tracked or the average score of all relevant documents for a given search profile[31].

Window adjustment

Window adjustment is concerned with the task of setting the size of the sliding window to an adequate value. In a static scenario where the window size is always the same, this task can be ignored. But in a highly dynamic setting like text streams it is desirable to adjust the window size accordingly. Therefore, different stream properties can be monitored and the window size is adjusted[32].

Finally, network analysis can be used if for instance the social graph of the user who created the search profile is also used[33] based on some decision rule.

Event types and volume

Mostly Statistics Events and Classified Events are used in this tier. Usually it does not emit any event, but uses the incoming information to adjust parameters of the event processing network. This includes the adjustment of the size of the sliding window, the re-calculation of threshold values or the update of machine learning models.

The main aspects are summarised in figure 5.9.

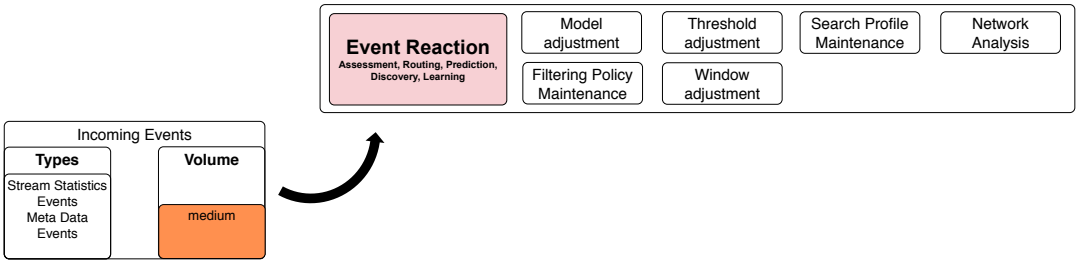

Figure 5.9: Event reaction tier

[31] The study of different threshold models in subject to section 8.4

[32] The automatic determination of sliding window sizes using *Index Entropy* is studied in section 7.6

[33] In this dissertation only text features are used as not enough social signals were contained in the available corpora and the focus was in fact only on the textual features.

F. Event Consumption Tier

Domain tasks

The top layer is dedicated to *event consumption*. As text is often un- or semi-structured, a comprehensible visualization of the results of the event processing network is crucial, in order to allow users to take action. This can be dynamic tag clouds, stream graphs (Byron & Wattenberg, 2008) or tree maps which illustrate the context of text events. The rich semantics provided by *Text Stream Property Events* and stream graphs allow visualizing the dynamics of text streams. The visualization of the output of an text stream EPN faces the same challenges as *graphical user interfaces* GUIs from other domains, i.e. results must be traceable; hence a GUI must allow for an informative drill down, in order to allow the user to understand how the presented results were created.

Event types and volume

The event consumption tier processes all kinds of events. Usually, in this tier event sinks are situated that transform the incoming events into a format that the required visualization understands. This is done by suitable adapters. In total, the volume of events in this tier is rather high, but can be addressed by splitting up the event sinks to handle specific purposes, so that they have to deal with fewer events.

A brief summary of the tasks in this tier is given in figure 5.10.

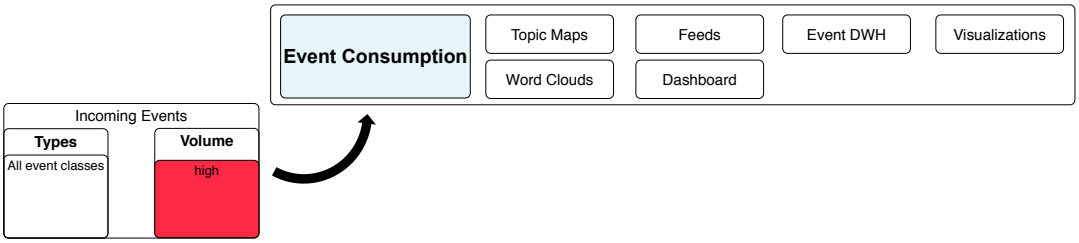

Figure 5.10: Event consumption tier

5.2.2.2 Process view

According to (P. B. Kruchten, 1995) a "process is a grouping of tasks that form an executable unit". This means the goal of a process view is to group semantically associated functions from the functional view into units of execution. These execution units[34] are then used to perform the various reference processes from section 4.2.5.
In this section the majority of the reference processes are described using one process view that shows the process of an event-driven information filtering system. The mapping of the various tasks on different event processing building blocks (source, event processing agents, sinks, etc.) is described, in order to provide a blueprint and insight into how such systems can be implemented by describing various design decisions. The process view also reflects the experiences from the implementation of the *StreamFI* system.

Information filtering Figure 5.11 provides an overview of the information filtering process, whose steps together with several design decision are now described in more detail. The notation is data-flow oriented and uses the artefacts from the functional view. The view contains all data flows where only few point-to-point connections were required. Connections that would have had to be drawn from n to m event processing components or from the execution units to the EPN components are only represented by a single data-flow that starts or ends at the border of the event processing network.

Concise context

The first design decision that was made is that each event processing agent represents a distinct functionality, i.e. a single event processing agent implements only as much functionality as required to modify properties of the incoming event or to instantiate a new event class. If intermediary results are generated that could be mapped to an event class from the event reference model than it is preferable to instantiate this event class and emit it so an event processing agent that is specialized on this event class can operate on it. This helps to keep clear semantics and facilitate separation of concerns within the event processing network. It is also easier to tune and optimize single, independent components in case of performance issues.

Receiving raw documents

The Document Source implements an adapter to the raw text source. For instance, if Twitter should be accessed the Document Source contains the logic for authentication and authorisation against the Twitter stream API. The source can be activated via a Control Event and it starts receiving docu-

[34] The difference between these execution units and the ones on the bottom of figure is that the latter describe infrastructure components that "execute" arbitrary work while the former execute a specific semantic action.

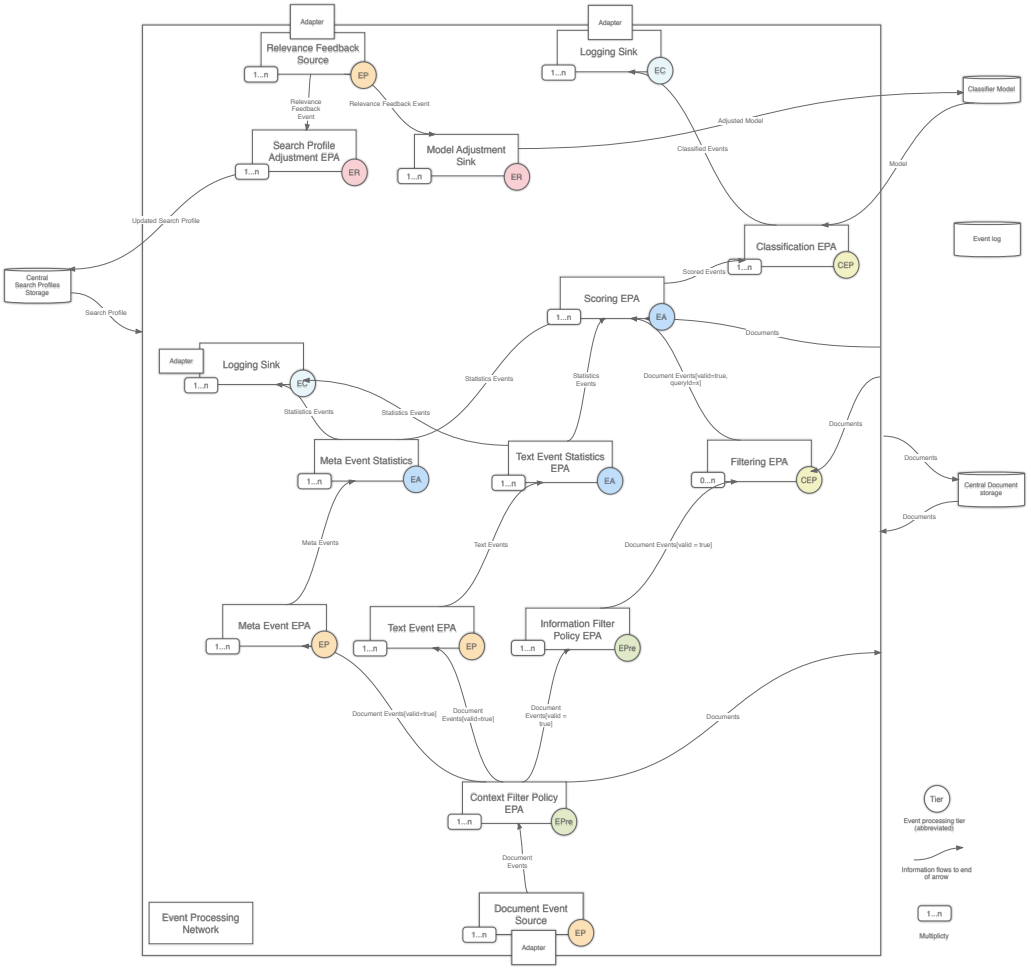

Figure 5.11: Filtering process in an EPN

ments from the raw text stream source. The incoming raw events are mapped to Document Events that are emitted into the event processing network.

Meta and Text Events

The Meta Event EPAs and the Text Event EPAs extract the features of the document, i.e. lemma count, stop word count, punctuation count, etc. and map them to Text Stream Property Events. Additionally, they save this information to an external data store. This is necessary, because the raw document features are required for scoring, filter policy application and model learning. This means, they are required within several EPAs and it would be computationally inefficient to recalculate the document features over and over again. Therefore, they are saved along with the raw text in a fast document storage from where the information can be retrieved quickly when needed[35]. As the

[35]During the physical setup of the document storage it must be considered how to provide each node on which EPAs are executed, with a synchronized document storage instance.

initial document and its features are only saved once and are never modified later on, fast and concurrent read-only access is possible and thus this centralized component is an efficient approach to sharing the document state within the event processing network.

The next step already involves filtering. There exist two kinds of *filter policy* EPA that act as a gate keeper to later event processing agents. The first is the *Context Filter Policy EPA*. This EPA removes all documents that are *definitely* not relevant to the pre-defined context of the event processing network, i.e. clear rules can be defined that describe which requirements a document must fulfil, in order to be emitted into the event processing network. Often this is done base on meta data and is focussed on removing documents that do not have the required language, geolocation information or timestamp. The second filter is the *Information Filter Policy EPA*. This EPA is now used to reduce the amount of Documents a Scoring EPA receives as much as possible by applying rules that already address the content and the potential relevance to the information need of the user. This filtering process is not yet related to a specific search profile, but filters out documents based on textual heuristics, e.g. the document must contain at least one link and two hashtags, or rules that are learnt via a machine learning algorithm. In section 8.2 filter policies are discussed in more detail and several approaches are evaluated. The results of this EPA are Document Events whose valid attribute is set to true.

Filter Policy EPA

Filtering EPA

The valid Document Events are then processed by *Filtering EPAs*. In this step a Document Event and the pertaining document are related to a concrete search profile by using term matching[36]. This term matching process can be based on plain Boolean matching, but could also include more sophisticated approaches using synonyms or search operators like wild cards, placeholders, etc. The multiplicity indicated by in lower left corner of the Filtering EPA also indicates that multiple instances can exist within the EPN as well as per search profile. This approach can be chosen if a search profile has already grown and contains many search terms that should be filtered, or if it contains multiple complex search operators that require a certain processing time. Each Filtering EPA only matches a Document Event against the information it received, when it was instantiated, i.e. it only contains a subset of the search profile. The complete profile, including the complete search string and positive and negative query expansion terms, is first used in the Scoring EPAs.

Document storage should be available at each node in order to avoid network latencies that add undesired overhead and process delay.

[36] As mentioned earlier, in this dissertation only textual features are considered. Filtering based on the social graph of a user or collaborative filtering approaches are not considered.

Scoring EPA

After a Document Event has been assigned to one or many search profiles, the *Scoring EPA* listens for events that have the *valid* property set to true and a non-empty *queryId*. Again, by decoupling the filtering from the scoring, performance and scalability are facilitated, because if the number of total event or events for a given search profile increases, new Scoring EPAs can be spawned, and a load balancing algorithm can distribute the load among the EPAs. The spawning and adding of new Scoring EPAs is also possible, because every new Scoring EPAs receives the same Statistics Events as the already existing ones. Thus, all Scoring EPAs have the same knowledge about the text stream and thus they have the same information that is required for the various scoring algorithms. Within the Scoring EPA the document information and the complete search profile are required to calculate how many matching features are contained in the document[37]. Therefore, the document is fetched from the central document storage and the search profile is fetched from the central profile storage. After a score has been calculated, the EPA emits a *Scored Event* that is further processed by the *Classification EPA*.

Classification EPA

The Classification EPA applies an algorithm to determine if a Scored Event may be relevant given search profile. There are two approaches to this process. The first is based on thresholds that try to provide a linear separation between the class of the relevant and the non-relevant documents. As most of the scoring functions are linear combinations of term weights and thus deliver monotonically increasing score values, the approach for the threshold based approach in this dissertation is to use a statistical moment like average or median to separate the two relevance classes. The second approach is based on machine learning. For this a classifier, for instance *Naïve Bayes* or *Support Vector Machines* is trained using the information from Relevance Feedback Events and the feature vector that was derived from the document and Statistics Events[38]. The Statistics Events are required to normalize or standardised the document features, so the features are within the same scale and the machine learning algorithms can learn a reasonable model.
When the classification has been completed, a Classified Event is emitted. This event is then sent to an event sink that presents the result to the user. Additionally, the event is also stored in the event log, in case of drill down or debugging.

Relevance Feedback EPA

If the Classified Event was presented to a user and the user provided relevance feedback, this information is collected and sent back into the event processing network via an event source that accepts relevance feedback in-

[37] This approach corresponds to standard Boolean term matching. But within the Scoring EPA, other models such as the Extended Boolean Model could also be used.

[38] Here also the central document storage is required to retrieved the document for this purpose

formation. This information is used in two ways. First, it used to adjust the learnt classification model and second, it can be used to adjust the search profile. For the latter case a *Search Profile Adjustment EPA* is used that applies a relevance feedback algorithm[39] and modifies the search profile. The update is written back to the central search profile storage. A central search profile storage is required, because there can be several Scoring EPAs for a single search profile and they should have the same information available[40].

In general, all the event processing network components are only loosely coupled via the event classes they accept, i.e. they do not share information among each other, but exchange necessary state changes via events. Common information that is required by multiple EPAs is only shared via a shared state facility like the central document storage or the central search profile storage.

In this section various domain tasks from the functional view were mapped to distinct event processing building blocks. Also the flow of information and design rationale of each event processing building block was described. In the next section a basic set of event processing patterns is presented that summarises the experiences that were made during the development of the functional and process view.

5.2.2.3 Patterns for event based systems

From the process view and the architectural design decisions that were made, a few architectural event-driven patterns can be distilled that are relevant in this context. The following set is not an exhaustive list of patterns for event-driven systems, but it describes a few patterns that have not been described in the context text processing and event processing before and that can be handy when implementing event-driven text stream processing systems. The principles of the patterns are based on the Actor Model presented (Hewitt, 2010)[41], ideas from domain driven design (Evans, 2004, p. 335,354) the basic principles of event processing networks presented by (D. Luckham, 2002, p. 211f)[42].

[39] Various approaches are studied in section 8.5.

[40] The shared state approach in contrast to a distributed state approach (cf. next section) was chosen, because this implementation impose less development effort and provides acceptable performance.

[41] The principles of the Actor Model were presented in section 2.3.8.

[42] It must be noted that this section is about architectural patterns and not distributed algorithms. The latter are beneficial for implementing the former, but they are used to solve the requirements that are defined by the architecture. Therefore, seminal algorithms such as *quorum*, *leader election*, *Paxos* or *consensus* are not discussed here. For further details cf. (Lynch, 1996)

Pattern structure

The patterns either address event processing directly or specific text processing aspects in an event processing setting. The pattern description is based on the structure presented in (Gamma et al., 1994, p. 6ff), but customized where necessary. Table 5.5 summarises the structure as introduced by (Gamma et al., 1994) and which aspects are used for the patterns described here.

The patterns described here are also in contrast to other pattern collections in the area of event processing, as these are architectural patterns, while the pattern presented for instance in (Paschke, Vincent, Alves, & Moxey, 2012) or in (Bruns & Dunkel, 2010, p. 177ff) can be considered as *functional* patterns. Examples for such functional patterns are *filtering*, *enrichment*, *transformation*, *routing*, etc.

Structure element	Description	Used
Pattern name and classification	A good and concise naming that represents the essence of the pattern along. If numerous patterns are introduced an organisational scheme is helpful to structure the patterns.	Name, yes. Classification, no.
Intent	A brief statement that tells what the patterns intends to achieve and which problems it addresses.	Yes
Also known as	Another name under which the pattern is known	Yes, if available.
Motivation	A scenario that illustrates the design problem	Yes
Applicability	Situations in which the pattern can be applied. When does it not make sense to apply the pattern.	Yes
Structure	Graphical representation of the pattern using a consistent notation.	Yes, if applicable.
Participants	Class and objects that are involved in the pattern.	No, it is not about object oriented design pattern.
Consequences	Which trade-offs must be made when the pattern is used? What are the results when the pattern is applied?	Yes

Implementation	Concrete hints on how to implement the pattern in a certain technology.	Yes, if applicable and without focus on a specific technology.
Sample code	Code sample	No (cf. previous)
Known uses	Examples from real-world scenarios	No[43].
Related Patterns	Other patterns that are closely related	Yes, if possible and available.

Table 5.5: Pattern structure according to (Gamma et al., 1994)

Below, several basic patterns are described that evolved in the course of the dissertation. Not all of them were applied during the implementation of *StreamFI*, but they were considered as possible solutions or considered as useful if *StreamFI* is supposed to be extended. In general, the pattern are technology agnostic, but when it seems beneficial examples using a specific framework are given. The choice of a specific technology depends on the use cases, the budget, the available infrastructure and the knowledge of the architects and developers.

At the time writing this dissertation a couple of technologies that would be suitable are *Apache Spark*[44] or *Apache Storm*[45] for building large-scale event processing network topologies, actor frameworks like *Akka*[46] or *S4*[47], cluster resource managers like *Apache Mesos*[48], *Apache Zookeeper*[49] for reliable distributed coordination to be used for the control centre, or high-performing message infrastructures like *Apache Kafka*[50] or ØMQ[51]. But due to the rapidly changing technology landscape, a mapping of components from the reference architectures of the aforementioned technologies limits the general purposes of the patterns and is therefore omitted.

[43] The only available systems for comparison is the system implemented for this dissertation and thus this column would always contain *StreamFI*

[44] `https://spark.apache.org`

[45] `https://storm.apache.org`

[46] `https://akka.io`

[47] `https://incubator.apache.org/s4/`

[48] `https://mesos.apache.org`

[49] `https://zookeeper.apache.org/`

[50] `https://kafka.apache.org`

[51] `https://zeromq.com`

P1 – Concise EPA context

Intent

Restrict the tasks, which an event processing building block executes, to a minimal semantic scope.

Also known as

Small-step processing (D. Luckham, 2002, p. 212), Micro services

Motivation

Event processing systems are distributed systems with many moving parts. If an error occurs debugging, tracing, optimization or maintenance is difficult if the process logic is spread over various components.

Applicability

By limiting the scope of an event processing building to a limited and closed process context that represents a semantically clear unit, these tasks are alleviated, because errors, optimization or maintenance can be done in a single spot without affecting other components. This pattern can applied during each design process of a concrete event processing building block.

Structure

N/A

Consequences

A strict separation of concerns by means of semantically concise event processing building blocks means that an event processing system consists of more individual components than are more monolithic approach that executes several semantically distinct tasks. The consequence is that tracing or debugging can span various components until issues are found. This increases the complexity of these tasks, but if the issues are found they can be addressed within a single processing unit and other units are not affected. This trade-off should be considered.

Implementation

An event processing source should only emit one type of event classes (cf. reference model). Event processing agents can accept various types of event classes, but should emit only one specific event class. If during the logic execution of the event processing agent an intermediary result can be mapped to an event class of the reference model, then it should be emitted, because this is a good indicator for the end of the scope of the event processing agent. This also means that an event processing agent can perform several semantically distinct tasks as long as the event class does not change. For instance, filtering and enrichment of an event can be combined within a single event processing agent, because the filtering task only sets the *valid* flag or updates the *context* property, while the enrichment task modifies the payload for a certain context without changing the event class. For instance, filtering a document by a term filter and applying other heuristic filter policies does not have to be split up into distinct EPAs, because the event class does not change. The advantage here is that a concise context is given in terms of the processed event class. The overhead of serializing and de-serializing between EPAs is thereby also omitted. Event processing sinks can also accept multiple event classes, but only if the subsequent use within the event processing sink of the events is within the same semantic context, e.g. one sink can accept different event types and write them to a log file. For instance, there is no need to have separate logging sink for Lemma Events and Hashtag Events.

Related patterns

N/A

P2 – Shared state

Intent

Share relevant information among event processing agents without congesting the event channels unnecessarily.

Motivation

Large objects, which contain information that needs to be shared among various event processing agents, for instance search profile, relevance feedback corpora or a text collection, can have serious impact on the processing performance of the event processing network, if this information needs to be sent around redundantly and continuously to the various component that required the information.

Applicability

This pattern should be applied if the event channels within the event processing network or the physical infrastructure already have to deal with a large volume of events. A shared state is also easier to implement and errors are easier to trace.

Structure

Figure 5.12 shows a basic structure of this pattern.

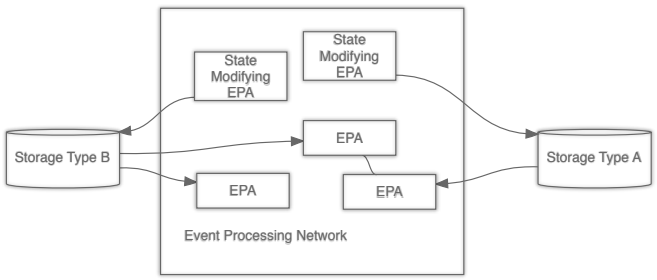

Figure 5.12: Structure of shared state pattern

Consequences

Four consequences are introduced by this pattern. First, the management of an additional infrastructure component is required. Second, more physical memory is required as the storages should be in-memory storages, in order to provide decent performance. Third, concurrent read/write access to the storage can introduce further latencies as well as small inconsistencies, because exclusive locks during write access should be avoided. But these consistencies are negligible in terms of information filtering and text processing, because the unstructured nature of text already introduces more ambiguities than a scoring algorithm whose score is based on slightly differing counts. Fourth, a shared state can introduce synchronization issues and can be also the cause of bottlenecks if the storage cannot serve the requests on time. Therefore, it must be deliberated whether a shared state or a distributed state (P7) are better suited for the purpose.

Implementation

Each event processing agents needs a connector to the in-memory database where the required data is stored. For this a generic interface can be designed that specifies the methods the storage should support. Then each event processing component that is is need can instantiate the required connection to the storage. For instance, *StreamFI* used a Redis[52] based storage and each event processing component was made aware of the connection properties for the storage and could thus access the shared state stored in the Redis storage. The setup and configuration of the in-memory database is product-specific and thus out of scope.

Related patterns

P5 – Dynamic EPA, P3 – Distributed state

[52] `https://redis.io`

P3 – Distributed state

Intent

If synchronization or access to a central state storage is prohibitive, the state must be replicated and transported to event processing building blocks that require the same information.

Also known

N/A

Motivation

A shared state (as proposed in P2) can introduce dependencies on storage component that can cause problems with performance (if synchronization is required or the shared state facility stalls) or inconsistencies[53]. In order to avoid dependencies and latencies and to improve concurrent processing and performance, it can be beneficial to replicate the state among all event processing agents that require the same information. This approach is also put forth by the Actor Model (Hewitt, 2010, p. 5).

Applicability

This pattern should be applied if a shared state cannot be made stable, reliable or to perform well. It can also be applied if there are enough network resources to handle the duplicated data and the processing power of the event processing components is sufficient to deal with the state information as well as the event information.

Structure

Consequences

A rather obvious consequence of this pattern is that information needs to be replicated as often as there are event processing agents that require the same information. For instance, if several dozen dynamic scoring EPAs are spawned by a single search profile based on query expansion (cf. P5 dynamic filter EPA pattern), all these new scoring EPAs need to receive the updated

[53] As mentioned in P2 due to the messy nature of text slight inconsistencies can be accepted, because the real and exact state of the text stream can hardly be determined at all.

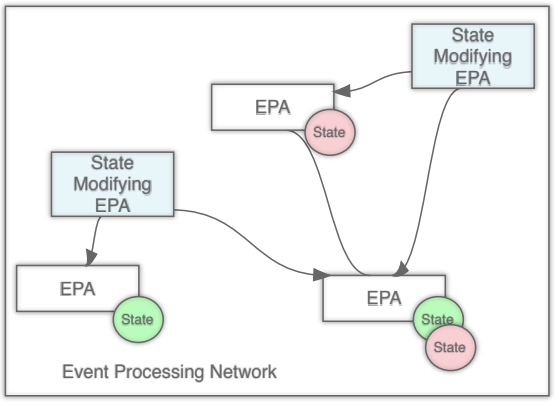

Figure 5.13: Structure of distributed state pattern

search profile, in order to be able to perform document-query matching using the same search profile information. The asynchronous information passing of all such non-event data also introduces overheads regarding message serialization and de-serialization as well as additional load on the physical network channels. These consequences must be compared to the consequences of the shared state pattern.

Implementation

Event processing building blocks require additional message adapters for all the non-event information required for their intended functionality. Thus, a component requires more communication interfaces.

Related patterns

P2 – Shared state, P5 – Dynamic filter

P4 – Control events

Intent

Modify the behaviour and configuration of an event processing network during execution.

Also known

Distributed coordination, dynamic configuration

Motivation

Event processing networks should be dynamic and adaptable, i.e. they should be able to adjust to different situations by creating, removing or updating components of the event processing network. Therefore, means are required to control the behaviour of the building blocks of the event processing network.

Applicability

This pattern can be used when parts of the event processing network should be dynamically created, deleted or modified. Also if load balancing, EPN topology reconfiguration are required, this pattern should be considered. This contradicts partially the *free of commands* pattern described by (Chandy & Schulte, 2009, p. 10), as control events require an event consumer to perform a certain action. But it only contradicts partially, as the control event does not tell the consumer how to perform a business functionality, but rather to execute some maintenance, non-functional, configuration related action.

Structure

Figure 5.14 illustrates the Control Events pattern.

Consequences

This pattern allows to dynamically scale the event processing network as required. Configurations can be changed during the execution of the network without the need to stop and restart the complete network. Nevertheless, depending on the required capabilities of the control centre, this can require major implementation efforts.

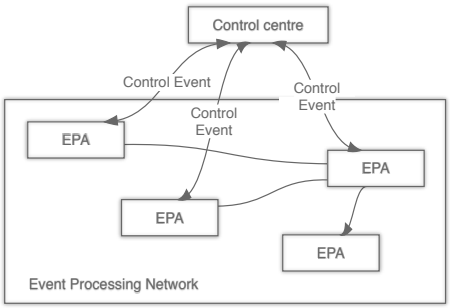

Figure 5.14: Structure of control events pattern.

Implementation

A control centre is required that emits the required Control Events. A basic implementation should contain facilities to start, stop and remove event processing agents.
Furthermore monitoring of the status of the EPAs is required. More sophisticated control centres can balance the load within the event processing network as well as configure running EPAs to use inter-EPN communication or intra-EPN communication on demand. This means that depending on the load, dynamically new EPAs with the required communication pattern can be instantiated. The EPAs in the event processing network also need to emit Control Events. This events contain information about their state and provides feedback to the Control Centre.
Finally, the control centre can also contain a central registry where event processing buildings can register themselves along their connection parameters and also dynamically lookup event processing sources or agents that emit the event classes they require.

Related Patterns

P5 – Dynamic Filter EPA, P7 – Hybrid communication

P5 – Dynamic Filter EPA

Intent

Create and remove autonomously Filtering EPAs from the event processing network.

Also known

Dynamic EPN composition (D. Luckham, 2002, p. 211))

Motivation

Based on relevance feedback the initial search profile is modified, for instance by using query expansion. In order to keep up the process speed of each single Filtering EPA and in order to avoid processing losses due to halting and starting modified Filtering EPA, it is beneficial that several EPAs share the load of term matching. Thus, based on new feedback terms, new Filtering EPAs are spawned *and*, if within a given time frame no matching document is found, again removed from the network.

Applicability

This pattern can be used in scenarios where relevance feedback and query expansion are used.

Structure

In figure 5.15 the structure of the dynamic filter EPA pattern is shown.

Consequences

By using this pattern the initial search profiles are up and running all the time, because no downtime for reconfiguration is required. Also computationally expensive matching strategies like wildcards or range matching can be balanced to different EPAs. The dynamic kill feature also ensures that unused EPAs are removed from the EPN to keep it clean. A downside is that depending on the amount of initial search profile several dozen or hundred dynamic EPAs could be generated, which can have serious consequence to resource consumption. Thus, resources and the amount of dynamic EPAs should be controlled.

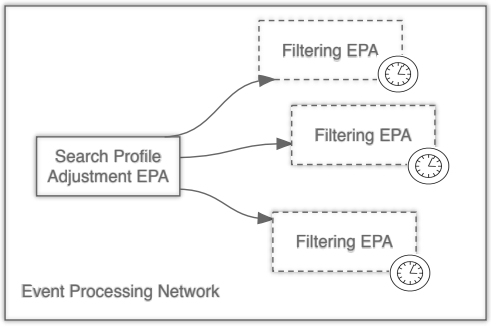

Figure 5.15: Structure of dynamic filter pattern

Implementation

When a Relevance Feedback Event is processed and new relevance feedback terms are derived for expanding the initial query, a new Filtering EPA gets spawned that monitors the stream for this particular query expansion term. The dynamic EPA is created together with an idle time after which the EPA deletes itself.

Related Patterns

P4 – Control events

P6 – Filter policy

Intent

Narrow the focus of the event processing network by removing obviously irrelevant events.

Also known

Context filter

Motivation

If numerous event sources are connected to the event processing network the probability is high that many events are not relevant for analysis at all. For instance, a scoring algorithm should not be concerned with scoring Spanish documents, if the focus is on English documents only.

Applicability

This pattern can be used if the volume of events is large, but only few are relevant at all. Thus, if no proper reduction happens, computationally expensive stages of the event processing network would get flooded with obviously irrelevant events. In order to decrease the load on the components of the event processing network, a context filter can be implemented that filters obviously irrelevant events before they can enter later analysis stages. For instance, an event processing network is only interested in Spanish documents that occur between 10 and 11 am all other events that do not match these rules can be rejected without further processing.

Structure

Figure 5.16 depicts the pattern.

Consequences

The load on the event processing network is decreased and the purpose of the event processing network is clean and concise. But depending on the approach that is chosen for the filter policy generation process[54], it is important

[54] Filter policy generation is discussed in detail in section 8.2

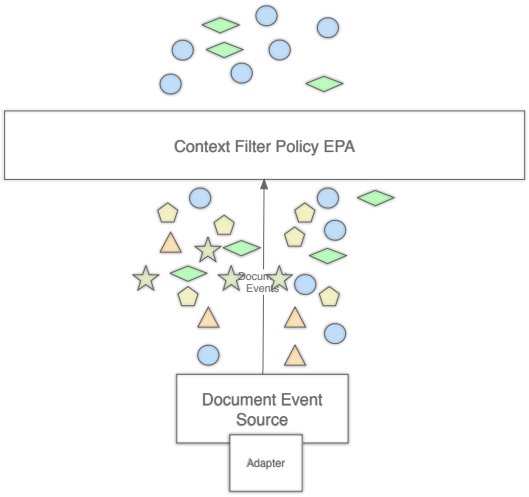

Figure 5.16: Structure of context filter pattern

not to remove too much documents and to keep the rate of false negatives low.

Implementation

First, filter policies can be chained. This means the output from one filter policy, works as input for the next one. Second, all filter policies contain a set of rules. All events that do not match the rules are rejected.

Related Patterns

P1 – Concise context

P7 – Hybrid communication

Intent

Event processing building blocks should support different communication patterns.

Also known

Message passing / shared memory

Motivation

Event based systems that are supposed to process a large amount of data should be able to scale up as well as scale out if required. Therefore, facilities are required to address this requirement.

Applicability

This pattern can be used when the event processing network is supposed to scale, but the scaling mechanism is not supposed to be predefined or needs to be flexibly chosen. Scaling up improves the capabilities of a physical node – often by adding more memory – and allows a single node to process more data. Thus, more data can be processed locally and the network overhead is reduced. This is called an intra-node communication, because the communication between the parts of the event processing network are kept local. If scaling out is required, i.e. adding new nodes to the systems, it must be possible to communicate with other physical nodes. This is called an inter-node communication.

Structure

Figure 5.17

Consequences

The first consequence is that the event processing network is able to scale in flexible manner. But, as a trade-off the complexity of the event processing components increase, as they must be able to deal with both communication paradigms, as well as the complexity of the control centre increases, because

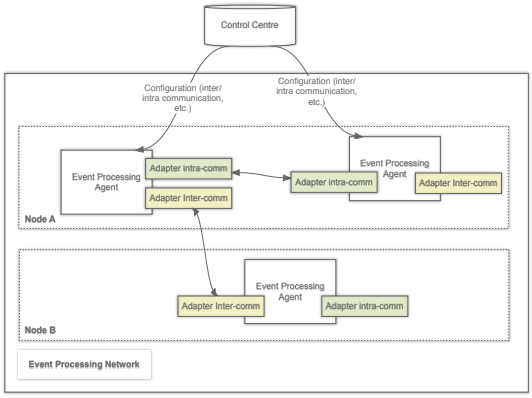

Figure 5.17: Structure of hybrid communication pattern

it must keep track of all used communication schemes and prevent incomplete data supply to event processing agents.

Implementation

In order to implement this pattern, an event processing component needs facilities for exchanging messages and events over network as well as facilities for shared-memory usage. This means, the components requires a network address and means to dynamically join, for example, a running JVM process. Additionally, the components must be able to process Control Events that configure the intra-node as well as the inter-node communication. Also it is important to note, that the Control Centre must keep track which communication scheme an event processing agent implements. This becomes foremost relevant, if inter-node and intra-node communication schemes are mixed.

Related Patterns

P1 – Concise context, P4 – Control events, P8 – Degree of distribution

P8 – Degree of distribution

Intent

Allows an event processing network to be explicitly distributed, implicitly distributed or a mixture of both depending on the given requirements, i.e. there exist various levels of opacity how an event processing network is distributed.

Also known

N/A

Motivation

This pattern addresses the same problem context as pattern P7, i.e. event based systems that are supposed to process a large amount of data should be able to scale up as well as scale out if required. Therefore, facilities are required to address this requirement.

Applicability

Depending on the given infrastructure, scalability requirements or pre-defined overall system architecture, it can be necessary to distribute an event processing network in different ways. Thus, components of an event processing network need to be distributed *piecewise*, i.e. for instance that event processing networks are involved in a heterogeneous larger event processing network, while at the same time the event processing component should scale or provide functionality that requires the instantiation of an internal event processing network.

Structure

There exists three versions of this pattern. First, implicitly distributed event processing networks are defined within a central component and the scaling and distribution is done under the hood. This means the definition of the network is decoupled from its scaling. Second, explicitly This means each event processing agent is available to the rest of the world. The If, for instance, event-driven text stream processing should be integrated into an existing, heterogeneous event processing network, where different event processing technologies already interact, it could be required to encapsulate the whole

event processing network that is concerned with text stream processing and let this event processing network communicate with the *rest of the world* only via its output and input events. This means the inner life of the event processing network is not available to the rest. This also means that the isolated EPN must scale and provide other quality-of-service levels by itself.

Consequences

If implicit and explicit distribution are mixed, knowledge about more frameworks is required. This also complicates implementation, tuning and maintenance. But on the other hand an accurately fitting event processing solution can be built, as different event processing framework offer different advantages that can then be combined.

Implementation

Implicit distribution must be supported by the event processing framework that is used. In a first step, the event processing network is defined and then submitted to a cluster that distributes the work load of the event processing network. For instance using Apache Storm the submission of an event processing network or *topology* as called in that context in shown in listing 5.1.

Listing 5.1: Apache Storm implicit distribution example

```
Config conf = new Config();
conf.setNumWorkers(20);
conf.setMaxSpoutPending(5000);
StormSubmitter.submitTopology("mytopology", conf, topology);
```

Explicit distribution in contrast means that each part of the event processing network are distributed *piecewise* of the cluster of computers. For instance, Esper's event processing engine cannot be distributed internally, but runs rather within each computing node. A mixture of both can be implemented, if the parts are distributed piecewise and explicitly, but the scaling to address work loads is distributed internally.

Related patterns

P7 - Hybrid communication

5.2.2.4 Quality attributes

The final aspect of the reference architecture is a discussion of quality attributes that are especially relevant in the context of text processing and information filtering. In the context of this dissertation, two discussed attributes are especially important, namely scalability and performance, as one of the goals of this dissertation is to provide the text processing and information filtering capabilities instantaneously and under various document volume scenarios. To fulfil this goal, both aspects are crucial. There are several more quality attributes (cf. (Bass et al., 2012, p. 63)), but these are only briefly addressed.

Scalability

In general, there are two kinds of scalability: *horizontal* and *vertical* scalability (Bass et al., 2012, p. 186). Horizontal scaling refers to *scaling out*, this means for instance to add another server to a cluster, while vertical scaling refers to *scaling up* which means to enlarge the resource of a single processing unit, e.g. putting more memory into a server (p. 186). In order to support both types of scalability, the event processing execution units like event processing agents must support hybrid communication as described in 6P. Inter–EPN communication is done by providing input and output adapters that can communicate via a messaging mechanism. Thus new event processing building blocks like source or agents that reside on different physical machines can be added to the event processing network. Intra-node communication can be achieved by shared-memory approaches or by inter-thread communication. Scaling using this approach allows adding new components to the event processing agent on the same physical machine while facilitating high throughput. In general, it is preferable to avoid inter-node communication, because network communication introduces unforeseen latencies that can slow down the processing rate.

Performance

(Bass et al., 2012, p. 131) defines performance as "...the software system's ability to meet timing requirements" and is an quality attribute that is closely linked to scalability. The ability to increase the computing power, "while still performing well" (Bass et al., 2012, p. 131) is crucial for modern software systems and foremost for event-driven systems that need to process large volumes of data.

A central aspect of performance is *concurrency*, i.e. the concurrent processing of data by various execution threads. Concurrency is a complicated topic and often introduces problems like *race conditions*. In order to minimize this problem, P1 and P4 can be used. By defining a concise context (P1) the operations of an EPA are limited to a minimum semantic scope that decreases the requirements for EPA-internal parallel processing. This pattern affects

the internal design of the EPA. The Dynamic Filter EPA Pattern (P4) allows adding more event processing agents dynamically if the number of topics and search queries increases. Furthermore, these EPAs implement P6 and can thereby communicate using different approaches.

By its design the reference architecture supports scalability, as new event processing agents can be added dynamically to the event processing network. This is possible because each agent participating in the network knows to communicate with other agents due to the reference model. The pub/sub model that is implemented by each agent or sink also guarantees independent processing, because the agents only need to know which event types are available in order to subscribe to them. Critical aspects of this architecture are the centralized components that need to be accessed in order to perform a filtering task. This is necessary for the reasons mentioned above (complete term vector of a document, complete feedback information for a search profile).

The second aspect to consider is to control the amount of resources an EPA requires. Again, by using a concise context the computational requirements are by design minimized[55].
A third aspect is the dependency on other computation (Bass et al., 2012, p. 135). This comprises the need to wait for the synchronization with the results from other components. In general such dependencies should be avoided, in order to prevent locks or waits for the components. But as more and more in-memory computation is feasible that allows for very fast access to centralized data, it is a central design decision to go for the shared state pattern or the distributed state pattern.

Additionaly quality attributes

The additional quality of service aspects like reliability, availability or security can be addressed by different technologies that have to be chosen when a concrete software architecture is designed. In the context of this dissertation, these aspects were not addressed, because they did not matter. They are foremost important in the context of productive systems, where a violation or non-compliance to such non-functional requirements has serious business impacts. Nevertheless, some advice can be given.

In terms of reliability, the choice of the communication facilitator is important. If the communication pattern is based around fire-and-forget the throughput is probably higher, as such communication requires less overhead. But in case of failure, it must be considered that events are lost. To address this, approaches like *event sourcing* provide means to restart and replay certain event sequences in case a consumer or communication broker fail.

[55]This does not imply that every EPA requires only few resources, but the concise context assures that the EPA is designed to use the least resource as required.

Availability can be addressed by resource managers and environment monitoring tools. Event processing based components like sources or event processing agents can also emit quality-of-service events like pings or memory threshold warnings using Control Events. These events can be used by a resource manager to restart failed components or to start additional components to balance the processing load.

Security

Security is an aspect that must be considered from two points of view. The first is, the security outside of the event processing network and refers to the raw event sources. Here the source's security can also be protected already when events are being captured by the source – this means physical access limitations prevent sensors from capturing certain events – or when the raw sensors try to transport the raw events to the event sources. The second point of view is the aspect of security within the event processing network. Here security depends on the purpose of the event processing network. In scenarios, where the event processing network is supposed to provide public event clouds or event streams, a regular access limitation based on authorisation and authentication concepts is required. This means that before a new consumer can consume events from the event processing network it has to verify the identity of the person or machine that tries to access the data (authentication) and that this person or machine owns the required privileges (authorisation). After this has been done, the event stream can be transmitted over secured channels. But it is important to take potential effects on throughput into account due to encryption and decryption.

5.2.3 Architectural evaluation

As pointed out in section 5.1.4 a detailed evaluation of a software architecture using for instance ATAM can take up to several weeks involving multiple stakeholders and domain experts. Such an approach was not applicable in this context. Therefore, a lightweight approach was chosen and a checklist is provided that recapitulates the properties and goals of reference architectures as introduced in section 5.1.4 and 5.1.2.

In summary, most of the aspects were addressed by the reference architecture. Of course, an evaluation using several real-world implementations based on the reference architecture would be beneficial, but at the moment there are none available except the *StreamFI* system used in this dissertation. Nevertheless, by the introduction of a functional view, a process view, a set of basic patterns as well as discussion of the most relevant quality aspects, the author thinks this reference architecture is a valid proposal to address the problems in event based text processing and information filtering.

Criteria	Justification	Fulfilled
Common information	Cf. section 5.2.1.2	Yes
Stakeholder description	Cf. section 5.2.1.3	Yes
View point definition	Cf. section 5.2	Yes
Used view points	Cf. section 5.2.2	Yes
Known inconsistencies	Not known nor addressed.	No
Abstract	The level of details varied. The process view and patterns already provide concrete design advice and are hence not abstract. The functional view is high-level and therefore abstract. Furthermore, the description of the quality attributes only provided abstract advice on how to address the issues.	Partially
Generic	The reference architecture is generic for the scope of information filtering and text processing, but it is subject to further research if the patterns and process views can be transferred to other related tasks like summarisation or plagiarism detection[56]	Partially
Technology agnostic	Yes, the description does not required a specific technology.	Yes
Instructive	The proposed process view and patterns are qualified to support the stakeholders during the design and implementation phases. This was verified by the implementation *StreamFI*.	Yes
Informative	The proposed process view and patterns are qualified to support the stakeholders during the design and implementation phases. This was verified by the implementation *StreamFI*.	Yes
Regulative	No, it is not an officially agreed or accepted reference architecture. It is only a proposal.	No

Table 5.6: Evaluation checklist of the proposed reference architecture.

5.3 Summary

This chapter proposed a reference architecture for event-driven information filtering and text stream processing systems. First, a definition and scope of reference architecture was given by discussing different examples of reference architectures. Based on relevant research literature in the field of reference architecture design a three-phase design process was introduced along a set of process aspects, like stakeholders, viewpoints or design rationales. After the structural outline was defined, the concrete reference architecture was developed. In the first phase, the architectural analysis phase, stakeholders were defined and the scope of the architecture in form of a functional view, a process view, a set of patterns and a discussion of relevant quality attributes was set. In the second phase, the architectural synthesis phase, this scope was elaborated. The functional view introduced a set of domain tasks, which were based on the reference processes from Chapter 4, and showed their semantic relationship by arranging them into different event processing tiers that were based on the *EPTS* reference architecture. Each tier was then discussed in detail, whereby the tasks from the text processing domain received a logical grouping, which is intended to facilitate the design of event-driven components executing these tasks. Then, a process view was introduced that studied aspects of the essential building blocks of event processing components, i.e. the flow of information between components was described. Based on the experience from the implementation of *StreamFI*, the prototype software system used in the next chapters and the design of the functional as well as of the process view, a set of basic event processing architecture patterns was devised. These patterns form a basic set of architectural guidelines for several aspects relevant to the design of event-driven text stream processing systems, and can be seen as the baseline for an event-driven text stream processing pattern library. Finally, relevant quality attributes were discussed which should be considered during the design of a concrete architecture. Finally, the evaluation of the reference architecture was based on a lightweight approach that included a checklist and the reference to the successful implementation of the research system *StreamFI*. In the next chapter the concrete implementation of the prototype system is discussed.

Part III

Implementation and evaluation

Chapter 6

StreamFI – Implementation of an event-driven information filtering and text stream analysis system

In this chapter the implementation of *StreamFI* and its software engineering aspects are described. The goal is to show how the presented event reference model, the reference architecture and the architecture patterns can be used to instantiate a real-world software system. Also this chapter addresses several aspects from the used Design Science approach. First, it addresses the aspect of *Design as an artefact*, because the implementation represents a viable and instantiated artefact. Second, it addresses the aspect of *Design evaluation*, because the implementation allows to evaluate the proposed concepts like the event reference model and foremost the reference architecture. Third, it addresses the aspect of *Design as a search process*. The implementation was iteratively refined, new findings were incorporated and the software was continuously adapted and improved.

This chapter is structure as follows. First, the used technologies are discussed. Then, the used event types are introduced as well as an overview of the applied process and architecture pattern is given. Finally, the implementation itself is briefly described[1].

[1]The source code can be found a Bitbucket repository. `https://bitbucket.org/streamfi/streamfi`, user name*streamfi* and the password *BX0lzV7ZLcEi*.

Technology	Purpose
Java JDK 8	Implementation language
Esper 5.2.0	Event processing engine
Redis	Context store
CouchDB	Document store
JSAT 0.0.2	Machine learning framework
Apache Lucene 4.10.0	Search engine
ØMQ	Event channel

Table 6.1: Used technologies for the implementation of *StreamFI*

6.1 Choice of technologies

The implementation that is described in this chapter represents only one possible solution using the proposed reference architecture and model. This is in fact logical as the reference architecture and reference model were designed to be technology agnostic and thus each technology stack that allows to process events can be used.

StreamFI was implemented using the following technologies:

The event processing components were build using *Apache Maven*[2]. The dependencies and packages were managed using *Sonar Nexus*[3] and the application was built using *Jenkins*[4]. The lifecycle followed a continuous development cycle, i.e. the software was incrementally improved, adjusted and deployed.

Alternatives

In the course of this dissertation many new, interesting technologies arose. Foremost, in the field of event processing a multitude of frameworks were created, which – by the mid of 2015 – already started dominating the *Big Data* community. For instance, *Apache Storm*[5], *Apache Spark Streaming*[6], *Apache Samza*[7] or *Apache Flink*[8] are amongst the most popular projects when it comes to stream processing. Thus, it would be interesting to implement the proposed reference architecture and reference processes with these framework, in order to assess the applicability of the proposed artefacts and to compare

[2] `https://maven.apache.org`
[3] `http://www.sonatype.org/nexus/`
[4] `https://jenkins-ci.org/`
[5] `https://storm.apache.org`
[6] `https://spark.apache.org/streaming`
[7] `https://samza.apache.org`
[8] `https://flink.apache.org`

the computational performance.

Finally, a note on the used hardware. The runs were executed on a Octa core AMD FX-8320E Eight-Core (-MCP-) with 16MB, 16 gigabyte of RAM and a 1 terabyte RAID-1 running Ubuntu 14.40.3 LTS (trusty).

6.2 Applied artefacts from the reference modelling section

As argued in the reference model section, a reference model allows to communicate components using an agreed vocabulary with clearly defined semantics. Therefore, the event types, which were needed to implement the required functionalities, were implemented using Java. It was implemented self-contained, i.e. it does not require any further dependencies and can be, thus, easily be integrated by different event processing components. The event model was packed into a single independent component that was made available to the various components via a repository management system.

In addition, several architectural aspects and patterns from the section 5.2.2 were implemented, in order to verify their applicability. The various aspects are discussed when the different components are described in the next section.

6.3 Implementation

The implementation artefacts are described along the process steps for event-driven information filtering as presented in section 5.2.2.2. The various stages are described briefly in terms of the implemented package and the underlying structure. The complete class diagrams as well the source code are omitted here due to the volume of information that is not necessarily required here. For further details, the source code can be checked in the Bitbucket library that comes with this dissertation[9]. Also it must be noted that *StreamFI* represents a prototypical implementation, i.e. common non-functional requirements like optimum throughput, availability, fault-tolerance or restartability were not addressed. The system was implemented in two version. A local monolithic version to conduct the experiments and distributed one to verify the applicability of the event reference model and reference architecture in a distributed environment. Here the pattern *Degree of distribution* was

[9] `https://bitbucket.org/streamfi/`, user name *streamfi* and the password *BX0lzV7ZLcEi*.

applied as described in section 5.2.2.3. The monolithic version was used due to performance reason, as it was easier to tune. The distributed version is depicted in the next sections, in order to show how the described patterns can be applied. But in summary, the only difference between both implementations is that the single EPAs communicated via the same JVM in the monolithic version, while the distributed version uses ØMQ. This shows that the proposed patterns and architectural advice work in different scenarios.

6.3.1 EPA template

The EPAs were all built using the same, basic template. This was possible due to the event reference model and the proposed patterns, because both provide instructions how to model event processing components in a reusable way. Thereby the external *appearance* of the EPAs is the same and only the internal processing differs depending on the purpose of the EPA.

The starting point is the *StartEPA* class. This class instantiates the Spring *AnnotationConfigApplicationContext* that takes cares of the instantiation and injection of the required classes. This means it set ups the RedisContext, which serves as the control centre, and starts the ZeroMqsEventSink that listens for the events that come from other external EPAs (cf. hybrid communication pattern in 5.2.2.3). The event classes are defined on start up via a properties file. Also a TimerEventSink is set up that listens for CurrentTimeEvents, in order to synchronize the internal clocks. Sinks or EPAs register themselves at the ZeroMqEventSink with the event type they are interested. When a new event arrives and has been de-serialized, the ZeroMqEventSink calls the sink or EPA and the processing continues. The AbstractEsperBaseEPA represents the entry point to the used event processing engine Esper. It receives incoming events and emits them into the engine, which can then perform actions like pattern matching or sliding window calculation.

Figure 6.1 shows the class diagram of the EPA template.

6.3.2 Document pre-processing

The document pre-processing follows the steps as described in section 4.2.5.2. Instead of an external source a *DatabaseAdapter* is used to stream the documents from the database. Then *DocumentFactory* creates a *Document*, from which a *DocumentEvent* is created. The *ZeroMqEventSource* broadcasts the Document Events to the network. Figure 6.2 shows the class diagram.

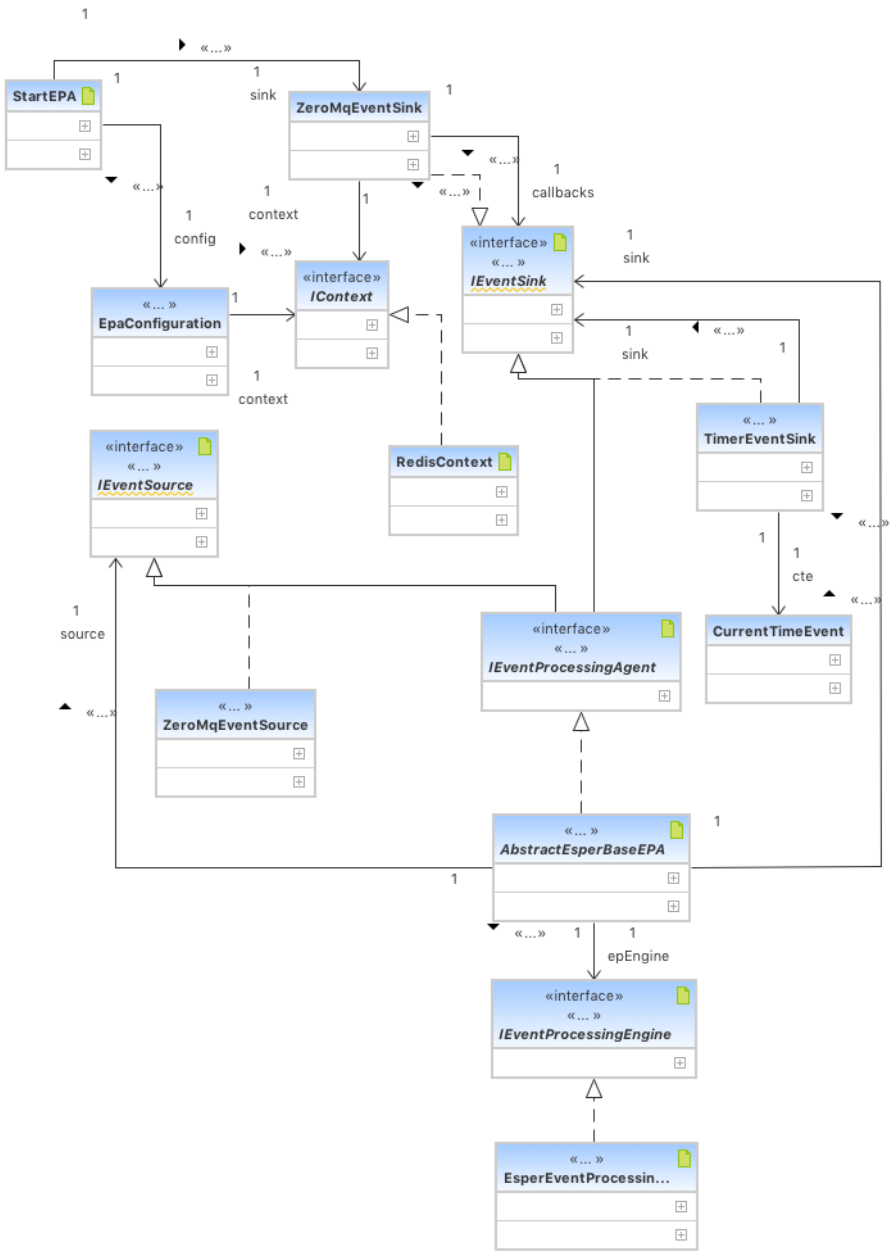

Figure 6.1: EPA template for *StreamFI*.

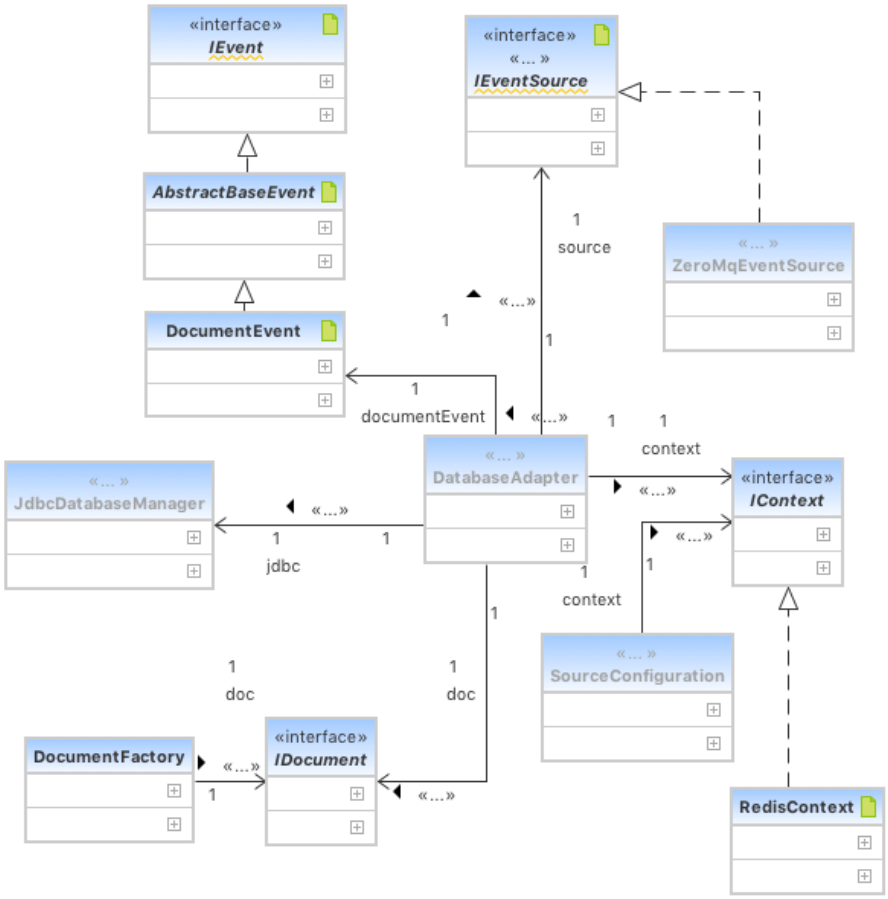

Figure 6.2: UML class diagram of event source for Document events.

6.3.3 Text event and meta event generation

Text events and meta events represent information about the structure of a document. As described in section 4.2.4.2, these event types are required to perform scoring and filtering tasks. Thus, they are essential for purposes of the next chapters. The EPAs that are in charge to generate the Text and Meta Events receive the required DocumentEvent instances via ØMQ. During startup EPAs register their IP address along the event classes they are interested in at the *Control Centre* as it was described in section 5.2.2.1. For the implementation, a simplified control centre was used called *IContext*. This interface provided basic functionalities to store and lookup documents, events, endpoints and search profiles For the concrete instantiation of the control centre *Redis* was used, as high-performance key-value store. Figure 6.3 shows the involved classes. All the events were generated by a single event processing agent for the purpose of this system. But due to the event reference model is easy to split the process over various EPAs.

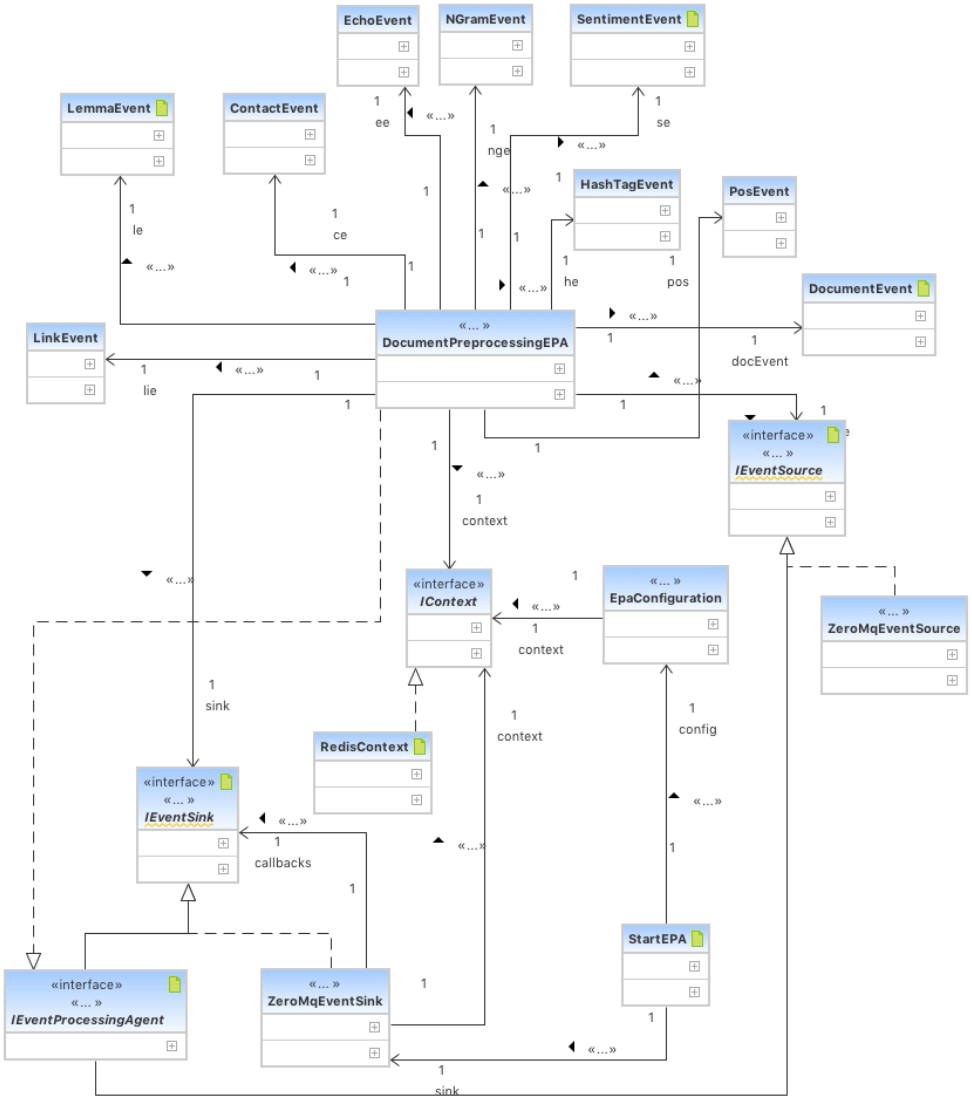

Figure 6.3: UML class diagram of document pre-processing and text event EPA.

It must be noted that the document pre-processing for the *StreamFI* happened up-front. This means the documents were analysed in term of document length, parts-of-speech, etc. and the results were saved to a *MySql* database. This was, mainly because the pre-processing lowered the throughput and evaluation runs took longer. As pre-processing is nothing that had to be repeated from scratch for an evaluation run, because it does not influence any performance measure, the information was streamed from the database, in order to save time during the evaluation. In general, all pre-processing steps except parts-of-speech tagging could be done in real-time even on the server, which was used for the experiments. Parts-of-speech required more computational effort due to the calculation of the *Hidden-Markov-Model*, on which the used parts-of-speech tagger was based. Nevertheless, for more current hardware based server this issue should be negligible.

6.3.4 Context filtering and scoring

The scoring and context filtering is the central component of the event-driven text stream processing system. As mentioned in section 5.2.2.3 about the *Concise Context EPA*, it is possible to have multiple functions within a single EPA as long as there is not a new event type involved that is not the final one that is emitted by the EPA. In this case, filter policy application and scoring take place within the same EPA, because it would be unnecessary overhead to use two EPAs for this purpose. Each topic from the corpora instantiates runs within a separate EPA, this means for an evaluation run of the TREC topics 36 scoring EPAs would be instantiated.

The scoring process consists of several steps. First, the document is matched against the search profile. This is retrieved from the shared stated context, in order to be able to get the latest updates from the relevance feedback EPA. After the matches, feedback matches, hashtag matches, etc. have been detected, the filter policies come into play. First, the TrecFilterPolicy is applied. This policy checks if the document lies within the pre-defined time range of the search profile. Then, one or more heuristic or automatically derived filter policies are applied[10]. The next step is to score the document using a scoring function that was also defined in the properties file. Important for scoring are the term and text stream statistics. These are kept individually in each EPA. The term statistics are calculated using Esper. This means the EPA does not only listen for new Document Events, but also for Text Stream Property Events. They are used to keep the text stream and term statistics up-to-date.

As the basic class structure was already shown in the previous two section,

[10]Filter policies are discussed in more detail in section 8.2.

here a sequence diagram is shown to illustrate the process. Figure 6.4 shows the according UML sequence diagram.

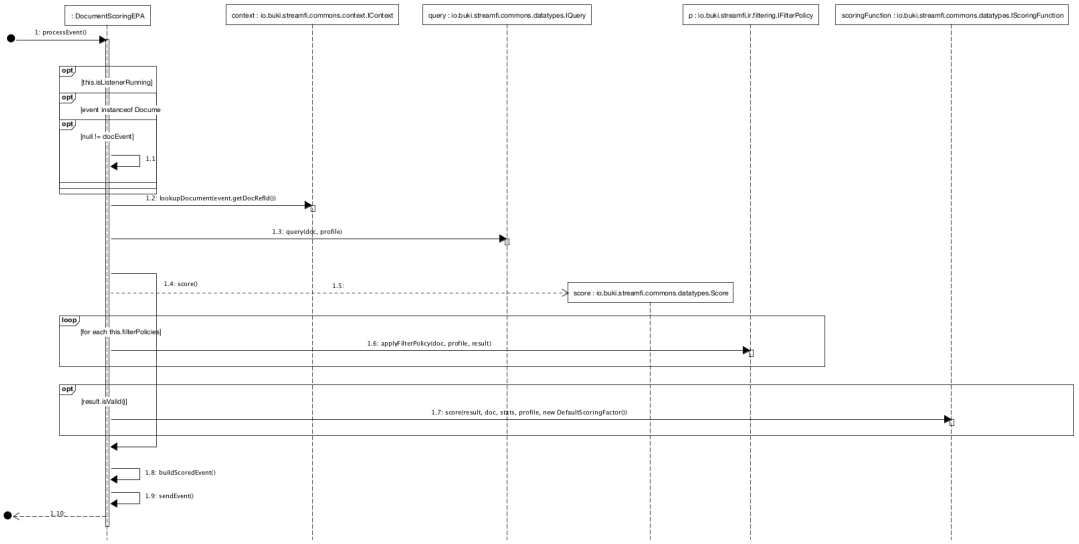

Figure 6.4: Sequence diagram of scoring process.

Figure 6.5 shows the UML class diagram for the scoring and context filtering EPA.

6.3.5 Classification and information filtering

After the score of the document has been calculated, ScoredEvent instances are emitted into the event processing network. The classification of the events is performed by a separate ClassificationEPA. This is because both created different event types and according to the concise EPA context pattern the tasks should be performed by different EPAs.

The classification and information filtering can be executed based on two approaches that are studied in more detail in section 8.4: score threshold based methods and machine learning based methods. The score threshold based methods use the score of the documents to discern relevant from irrelevant documents. As the used document scoring functions produce monotonically increasing score values the event processing engine can be used to monitor for instance the average score of all documents within a given sliding window and use this value as threshold.

The second approach is to train a classifier like *Support Vector Machines* or a *Naïve Bayes* and use the classifier to categorize documents into relevant and non-relevant documents. This approach is also studied in section 8.4.

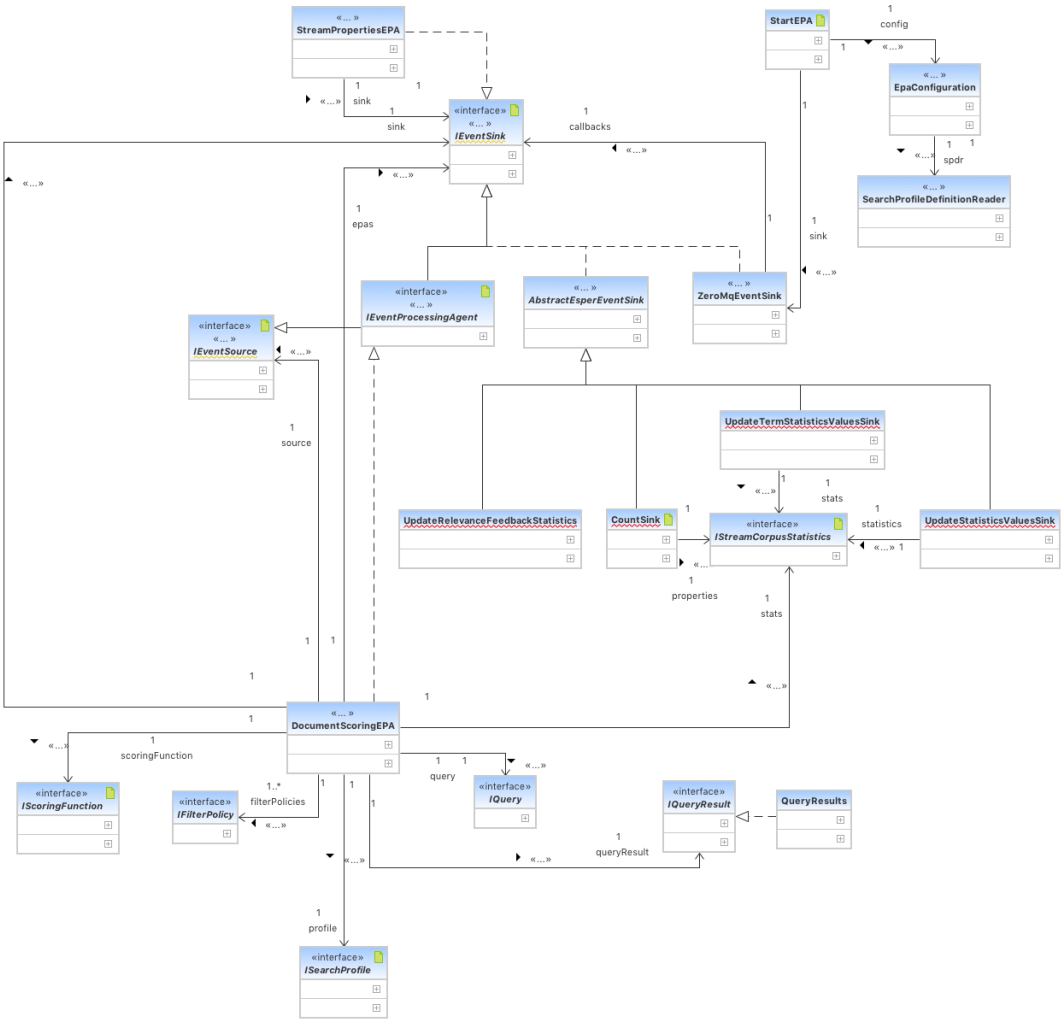

Figure 6.5: UML diagram of document scoring EPA

The main difference in implementing both approaches is that the values for the score threshold approaches can be monitored within the EPA, while the classifier model is trained and tuned in a separate module and then called by the EPA or continuously substituted[11]. The former is because event processing engines are easily capable of calculating stream statistics like average or medians for several distinct sub-stream Thus, the effort for externalizing this is not required. The latter is because machine learning models usually require manual tuning as well as feature engineering and this is better done offline. In this scenario the model was also trained and evaluated within the event processing agent, as the amount of available training items was only a few thousand and could therefore be integrated. For larger training sets this should be externalized, as mentioned.

The UML diagram correspond in most of its parts to the previously shown ones and are thus omitted.

6.3.6 Relevance feedback and model adjustment

Relevance feedback can be provided in two ways: from a user and from the system itself (pseudo relevance feedback). For the evaluation of the query expansion and relevance feedback component, the relevance assessments where streamed from the database. This means a source emitted RelevanceFeedbackEvent that were processed by a RelevanceFeedbackEPA. This EPA joined the relevance feedback information with ClassifiedEvent instances, in order to adjust the relevance feedback corpus, which contained the relevance feedback and query expansion terms of each search profile.

The UML diagram correspond in most of its parts to the EPA template and are thus omitted.

6.3.7 Logging and event sinks

The logging and event sinks are simple. The events sinks registered themselves as interested in ClassifiedEvent instances as well as StreamPropertyEvent instances at the ZeroMqEventSink. When a new event arrives, the processing starts. This is not different to process described in the EPA template section. Therefore, no further diagrams are required here.

[11] This could be done for instance by wrapping a fresh trained model in a TrainedModelEvent, for instance, that contains the model described for instance in *Predictive Model Markup Language*.

It should only be noted that the sinks form the exit point from the event processing network, i.e. from here events for e.g. visualizing the data stream can be sent to a GUI. In a proof-of-concept Stream Property Events were collected in a special sink that forwarded them to a web application via WebSockets in real-time and displayed the values as line charts.

The UML diagram correspond in most of its parts to the EPA template and are thus omitted.

6.4 Summary

In this section the basic aspects of the implementation of the prototype system *StreamFI* has been described. First, the used technologies were introduced as well as a brief list of alternative stacks was discussed. Then, the components of the implementation were described along the process view of information filtering that was introduced in section 5.2.2.2. In summary, the chapter showed that the proposed event reference model as well as the reference architecture are useful tools for the concrete implementation of event-driven text stream processing systems and that the implemented system can be used as foundation for the experiments and evaluations in the next chapters.

Chapter 7

Event driven analysis of dynamic stream corpora

The previous chapter presented the software engineering aspects of the proposed event based filtering system *StreamFI*. It demonstrated how the event reference model and reference architecture from chapters 4 and 5 were used to implement the system and how both artefacts supported the process.

Up to this point, the following aspects of the Design Science approach according to (Hevner et al., 2004, p. 83) have been addressed:

- Design as an artefact: A reference model and architecture have been designed and a real-world instantiation has been introduced.
- Problem relevance: The problems that are addressed with this dissertation have been discussed in Chapter 1 and the case for the approach has been made.
- Research contributions have also been outlined and justified for the reference model, architecture and implementation in the previous chapters.
- Design as a search process has been also addressed by the previous two chapters, where the implementation and the reference architecture iteratively influenced and improved each other.

Now this and the next chapter address two other guidelines. The guideline *Research Rigor* asks for "rigorous methods in both the construction and evaluation of the design artifact[sic]" (Hevner et al., 2004, p. 83). This guideline is addressed by application of well-established statistical methods as well as methods from exploratory data analysis to investigate the text stream properties of three social media corpora. The guideline *Design Evaluation* requires

that "[t]he utility, quality, and efficacy of a design artifact[sic] must be rigorously demonstrated via well-executed evaluation methods" (Hevner et al., 2004, p. 83). This is addressed by the successful application of the proposed research system on various text streams and the utilization of the collected results for the investigation of dynamic corpora structure.

In summary, this chapter addresses the following tasks:

- Investigate the structural similarity of dynamic text corpora based on sliding windows of varying sizes.
- Determine empirically a reasonable lower bound for sliding windows.
- Study methods for automatic sliding window size determination.
- Show the real-world applicability of the proposed event reference model by analysing three social media corpora using dynamic text corpora.

Chapter structure

As pointed out earlier, it is reasonable to minimize the amount of data that is necessary to execute a text stream analysis, in order to process large volumes of data efficiently and effectively. Although modern computing power is about to diminish the problem, focussing on the relevant data while leaving irrelevant data out is still important, because it helps to make results traceable, graspable and understandable for human beings. A well-defined and observable context is therefore important, and an adequately sized corpus is helpful here. *Dynamic corpora* are corpora that are based on sliding windows, i.e. they only contain a certain number of documents and "forget" old ones while new ones are added. Therefore, they are an efficient data structure in a high-volume text streams setting. But, before they can be used as a basis for information filtering, this thesis must investigate whether dynamic corpora contain enough information for such an area of application, and what are the lower size bounds that must be met.

In order to address these questions, the following aspects were covered. First, a reasonable lower bound of window sizes for dynamic text corpora was determined by studying the structural similarity and the statistical properties of various text properties for different sliding windows. By applying methods from exploratory data analysis and statistical tests a suitable minimum size for a sliding window per social media corpus was derived.

Second, an automated way to determine a reasonable window size using *Index Entropy* was investigated. The goal was to avoid the overhead of the

manual evaluation of different window sizes and to derive a suitable window size automatically.

Finally, three social media corpora were streamed into *StreamFI*, the values for various text stream properties were logged and finally analysed using *R*[1]. The goal was to show that properties can be studied based on the designed system and the determined sliding window size text stream, which can gather interesting insights into the text stream. The corpus analysis focussed on textual properties and was not intended to be a social network analysis [2].Nevertheless, based on the timestamps and the assumed user group of the social media platforms from which the corpora (Twitter and *Facebook*) were collected, a series of analysis results were derived and a few assumptions were made about what the data could mean.

7.1 Related research

Before the results are presented a short overview of related research regarding sliding windows and dynamic text corpus analysis is given, in order to be able to better judge the results of this chapter.

An early and fundamental approach to information filtering with sliding windows was presented by (Widmer & Kubat, 1996). They studied online learning in data streams and described algorithms that were able to address concept drift as it is present in text streams. The goal was to derive an optimal size of windows containing examples for the learning algorithm. Nevertheless, online learning is relevant in the context of text streams, their goal was focussed on the machine learning algorithms itself, while the aspects studied here are concerned with a preliminary stage, the feature stage. This means the sliding window in this context is concerned how to efficiently calculate features that can be used for scoring, thresholds, stream analysis or feature modelling for machine learning algorithms.

(Bifet & Gavaldà, 2007) presented a method that continuously monitored two differently sized windows for a statistical significant change of their underlying distributions. If a change was detected that sliding window size was shrunk until no more difference was detected. In (Bifet et al., 2011) this

[1]In a production ready system the stream statistics would be fed into a real-time dashboard that would allow conducting the same analysis as with R. But the implementation of a analysis interface was not in the scope of this dissertation

[2]The only available context information for the corpora was the timestamp of the documents. No further information like gender, geolocation or age was available to carry out a detailed socio-linguistic analysis, which can be considered as a relevant use case for text streams.

change detector approach has also been applied to the Twitter stream, but they focussed on sentiment analysis using *Hoeffding trees* and did not investigate properties of text streams beyond *tf.idf*. Foremost, the *null hypothesis* for their technique is to state that there has not occurred a significant change in the average of the recent values, i.e. they keep the average constant. This is a desirable fact because for a constant stream the properties can be assumed to be constant as well, but they did not studied if this really has an effect on filtering performance or if this is only the most efficient way to define a sliding window.

In 2012, a statistical comparison of sliding windows, weighting schemes and dynamic microblog corpora has been presented by (Kleisarchaki, 2012). They used *Hotteling's t-squared distribution* to study the statistical significant changes in the centroid of text clusters for given topics. The goal of this work has been to show how various sliding window sizes (count and time based) as well as different weighting schemes reflect change and evolution of cluster centroids of an information need. The results presented in this chapter are different. The major difference to their approach is that the complete corpus including various semantic concepts like hashtags or links are studied not only the *top-k* terms of a certain topic. Furthermore they only used a limited (3) amount of non-overlapping windows. Here a continuous series of several thousand sliding windows is used, i.e. the approach presented here is more exhaustive and focussed on more details in terms of text stream properties. Also, the approach in this dissertation is also based on aspects of continuous text stream processing and incorporates ideas from event processing. The latter introduces a software engineering aspect that was not covered in (Kleisarchaki, 2012).

In terms of text stream analysis, in (S. Kumar et al., 2013) a study of Twitter lexical data was presented. But they showed merely a general approach on how to analyse Twitter data and did not present specific results for Twitter corpora. (Hong et al., 2013) presented a similar approach. They also focussed on microblog text streams and also used adaptive sliding windows, but the main difference is that they did not leverage event based concepts as presented in this dissertation. Furthermore, they used an incremental corpus and not dynamic sub-corpora like in this thesis. They also restricted the application of sliding windows to their relevance feedback algorithm. Also they provided no details on how to efficiently derive thresholds for the given text stream filtering scenario. Finally, they used external information by resolving embedded links and incorporating this information in their filtering process. While this is a reasonable approach to overcome the character limit of Twitter or the short length of Facebook posts, it is apparent that the reso-

lution of links, their parsing and incorporation into the results in undeniable performance impacts.

Also similar is the work presented in (Mouratidis & Pang, 2011). They used sliding time windows to studied the effect on an *incremental threshold algorithm* that is used for document classification coming from a text stream. But they neither investigated an optimal size of the sliding windows to be used, nor if it is reasonable to use an incremental corpus or dynamic ones. These open research questions are addressed in this chapter.

In (Reed et al., 2006) a term weighting scheme called *tf.icf* was proposed. The rationale of the *icf* component was not to update the corpus statistics when new data arrives, but to calculate static, *corpus* wide document frequency values for a sufficiently large sub corpus upfront and use those for term weighting. Their approach is similar as they study the use of smaller sub-corpora instead of the actual corpus. In contrast, the approach here focusses on much smaller and more dynamic corpora. Furthermore, the sub-corpora are not fixed but change over time. Finally, the use of adapting sliding windows as presented in section 7.6 goes beyond the pure empirical approach in (Reed et al., 2006). Nevertheless, their work supports the idea that small sub-corpora are sufficient for any preliminary text stream processing and provide accurate information to perform information filtering tasks on them.

In (Klinkenberg & Renz, 1998) a heuristic based approach for adjusting the window size of batches was presented. They monitored changes in recall, precision and accuracy of the classifier and if the values differed more than α times the standard error of each value and concept drift detected and the windows were shrunk. (Klinkenberg & Joachims, 2000) used adaptive sliding windows to optimize the performance of a *support vector machine* based classifier. They trained the classifier on various sub windows and used the window size that performed best on the data. Both are valid approaches, but a rather static compared to dynamically adapting sliding window approach below that uses information directly from the text stream without the overhead of comparing various window sizes or using relevance feedback information.

7.2 Overview of used social media corpora

Three social media corpora were used in this dissertation.

- the *TREC Tweets 2011 Twitter* corpus (McCreadie et al., 2012)
- the *ICWSM 2011 Spinn3r* Facebook corpus (Burton et al., 2011)

	Tweets 2011	ICWSM 2011	Events 2012
Total no. documents	16,141,809	231,861,650	121,747,031
No. downloaded documents	12,154,880	34,161,850	60,658,567
No. used documents	4,537,142	16,299,730	44,517,904
No. available relevance feedback	50,324	N/A	152,950
No. topics	49	N/A	506

Table 7.1: Statistics about the three used social media corpora.

- the *Events 2012 Twitter* corpus (McMinn et al., 2013)

Table 7.1 summarises the statistics of the three corpora. The two Twitter corpora were created according to the Twitter guidelines for distributing Tweet information. This means only the id of the Tweet along with the user id was available and Twitter had to be crawled in order to download the Tweet's content. As Tweet ids are no unique lifelong valid identifiers, it was not guaranteed that all Tweet Ids in the corpora files also be crawled. That is why the number of total documents diverges from the number of available documents. The number of used documents shows how many of the available documents were used within the experiments. For the *ICWSM* as well as the *Trec Events 2012* only a subset of documents was used due to the respective total size of these two corpora.

7.3 Types of sliding windows

There are mainly four types of sliding windows that are used in data stream processing, *viz.*:

- *time based* sliding windows (rolling and batched[3])
- *length based* sliding windows (rolling and batched)
- *mixed time and length* based sliding windows (rolling and batched)
- *adaptive* sliding windows (both time and length based as well as rolling and batched)

[3] *Rolling* means that always the most recent n seconds or elements are kept. *Batched* means that when n seconds or elements have been recorded, they are discarded in a single batch and the windows starts again gathering data till the specified boundary is reached.

Time based windows

In this thesis time based and adaptive sliding windows are used. The various window types are used in different situations. *Time based* windows take all events into account that have happened within a specified time frame. Thus, they are well suited to reflect the temporal nature of a text stream, i.e. the sliding window reflects if few or many documents arrived. This aspect is a crucial aspect in text streams, because high-volume can indicate burst and special topics of interest. Also periods of low event volume are interesting as they can indicate the opposite, i.e. they can reflect if the interest in a topic has decreased. But also an overall measure of the event volume is an interesting indicator, e.g. when restricted to a certain geographic context, volume can help to make assumptions about the diurnal rhythm of the observed event producers. This kind of windows is mainly used throughout the course of dissertation.

Length based windows

Length based windows only take into account the recent n events disregarding when these have happened. This type of window is useful to derive stable values for instance for average calculations of data streams where events occur unevenly or sporadically. In information filtering such windows can be useful, for instance if the topics that should be filtered are *uncommon* and thus only few documents spread over a long period of time arrive. By using a length based window it is still possible to calculate statistical values. An example, *Google Scholar Alerts* sift through a vast amount of sources to send updates, if new publications concerning a defined topic have occurred. While there is probably a plethora of documents that need to be processed by *Google* to filter all new publications, probably few will satisfy the given filtering query of a users. For instance a match for the query *"Socio-linguistic studies of pre-latin languages between 1856-1901"* will happen very rarely. Therefore a *time based* windows would not work, because a reasonable size of even a few days would probably contain none or very few matches. This has consequences also on the design of *relevance feedback* mechanism or *term weighting* schemes, which will be studied in the next chapter, because it affects the design of the concerned component.

7.4 Overview of sliding windows used

Heuristic window sizes

The various window sizes that are studied are selected from an heuristic point view. This means the windows for the *TREC 2011* corpus were chosen to be between 30 and 7200 seconds, because of the event density available in the corpus. As described previously approximately 4 million *Tweets* were available for this corpus, which yields an event density of around 3.3 *Tweets* per second. Thus the smallest window size was chosen to be 30 seconds, in

order to obtain at least 100 events within the window per average. In contrast the window sizes for the ICWSM 2011 and the Events 2012 corpus were smaller, because of their much higher event density. Both corpora have an event density of around 60 documents per second. Thus the window sizes were chosen to be smaller and the smallest started at one second.

In summary, the used window sizes for the corpora were:

- TREC 2011: 30, 90, 180, 300 600, 1800, 3600, 7200 seconds
- ICWSM 2011 and Events 2012: 2, 5, 10, 20, 40, 80, 160 seconds

The proposed sliding windows were used a parameters for the standing queries that were used by *StreamFI*. The standing queries were implemented in *Esper EPL*. Listing 7.1 shows how such values can be calculated for various Text Stream Property Events. The parameter *stats_window* holds the value for the window size.

Listing 7.1: EPL - text stream property statistics

```
insert into stats_stream
select coalesce(avg(value),0) as averageValue
, coalesce(stddev(value),0) as stddevValue
, coalesce(count(*),0) as datapoints
, coalesce(sum(value),0) as sumValue
, coalesce(min(value),0) as minValue
, coalesce(max(value),0) as maxValue
, coalesce(median(value),0) as medianValue
, metaData.source as source
, type
, current_timestamp() as ts
from TextStreamPropertyEvent.win:time(stats_window sec)
group by type, metaData.source
output last every var_output sec
```

Furthermore, the terms *small* and *large* sliding window are not absolute and are not defined in a strict mathematical sense. As stated the window sizes were chosen from an heuristic point of view. They should illustrate the contrast of small, concise corpora, which are based on *short term* sliding window starting at only a few seconds, to the other extreme of an *incremental* corpus, which is based on a continuously growing sliding window whose size in theory could grow to ∞.

The problem with static sliding window sizes is to chose the proper size of the window. If the window is too large, it becomes insensitive to changes

The effect of the window size

in the distribution of the underlying text stream. Furthermore, such a window allocates more memory and processing power for maintenance purposes than required. Although memory and disk space become cheaper and cheaper, an obvious waste of resource is never acceptable. In contrast, if the window is too small, it is sensitive to even subtle changes. Therefore, an adaptive sliding window is desirable. In terms of information filtering in high-volume text streams there does not exists a method to determine the size of a sliding window adaptively and therefore two approaches are evaluated after the study of the structure of the dynamic corpora.

7.5 Studying the effect of sliding window sizes on the structure of dynamic text corpora

The objective of this section is to study the influence of various window sizes on the structural and statistical properties of text stream characteristics, i.e. how does a change in the size of a sliding window affect the distribution and time series progress of properties like average lemma count. The goal is to determine if small sliding windows are sufficiently similar to larger ones and thus can be used instead of the larger window. This would improve resource consumption and processing efficiency. Furthermore, there are numerous publications (e.g. (Mouratidis & Pang, 2009), (C. Chen et al., 2007), (Yogatama et al., 2014), (Shou et al., 2013)) that use sliding windows, but the size of the sliding window is often chosen arbitrarily. Therefore, this investigation can help researchers to justify their choice of the window size by referring to this dissertation.

Defining structural similarity

A reasonable lower bound is supposed to be found if the statistical and structural properties of the smallest chosen sliding window still resemble sufficiently large corpora or an incremental corpus, which holds all documents. Resemblance is defined based on the following criteria:

- A visual inspection of box plots, time series and density diagrams show strong similarities
- The central tendency of text stream properties values, e.g. document length, is the same from a statistical point of view
- The variance, i.e. the spread of the values is similar from a statistical point of view

If these criteria are met, a small sliding window can be used instead of a larger one.

Experimental setup and mathematical methods

For the structural study only time based sliding windows were used, because the focus is on the timely nature of the underlying text stream. Length based windows were used to stabilize statistics values for events that do not occur too often, e.g. the score of Relevance Feedback Events for a search profile, where only little feedback is available.

The focus of the analysis was on summary statistics about the text streams, i.e. various statistical moments of the text stream were gathered and analysed. The statistical values that were used to study the sliding windows were derived in real-time using *StreamFI*. The different Text Stream Property Events were fed into a Stream Statistics EPA that calculated the required Stream Statistics Events. The Statistics Events were emitted and consumed by a logging sink. Figure 7.7 outlines the process.

Instance-based and aggregation-based properties

The analysis used one *instance-based* and one *aggregation-based* property from the list presented in figure 4.11. Instance based properties are text stream properties whose absolute value of the statistical moments is not affected by the window size, but is determined by the number of occurrences within the document. E.g. if the maximum amount of characters in a message can be only 140 characters like with Twitter, then this value is not affected by the window size, i.e. regardless of the size of the sliding window this value is not affected. Nevertheless, the concrete value of a statistical moment like average of an instance-based property depends on the window size. E.g. if you have five documents with four lemmata per document, then the average is four, but if the window is larger and contains ten documents where the the first five have four lemmata and the last five have six lemmata, then average value changes to 5.5. The study of these value changes is the goal of this section.

Aggregation-based properties are in fact properties that are based on the *instance-based* ones, but are aggregated within the defined sliding window. These properties are used to reflect the overall occurrence of a property within a window, e.g. term frequencies, hashtag frequencies, distinct lemmata counts, etc. and their absolute value does depend on the size of the sliding window, e.g. if the window is small lemma *A* could occur five times, but if the window grows it can occur ten or twenty times or more.

In order to verify the research questions that were stated at the beginning of the chapter, the following points were investigated

- The time series progress of the properties for the three corpora was analysed.

- Statistical values like average, standard deviation and median were collected, reported and their structure was compared for several windows.
- The correlation of the time series progress were calculated and used as a measure to determine window similarity.

Statistical basics

The execution of these tasks requires the thorough application of statistical methods. Therefore, the following conventions are set, in order to achieve reproducible and sound scientific results.

1. The used sample size for each corpus was $n > 1.000$.
2. The p-value for statistical tests was set to 0.01. Values above this threshold are written in italic.
3. The similarity of the time series, which were generated for the two types of text stream properties, was assessed using *correlation* and *dynamic time warping*.
4. Time-series values were normalised or standardised – depending on the use case –, in order to scale the values to a comparable range.

The values that were used for the forthcoming analysis were standardised or normalised. Normalisation was used to bring values to a unit range of $[0, 1]$. For the time series visualizations the values were normalised, because a value between 0 and 1 removes a probably misleading meaning of magnitude that should not be expressed. The focus there was only on illustrating the visual progress of the time series. Equation (7.1) shows the used formula.

$$norm(d) = \frac{max(value) - d}{max(value) - min(value)} \tag{7.1}$$

In contrast, standardisation was used for the statistical calculations. Standardisation means to modify the values, so that the mean of the values is 0 and the standard deviation is 1. The values were standardised for the analysis using *dynamic time warping* (cf. below). The values for the calculation of *Pearson' product moment* was not standardised, as it uses the covariance of the standardised (z-score) values automatically (cf. (7.3)), and *Spearman's rank correlation* is based on ranks and thus does not required any normalisation or standardisation. The formula for standardisation is presented shown in equation (7.2).

$$z = \frac{x - \mu}{\sigma} \tag{7.2}$$

where:
μ is the average and
σ is the standard deviation.

Correlations

The examination of the histograms of the different sliding windows revealed that none of them follows normal distribution. Thus, the similarity of the different time series was examined by the correlation of their values. The calculation was based on *Pearson's product-moment correlation* and *Spearman's rank correlation*. Pearson's product moment is used, because it describes if there exists linear correlation of two variables. The scale is from -1 to 1, where 1 means a perfect positive correlation, 0 no correlation, -1 a perfect negative correlation (Wikipedia, 2015g). Spearman's rank correlation has the same scale and scale meaning and was used to identify a correlation if the values are not examined in their temporal but in the order of magnitude (Wikipedia, 2015j). Both are non-parametric tests, i.e. there is no assumption about underlying distribution required.

Formula for Pearson's product moment correlation:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{7.3}$$

where:
cov is the *covariance* of the z-scores
σ_X and σ_Y are the *standard deviations* of X and Y (Wikipedia, 2015g).

Formula for Spearman's rank correlation:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}. \tag{7.4}$$

where d_i is the rank difference between the two variables at rank i and n is the amount of raw scores (Wikipedia, 2015j).

Dynamic Time Warping

Besides the distributional similarity of the various time series, the temporal similarity of the time series was assessed using *dynamic time warping* (DTW). DTW is a well-known technique to assess the similarity of two sequences (Müller, 2007, p. 69). Initially, it was used to address problems in speech recognition. Utterances and samples were converted to a wave form and then compared, but speakers tend to pronounce words at different speeds to the sample. Although, both the sample and the utterance, are similar they do differ due to the temporal difference. Dynamic time warping addresses this problem by "warping" the sequences in a non-linear way, in order to produce a match (Müller, 2007, p. 69). In general, "[t]he rationale behind DTW is, given two time series, to stretch or compress them locally in order to make one resemble the other as much as possible. The distance between the two is computed, after stretching, by summing the distances of individual aligned elements" (Giorgino, 2009, p. 2).

But this approach is also applicable to any kind of value sequences and can be used for process monitoring, information retrieval or biometrics (Giorgino, 2009, p. 1). Also, it is considered as one of the most efficient similarity measures for time series (X. Wang et al., 2012) and was thus chosen to compare the time series progress of the time series of the different sliding windows. The results for the dynamic time warping based analysis are shown in a matrix where all windows are compared to each other. This measure reports *minimum global distance*. This is the optimal path that minimizes the so called *warping costs*, where warping describes the process of finding the best match between two sequences of values (Ratanamahatana & Keogh, 2004, p. 2). The warping costs are calculated by building a $n - by - m$ matrix, where i_th, j_th element in the matrix corresponds to the squared distance between two points q_i and c_j taken from the sequences of values Q and C (Ratanamahatana & Keogh, 2004, p. 2) describe then the optimal warping costs as

$$DTW(Q,C) = min\left(\sqrt{\sum_{k=1}^{K} w_k}\right) \tag{7.5}$$

"where w_k is the matrix element $(i,j)_k$ that also belongs to the k_th element of a warping path W, a contiguous set of matrix elements that represent a mapping between Q and C" (p. 2)[4]. In order to be able to judge the results and to get an impression of what the results mean, the DTW value of a random time series is also reported. This is because a random process should show a large difference to the other time series; thus the remaining distance values becomes more understandable. The values of the time series that were analysed using DTW were normalised before they were fed into the algorithm, in order to provide meaningful comparisons (Rakthanmanon et al., 2012, p. 263). This requirements were also put forth in a seminal paper by (Keogh & Kasetty, 2003, p. 362).

Online repository

All scripts can be found in the *Bitbucket* repository that was created for this dissertation. The link is `https://bitbucket.org/streamfi/streamfi`, the user name is *streamfi* and the password is *BX0lzV7ZLcEi*.

In the following sections, structural and statistical properties of the three previously introduced social media corpora are studied, in order to verify the claim that smaller sliding windows are sufficiently similar to larger ones and thus can be used instead of them.

[4]For further details on the dynamic time warping algorithm cf. (Berndt & Clifford, 1994)

Analysing the TREC 2011 corpus

The analysis starts with the TREC 2011 corpus, where two text stream properties are studied, in order to reveal any structural similarity or dissimilarity of the two properties within sliding windows of various sizes. The first property that is used is the *average lemma count*. This stream property reflects the average occurrence of a term within the given sliding window.

Figure 7.1 shows the time series progress of this property over the course of the complete Twitter corpus for all sliding windows but the smallest, 30 second window[5]. As previously mentioned, all values are normalised to a scale between *[0,1]*, in order to keep values comparable.

The first visual observation that can be made is that for the TREC 2011 corpus all windows have the exact same time series behaviour for the aggregation-based property *average lemma count*. The time series follows a regular overall circadian rhythm[6] expect a peak on 25 January 2011, where US president Barack Obama held a *State of the Union* address, which provided a major increase in volume of Tweets and thus raised the average. Nevertheless, this visualization can be taken as a first indicator that large windows and small windows resemble each other.

Correlation values in practice

Statistical evidence of this visual indicator is provided by the correlation of the various sliding windows. Table 7.2 shows the correlation values for the *average count per lemma* property using Pearson's product moment. Table 7.3 shows the values for Spearman's correlation. (Cohen, 1988, p. 284) provided an acknowledged rule of thumb on how to interpret the correlation values, and stated that correlation of $> .5$ is large, $> .25$ is considered as moderate and .1 as small, but still viable. Using this definition the correlation from the largest, 2 hour sized window to the 90 second window is still medium. Only the smallest, 30 second window shows a correlation around 0, which indicates that this small windows does not resemble the other windows at all. This observation is both valid for the Pearson's as well as the Spearman's correlation.

Figure 7.1 shows the time series using the absolute values. As the average lemma count is an aggregation-based property, it must be noted that the absolute value of the average count per lemma is consequently higher for large windows, because aggregation-based properties accumulate more values per window. Normalisation helps to eliminate this effect, and a look at the normalised time series in figure 7.2 underlines that the time series progress of

[5]This is only shown later, because this window fluctuates heavily and would spoil the visual presentation if used in every serial graph.

[6]Section 7.7 provides a more detailed description.

	7200 sec	3600 sec	1800 sec	600 sec	300 sec	90 sec	30 sec
7200 sec	1	0.839	0.650	0.508	0.453	0.368	-0.085
3600 sec	0.839	1	0.892	0.764	0.714	0.615	-0.061
1800 sec	0.650	0.892	1	0.920	0.864	0.744	-0.045
600 sec	0.508	0.764	0.920	1	0.964	0.838	-0.031
300 sec	0.453	0.714	0.864	0.964	1	0.872	-0.026
90 sec	0.368	0.615	0.744	0.838	0.872	1	-0.013
30 sec	-0.085	-0.061	-0.045	-0.031	-0.026	-0.013	1

Table 7.2: Correlation calculated using Pearson's product moment for average lemma count TREC 2011.

	7200 sec	3600 sec	1800 sec	600 sec	300 sec	90 sec	30 sec
7200 sec	1	0.824	0.689	0.587	0.544	0.447	-0.090
3600 sec	0.824	1	0.914	0.815	0.775	0.661	-0.062
1800 sec	0.689	0.914	1	0.916	0.873	0.746	-0.045
600 sec	0.587	0.815	0.916	1	0.940	0.791	-0.027
300 sec	0.544	0.775	0.873	0.940	1	0.824	-0.017
90 sec	0.447	0.661	0.746	0.791	0.824	1	0.001
30 sec	-0.090	-0.062	-0.045	-0.027	-0.017	0.001	1

Table 7.3: Correlation calculated using Spearman's rank corrleation for average lemma count TREC 2011.

the various windows are similar.

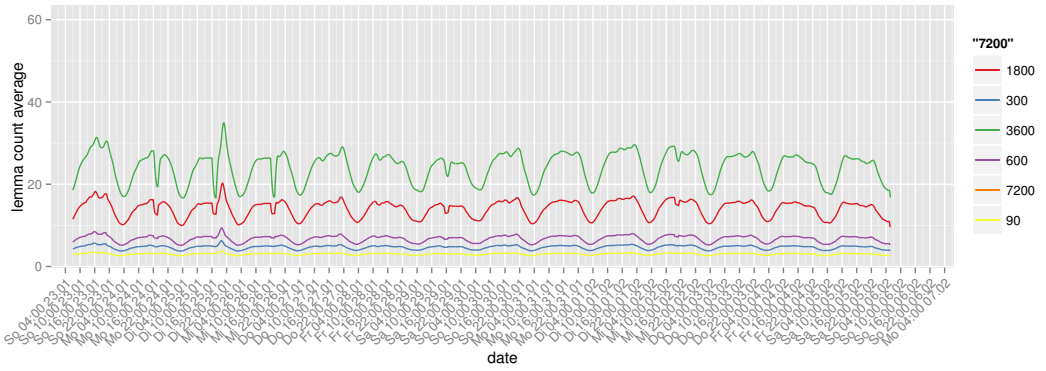

Figure 7.1: Time-series progress of *average count per lemma* for the TREC 2011 corpus.

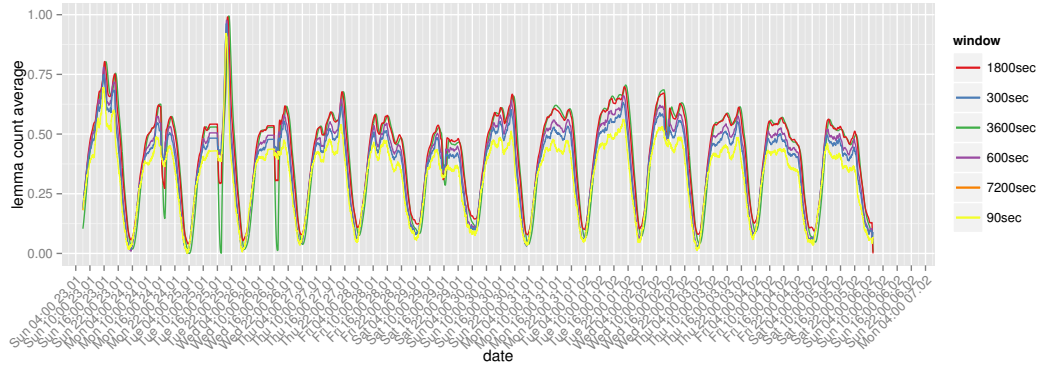

Figure 7.2: Normalised time series progress of *average count per lemma* for the TREC 2011 corpus.

Dynamic time warping similarity

Finally, the similarity of the various time series is reported using dynamic time-warping. As pointed out in section 7.5, dynamic time warping is an acknowledged way of determining the similarity of time series. Table 7.4 shows the optimal path costs for all windows calculated using dynamic time warp based on the standardised values of the average lemma counts. As dynamic time warping has a cost of $O(n^2)$ not all of the 23,000 log values were taken, but only a subset of 7,500, in order to keep the calculation time similar[7]. Nevertheless, a closer look underpins the previous results. The *random* window marks the upper bound of the distance within all comparison the matrix with values above 4,000 units, which is more than 2,000 units from the other values. This illustrates how different a random window is from the

[7] An analysis of the remaining values using the same 7,500 values window showed the same results. The results can be reproduced using the R scripts stored in the online repository.

	30 sec	90 sec	300 sec	600 sec	1800 sec	3600 sec	7200 sec	random
30 sec	0	3,269.8	3,445.3	3,473.7	3,542.0	3,601.4	3,605.5	4,206.3
90 sec	3,269.1	0	1,612.2	1,726.7	1,905.0	2,033.1	2,150.5	4,481.0
300 sec	3,445.6	1,612.4	0	493.7	732.5	828.3	952.2	4,774.7
600 sec	3,473.2	1,726.8	493.7	0	338.0	448.4	579.6	4,755.1
1800 sec	3,542.0	1,905.0	732.5	338.0	0	159.3	328.9	4,791.0
3600 sec	3,601.8	2,033.1	828.3	448.4	159.3	0	181.7	4,826.3
7200 sec	3,605.2	2,150.5	952.2	579.6	328.9	181.7	0	4,882.7
random	4,206.3	4,481.0	4,774.7	4,755.1	4,791.0	4,826.6	4,882.7	0

Table 7.4: Temporal similarity of average lemma count time series for TREC 2011 corpus calculated using dynamic time warping.

studied ones. The values for the windows, between 90 second to 7,200, are between 1,600 and 2,150 units, i.e. a very close range of only 550 units. In turn, the distance between the 30 second window and the next larger, 90 second window is more than 1,600 units, i.e. very far from the other windows. In summary, the random window and the smallest window are far distant to the time series progresses of the remaining windows and thus it seems reasonable to assume that small windows are similar to larger ones. This claim is now studied using another document feature and different corpora, in order to consolidate the argumentation.

Document length

The next property studied is the average document length. Again, the goal is to show that small windows are suitable to approximate the distribution and time series of larger ones. Document length is an instance-based property and the time series progress for windows of 90 to 7200 seconds is shown in figure 7.3. Again the time series progress is similar to the one of the average lemma count. The graphs n figure 7.3 are similar and match almost perfectly. Figure 7.4 shows the same data, but also includes the 30-seconds window, in order to illustrate how irregular the time series becomes if the size of the window is too small. While the windows from 90 to 7200 seconds are similar, the small, 30 seconds one oscillates heavily even though these values are smoothed by the same window as the values from the larger ones.

This behaviour is also supported by some statistical values. First, for this property additionally the density is plotted to examine the distribution of the values. In order to be able to compare the values they were standardised. Figure 7.5 shows the densities for all windows but the smallest. It is apparent

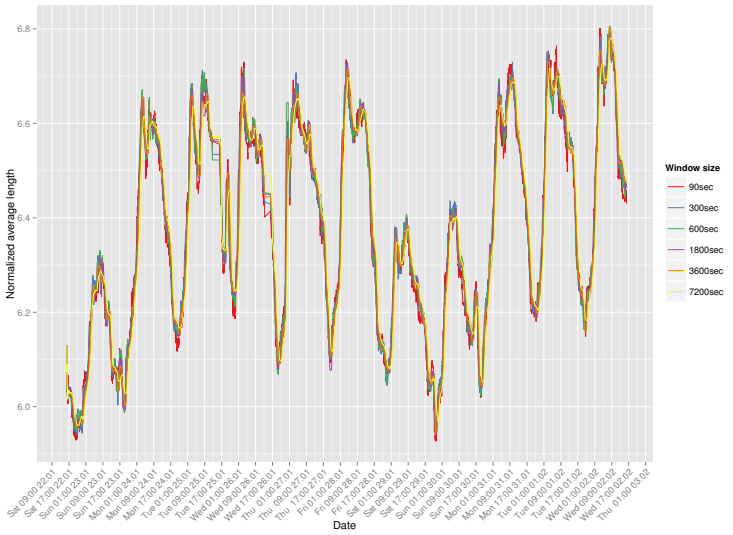

Figure 7.3: Time-series progress of *instance-based* property document length TREC 2011.

that they look similar and thus follow the same distribution. Figure 7.6 then shows the comparison of the density of the largest window as an example for the other similar distributions and the smallest window. The difference is obvious. The small window does not show the same bi-modal distribution as the other windows. Thus it can be assumed that a 30 second window does not reflect the text stream well. The correlation values in tables A.3 and A.4 also underline this assumption.

Tables A.3 and A.4 show the correlation values. Again, these values support the finding that a too small window does not well resemble larger ones.

The dynamic time warping values for the document length showed the same results as for the average lemma count. Table A.5 summarises the results.

The same analysis was conducted for the other two corpora using the same properties. The results were the same as shown in this chapter and details can be found in A.

Summary

This section has studied the structural properties of dynamic corpora. Two text stream properties were studied in terms of their temporal as well distributional similarity, in order to verify the claim that small dynamic corpora approximate well larger dynamic corpora and can thus be used as a memory saving and processing efficient alternative to large sliding windows. Visual as well as statistical evaluations showed that this claim is true by comparing the structural properties of the chosen text stream properties using correlation as well as time series similarity calculations. Based on the analysis, a lower bound for the TREC 2011 corpus was determined empirically to be around 90 seconds; the lower bound for the Events 2012 and the ICWSM 2011

	7200 sec	3600 sec	1800 sec	600 sec	300 sec	90 sec	30 sec
7200 sec	1	0.958	0.901	0.810	0.738	0.567	−0.027
3600 sec	0.958	1	0.960	0.864	0.787	0.603	−0.017
1800 sec	0.901	0.960	1	0.914	0.833	0.636	−0.011
600 sec	0.810	0.864	0.914	1	0.918	0.701	−0.019
300 sec	0.738	0.787	0.833	0.918	1	0.769	−0.027
90 sec	0.567	0.603	0.636	0.701	0.769	1	−0.022
30 sec	−0.027	−0.017	−0.011	−0.019	−0.027	−0.022	1

Table 7.5: Correlation calculated using Pearson's product moment for average document length TREC 2011.

	7200 sec	3600 sec	1800 sec	600 sec	300 sec	90 sec	30 sec
7200 sec	1	0.926	0.863	0.783	0.720	0.561	−0.019
3600 sec	0.926	1	0.941	0.837	0.765	0.592	−0.008
1800 sec	0.863	0.941	1	0.902	0.822	0.634	−0.003
600 sec	0.783	0.837	0.902	1	0.913	0.698	−0.017
300 sec	0.720	0.765	0.822	0.913	1	0.766	−0.028
90 sec	0.561	0.592	0.634	0.698	0.766	1	−0.021
30 sec	−0.019	−0.008	−0.003	−0.017	−0.028	−0.021	1

Table 7.6: Correlation calculated using Spearman's rank corrleation for average document length TREC 2011.

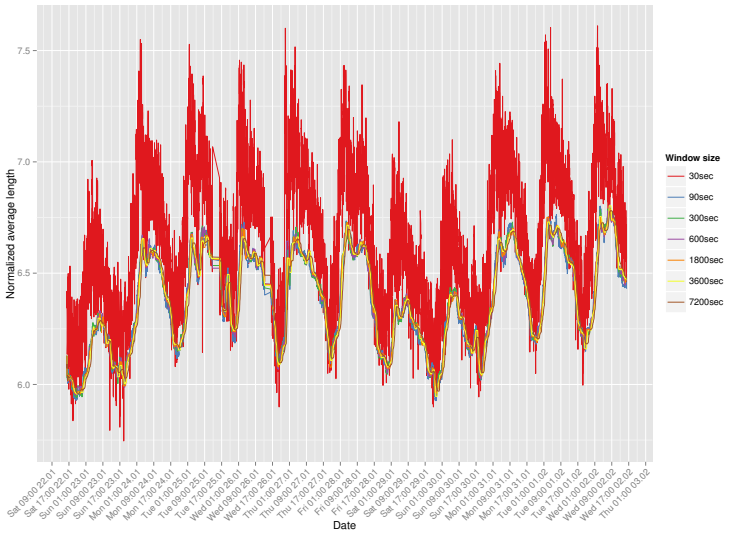

Figure 7.4: Time-series progress of instance-based property document length TREC 2011 incl. smallest window

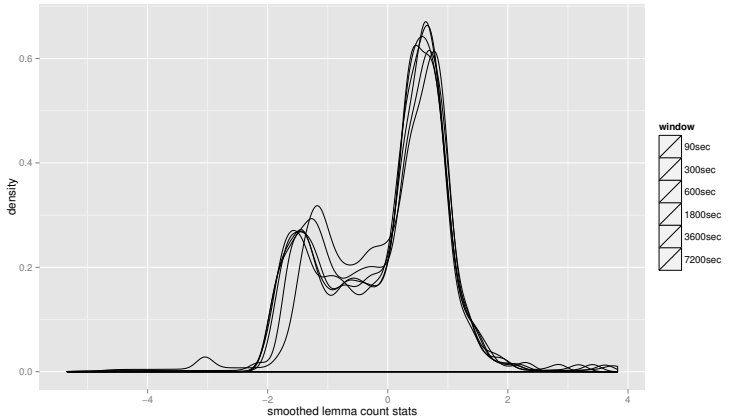

Figure 7.5: Density of normalised average count per lemma TREC 2011.

corpus was determined to be 5 seconds. Although it was shown that small and larger windows do not differ significantly and thus it would not matter which window size is chosen; it is still desirable to automatically determine a reasonable window size, unless the system's hardware configuration would allow for satisfactory performance either way. This would first avoid an up-front static analysis as the one presented here, and second it would guarantee the optimal use of resources. Therefore, the next section studies a way to automatically and adaptively determine the window size for dynamic corpora.

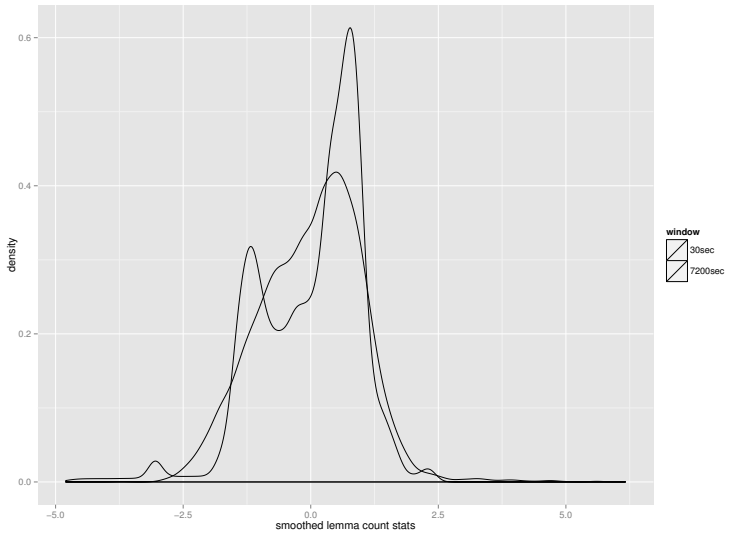

Figure 7.6: Density of normalised average count per lemma TREC 2011 - largest vs. smallest window.

7.6 Determining adaptive sliding window sizes

The previous section empirically demonstrated the effect of different sliding window sizes on stream statistics values of dynamic corpora. It was shown that small windows are very similar to larger ones in terms of their structural and statistical properites and thus can be considered as a valid substitute, thereby saving resources. But this was a *trial-and-error* approach studying a number of sliding windows, whose sizes were chosen arbitrarily before the analysis. An adaptive approach that automatically shrinks the sliding window to the smallest reasonable size while being sufficiently intelligent to grow the window when needed would relieve the developer from determining a window size upfront. Therefore, two methods are compared. First, the change in the index entropy is monitored using a change detector and second, the window is adjusted, so that it keeps only the number of documents in the window that are required to fulfil statistical boundaries for selecting sample sizes.

There are several approaches to dynamically adjust a sliding window (cf. (Bifet & Gavaldà, 2007), (Yang & Mao, 2013)) or (Klinkenberg & Renz, 1998) and they have in common to use change detectors to adjust the window size. This means they monitored term distributions, average accuracy or precision and if the values were above or below a predefined threshold, the window size was increased or decreased. A change detector is also used here, but the novelty of the method is to use index entropy as the underlying measure that is monitored. Formula (7.6) shows the common calculation approach as

	30 sec	90 sec	300 sec	600 sec	1800 sec	3600 sec	7200 sec	random
30 sec	0	3,269	3,445	3,473	3,542	3,601	3,605	4,196
90 sec	3,269	0	1,612	1,726	1,905	2,033	2,150	4,418
300 sec	3,445	1,612	0	493	732	828	952	4,716
600 sec	3,473	1,726	493	0	338	448	579	4,702
1800 sec	3,542	1,905	732	338	0	159	328	4,752
3600 sec	3,601	2,033	828	448	159	0	181	4,785
7200 sec	3,605	2,150	952	579	328	181	0	4,856
random	4,196	4,418	4,716	4,702	4,752	4,785	4,856	0

Table 7.7: Temporal similarity of average lemma count time series for TREC 2011 corpus calculated using dynamic time warping.

defined in (Shannon, 2001, p. 393).

$$H(X) = \sum_i P(x_i)\, I(x_i) = -\sum_i P(x_i) \log_b P(x_i) \tag{7.6}$$

Rationale

The rationale behind using index entropy is as follows. In the previous section it has been shown that the window size does not really affect the statistical and structural characteristics of text stream properties, unless the window is too small. So if the randomness decreases or increases within a defined range δ due to the varying window size, i.e. the entropy of the current subcorpus deviates only slightly from the previous one, then the randomness and thus the degree of contained information can be considered stable for the current context. The change detector together with an upper bound, which depends on the available memory, and a lower bound – this could be the resolution limit of the event processing engine – keep the window within valid bounds. In addition, the adjustment of the window size becomes independent of feedback information that is required if precision, recall or accuracy values are used and thus can rely only on text stream data.

Why adapt the size anyway?

One might argue that the window size could change randomly, as long as the window size stays within the defined bounds and thus there is no need to use index entropy and a change detector. Furthermore, modern data centres can provide computers with several terabyte of memory, which could make the problem of keeping the memory footprint small negligible. While these are valid arguments, an adaptive window is still useful for two reasons. First, as entropy is a well-defined measure in information theory, its value pro-

vides an understandable and traceable base for a dynamic sub-corpus. It is also valid to assume that by using the change detector, "enough" randomness is contained in the sub-corpus is contained, in order to allow for a reliable computation of scores and thresholds. Second, even though the costs of memory are decreasing continuously, more complex event processing agents could require several gigabytes of memory – depending on the variety of properties and the data volume – and if there were several hundred to several thousands of event processing agents, memory would become an issue again. And as mentioned earlier, even though in times of memory abundance, wasting memory without reason is still no choice. Neither from the cost perspective, nor from the infrastructure management view. Therefore an adaptive mechanism that is based on a sound theoretic foundation and a practicable implementation are real benefits.

The reason to use entropy as the function of choice is first that it is based on probabilities which are well manageable in a streaming scenario, second that a related concept *cross entropy* has been successfully used in the evaluation of the quality of language models, i.e. the entropy concept was already used in an information retrieval setting and third, as previously mentioned entropy is an acknowledge and accepted information theoretic measure. The following lemma summarises the idea:

Lemma 7.6.1. If the entropy (randomness) of a dynamic sub-corpus index I within a sliding time window t is stable, i.e. it differs only by predefined $delta$ from the previous sub-corpus, then the sub-corpus is considered as a valid representation of the overall stream and can be used for valid corpus related calculations.

Entropy can be also interpreted as the randomness of messages. If one considers a complete corpus as one single message by concatenating all the constituting documents, one can calculate the overall entropy for this corpus.

For detecting a change formula (7.7) was used, which represents a *Shewhart control chart* (cf. (National Institute of Standards and Technology (NIST), 2013)). The formula states that a change is detected if the average entropy deviated more than k standard deviation from the previous mean.

$$change(n) = \begin{cases} 0, & \text{if } n_1 \geq (n_0 - ksd(\dot{n}_0)) \wedge n_1 \leq (n_0 + ksd(\dot{n}_0)) \\ 1, & \text{else} \end{cases} \tag{7.7}$$

Sampling based adjustment

Another approach is based on sampling. As in a stream context documents are continuously added and removed, each sub-corpus can be considered as

a sample of overall stream seen up to the current moment. Therefore, statistical methods can be used to determine a sample size based on the required confidence interval, confidence level and the assumed population. According to Cochran (1975) (cited after (Israel, 1992, p 3.)) the sample size n_0 given the aforementioned properties can be calculated with formula (7.8)

$$n_0 = \frac{Z^2 pq}{e^2} \tag{7.8}$$

where:
Z is the confidence level defined as a z-score[8]
p and q are the proportion of the desired target value within the population. If this is not known the maximum variability for each class is used which defaults to .5, i.e. 50% for each class
e is the desired level of precision, i.e. the confidence interval.

For the experiments, a confidence level of 95% was chosen and the confidence level was set to $\pm 5\%$. Using the previous formula this yields:

$$n_0 = \frac{Z^2 pq}{e^2} = \frac{(1.96)^2 (0.5)(0.5)}{(0.05)^2} \approx 385 \tag{7.9}$$

Now that two approaches have been proposed, it must be evaluated how well both approaches perform in contrast to a static sliding window. Table 7.8 shows the results of all three approaches. The results were gathered by using *StreamFI*[9]. The speed improvements are only shown as percent values, because the absolute speed in events per second also is determined by configuration of the system as well as the used hardware. The figures should only highlight the potential of the usage of dynamic windows.

7.7 Studying temporal aspects of text stream properties

The Design Science approach is to find solutions to real-world problems. In this section, the case of analysing high-volume social media corpora is investigated by using the proposed event processing approach. Thereby it is

[8]The commonly used confidence levels are 90%, 95% and 99%. This corresponds to z values of 1,645, 1,96 and 2,575.

[9]The configuration was as follows. Okapi BM25 as term weight, average score as threshold, no query expansion and a filter policy that required two matching search terms

	Static: 90 second	Adaptive: index entropy	Adaptive : sampling
TP	575	399	420
FP	7662	5721	5443
TN	481	293	283
FN	10287	7057	6468
Macro precision	0.2724	0.2863	0.2777
Macro recall	0.2911	0.2118	0.2258
Macro F1 measure	0.2610	0.2447	0.2585
Speed improvements in %	0	19,36%	4,19%

Table 7.8: Comparison of filtering performance static vs. adaptive sliding windows for the TREC 2011 corpus.

shown that the proposed methods are viable tools for such cases. The approach is a temporal investigation of different text stream properties for a selection of days from the social media corpora. This is an exemplified analysis that shows which aspects can be studied and how. It is not intended to be an exhaustive study of the used corpora in terms of every available linguistic detail. This is not possible due to extent of such an analysis and foremost due to the lacking meta data which would be required for a detailed analysis.

Scope An analysis along various textual dimensions[10] can help to understand the underlying semantics of a text stream and help to spot interesting behaviour in social text streams. The graphs that will be shown in the next sections represent a post-hoc analysis of the corpus data using *R*[11]. Post-hoc means that values from *StreamFI* were logged to a *logging sink* and then analysed offline in *R*. But as the underlying *Statistics Event* arrive in real-time from the statistics event processing agent, these values can also easily be feed into a real-time dashboard for visualization. However, the implementation of such a dashboard was out of scope.

Goals The goal of this analysis is to show what kind of insights a visual analysis of the time series progress of different text stream properties can provide in terms of social media linguistics. (Zappavigna, 2012, p. 192f) called this "internet linguistics" and stressed the point that "given the episodic nature of social media as a form of streaming data, managing the time dimension is very important" (p. 192). Therefore, the event based approach to text stream

[10] These dimensions are referred to as *text stream properties* throughout the rest of the analysis.

[11] *R* is a widely used statistical computing software. `https://www.r-project.org/`

analysis can provide a real benefit.

Furthermore, this analysis should highlight the questions and assumptions that can be derived by such an analysis. A detailed statistical analysis or drill-down into specific, potentially interesting time series spots, is not provided. This could have only been done within a very limited scope; furthermore there were not enough meta data available to conduct a scientifically sound analysis. Instead, multiple text stream properties were studied, in order to demonstrate the potential use cases and analysis aspects in terms of "internet linguistics". The presented results are hypothesised based on the available information contained within the corpora (cf. next section).

7.7.1 Overview of text stream properties

The event reference model proposed in Chapter 4 allows studying a text stream in a variety of dimensions. In this section a set of properties will be introduced of which a few are studied in detail to show how *event based text stream processing* can be used in corpus analysis.

The following lists summarises properties that were available for the corpus analysis. This only represents the already defined properties of the proposed event reference model. But as shown earlier, the model can be extended as required, and for the current analysis the following properties were sufficient.

1. Document length; number of content words without stop words
2. Case counts: upper case or lower case
3. Number counts
4. Special character count (like ?, or &)
5. Character counts
6. N-Gram counts
7. Lexicon-based sentiment counts (positive, negative, netural)
8. Parts-of-speechs (nouns, verbs, etc.)
9. Link count
10. Hash tag count
11. Spelling errors
12. Named entities

Figure 7.7 shows a schematic view on the data gathering process .

General remarks

Before the analysis of the corpora is presented, some general statements about the corpora themselves are necessary, in order to better understand the data

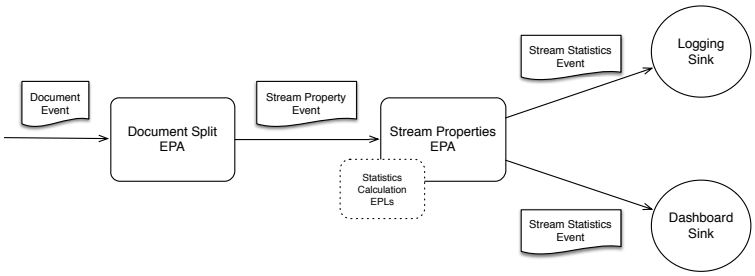

Figure 7.7: Stream statistics gathering process

and the graphs as well as support anybody interested in reproducing the results.

First, it is important to mention that the time series graphs are dependent of the chosen time zone. Social media platforms are usually globally available applications, which are used 24/7 throughout the world. Nevertheless, the popularity of social media sites vary from country to country and thus the usage statistics are not evenly spread over the day. In the case of Twitter, which accounts for two of the three corpora, more than 50 per cent of the users were from the USA (Beevolve, 2012) at the time (2011, 2012) the corpora were collected. Other sources state that approx. 25 percent of all actively tweeting users were from the US and approx. 6 percent were from the UK at that time, but as the corpora only contain English tweets, it is valid to assume that the majority of the documents in the Twitter corpora are from the US or the United Kingdom. Thus, this is the assumed geographic location for the analysis. The same is roughly true for Facebook, on which part of the ICWSM 2011 corpus was used. Additionally, in order to be able to compare the properties over time, the time zone for all three corpora is set to Pacific Daylight Time (PDT), i.e. to the local time of Los Angeles, US, and San Francisco, US, the most Western time zone of the US.

Location: USA and UK

Missing meta data

Unfortunately, none of the corpora used in this thesis provided further sociological, demographic or geographically detailed meta data. Therefore the text stream properties can only be studied under the aforementioned assumption that the majority of the users lives within the US time zones[12].

As shown previously, smaller time windows reflect the characteristics of the text stream as well as large windows do. Thus, the adaptive window approach was used and thus absolute values, foremost in terms of *aggregation-based* properties, are not relevant; hence only normalised values, i.e. scaled to values between 0 and 1, per corpora, are reported. Normalisation also has

[12]It must also be taken into account that the conclusions drawn from the data are only valid for Twitter or Facebook users and are not a representative sample of the general population.

the advantage of all values being on the same scale and that changes in the data are better visible.

Furthermore, only clippings of the various time series progresses are presented in the text itself, in order to be able to show the required level of detail. The complete time series graphs can be reproduced using the scripts from the online repository (cf. page 230).

Finally, the time series data of the various properties will be shown in contrast to one or more reference properties where appropriate. This is because interesting facts about a property often only become apparent if they are put in context with other properties.

7.7.2 Property: corpus size

The corpus size is an aggregation-based property and a relevant textual property in information retrieval. It is the starting point for many term weighting schemes like *tf.idf*. Properties like the number of unique lemmata or the average count per lemma can be derived from the corpus. The investigation of this property is foremost interesting as the values are all derived from the dynamic sub-corpora that are defined by the sliding window mechanism that is used.

The dynamic sub-corpora are viewed from three different aspects:

1. Average lemma count
2. Total lemma count
3. Unique lemma count

The average lemma count indicates how often lemmata in general occur per average. The total lemma count indicates how many lemmata in total are within the current dynamic sub-corpus. For instance, combined with the total number of documents, one can calculate the average length of documents and investigate how that length varies throughout the day. Finally, the unique lemma count represents the number of distinct lemmata within a dynamic sub-corpus. This could be taken as an indicator of the variability of the text streams, e.g. if more distinct lemmata are used, then the stream probably covers more different topics.

TREC 2011 corpus

The aforementioned aspects of the TREC 2011 corpus size are studied from two points of view. One is an intra-day progress analysis. The other one is a

view onto the overall corpus time series progress.

Figure 7.8 shows the smoothed[13] normalised values for the aforementioned three properties for a snapshot of Thursday 27 January 2011.

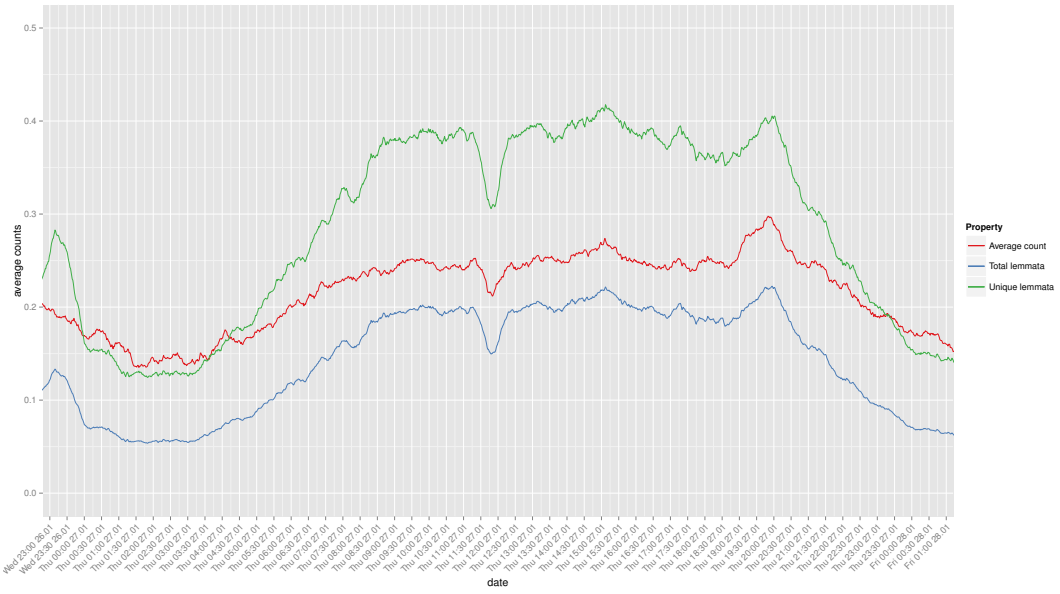

Figure 7.8: Time-series for average lemma count, total amount of lemmata and unique lemmata for 27 January 2011 in TREC 2011 corpus

Overall impression

First, the overall day time series progress is reviewed. The first thing that can be noticed is that between 12 a.m. and 2:30 a.m., the overall lows in terms of average counts per lemma as well as overall unique lemmata are reached. From 2:30 a.m. on the values start to increase, which could be an indicator of people starting to get up on the East coast – it is already 5:30 a.m. *Eastern Standard Time* (EST) – and while more and more people get up the length of the Tweets become more verbose and more different lemmata are used. The values reach their peaks around 7:30 a.m. and stay there until 7 p.m. From there on the values start declining again. This behaviour could be explained by the day / night rhythm of *regular* office people with a 9 to 5 job and a regular commute to work.

Detailed analyis

Now the graph is reviewed in more detail. The first interesting fact is that the unique lemmata count increases faster than the average count per lemma in the morning and decreases faster in the evening. This happens approx. around 6 a.m and 7:30 p.m. The increase of the average lemma count means that more people start writing about the same topics. Thus the same lemmata

[13]There are some gaps in the time series due to logging problems or missing data. This causes abnormal spikes that would distort the visual appearance. Therefore the time series are smoothed, in order to provide a clear view on the relevant data.

are used more often. Also the unique lemmata count goes up, but it increases in relation faster than the average lemma count. This means that more unique lemmata are used than in the morning and one could assume that people talk about more different topics. It could also mean that the style of writing becomes more versatile as more different lemmata are used. This assumption can also be supported by the fact that the total lemma count – in relation – does not increase as fast as the unique lemmata count, i.e. the total number of lemmata does not increase as fast as the uniquely used lemmata.

Another interesting point of this graph is the low around 11.45 a.m. PDT. This could be an indicator of lunch time. According to (Wikipedia, 2015f) the usual lunch time in the USA is between 11 a.m. and 1 p.m. This would fit the observed time series progress. The decrease starts around 11.15 a.m. PDT, reaches its low at 11.45 a.m. and is back at the same level at 12.30 a.m. These data – without having more detailed information about the geographical origin of the Tweets – could mean two things. First, people on the East Coast tend to eat lunch later than people on the West Coast, therefore both coincide at the same time in the graph and thus cause the low, or the majority of the Tweets comes from the West Coast and thus their number is sufficient to influence the overall time series progress.

Lunch time

In contrast to the intra-day analysis, figure 7.9 shows the three properties for the complete TREC 2011 corpus. Using the snapshot of nearly two weeks helps to reveal overall trends. In this case the corpus variety is the lowest around 1 a.m. in the morning, then rises to its peak around 9 a.m., stays nearly at the same level till 5 p.m. and then drops back to its low around 1 am. Besides this regular pattern there are always peaks (e.g. 25 January 5-6 p.m. PDT which is the probably the State of the Union address of President Obama which started at 9 p.m. EST) or changes in the ratios that can be considered as indicators for potentially interesting periods in the text stream which do not follow the regular pattern. Such peaks can be subject to topic detection and tracking systems, which register and analyse such uncommon processes.

Events 2011

Figure 7.10 shows the same corpus properties for a one-week period of the Events 2012 corpus. This graph roughly supports the interpretation of the TREC 2011 corpus data in terms of the time series progress.

But as this corpus has a much higher event density, it reflects properties of the corpus better, which can be seen in the more uneven time series progress. Furthermore different time periods mean different real-word events trigger-

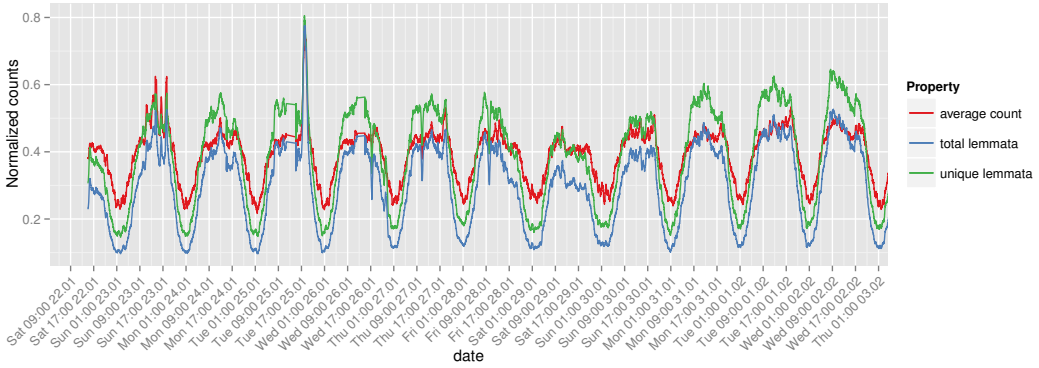

Figure 7.9: Time progress of corpus properties for the whole TREC 2011 corpus

ing Tweets. Thus the peak on 22 October 2011 7 p.m. or the low at 23 October 2011 3 p.m. could be triggered by some major real-world event and thus are interesting points in times for further investigation.

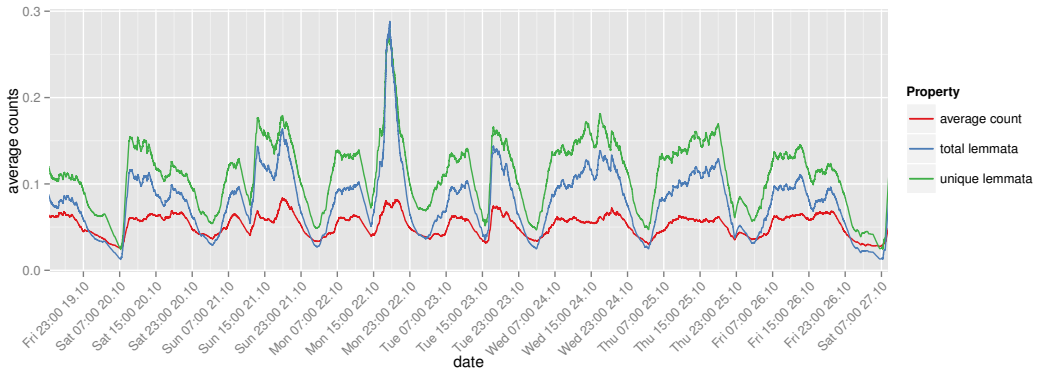

Figure 7.10: Unique lemmata, average lemma count and total number of lemmata of Events 2012 corpus

ICWSM 2011

Finally, the corpus properties of the ICWSM 2011 corpus are shown in figure A.1. Although this is a corpus based on Facebook posts, it shows the same time series progress as the other two corpora. Its event shows the same *switch* in terms of *unique lemmata* vs. *average count per lemmata* in the morning around 5 a.m. and in the evening around 7 p.m.

The abnormal peaks around 2 p.m. are also interesting. As these occur two days in a row this might be an indicator of some recurring pattern. Thus it would be reasonable to monitor a longer period in order to verify the pattern.

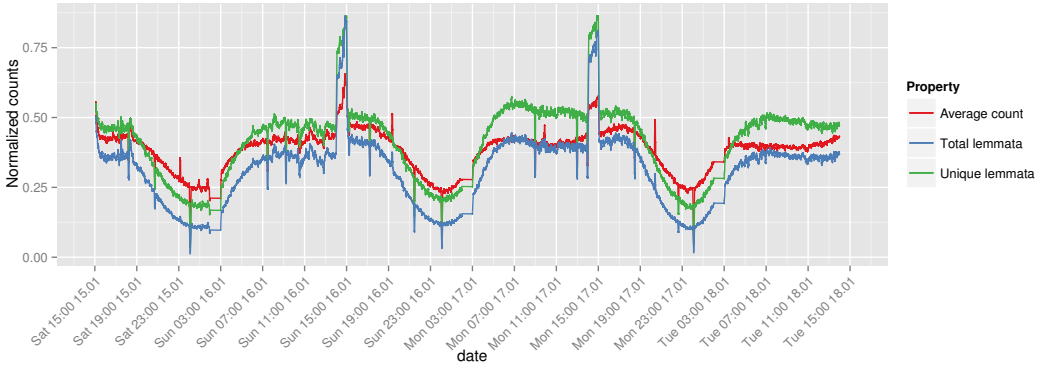

Figure 7.11: Unique lemmata, average lemma count and total number of lemmata of ICWSM 2011 corpus

7.7.3 Property: document length

The next property that is studied is the document length. In social media this is a very interesting property, because the length of a document tells something about the prevailing verbosity on a given social media platform. Sometimes the documents are limited by the platform itself, like on Twitter or its Chinese counterpart Weibo, but more often they are not limited like for instance on Facebook. But whether restricted or not, this property can tell a lot about the character of the exchanged messages. The most renown service in terms of restricted document length is Twitter, because it was the first service to limit messages to a certain number of characters. Facebook posts or blog posts in general are not limited by technical means, nevertheless the length the graphs show that the length centres around a certain value.

Content and function words

To recapitulate, this property is an *instance-based* property, i.e. it depends only the document itself. The document length is comprised of two values. The first is the number of *content words* and the second one is the number of *function* or *stop words*. In order to prevent any confusion the *content words* are represented by the *lemma count property* in the graphs. Thus these two names are used interchangeably.

A function word in contrast is a word that is "[..] used in word classification for a word whose role is largely or wholly grammatical, e.g. articles, pronouns, conjunctions "[p. 203]Crystal:2008vr. This means such words occur frequently, but do not convey an additional meaning about the topic of the document in contrast to *content words*, "which have a stateable lexical meaning"(Crystal, 2008, p. 108). Together these two types of words form the total length of a document. For the analysis of the document length, the two values are shown separately in the graphs, in order to spot interesting changes in their time series progress.

The stop word list that was used for the analysis is a combination of stop word lists provided by (Rijsbergen, 1979, p. 12), (Brahaj, 2009) and (Webconfs, 2013).

TREC 2011 corpus

Figure 7.12 shows the time series for the normalised *stop word count average* and *lemma count average* properties of the TREC 2011 corpus using a 90-seconds sliding time window.

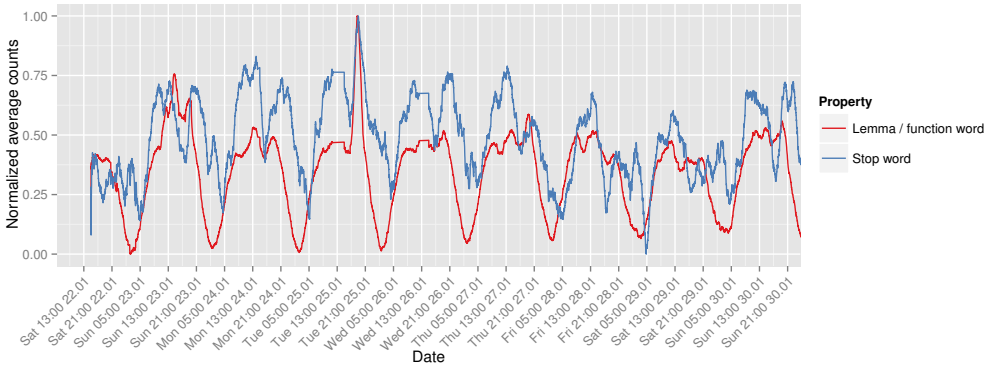

Figure 7.12: Normalised *content word* vs *stop word* average count for TREC 2011 corpus

The first observation is that the *content words* always reach a low around 2 a.m. PDT. This seems reasonable, because at this time the majority of people on the East Coast is still asleep, and the majority of the people on the West Coast probably has gone to sleep.

The second observation is that the peaks are reached between 9 a.m. and 5 p.m. There are small ups and downs in this time period, but in terms of the change this is negligible. Again the peak around the State of the Union address is clearly perceptible. In general, based on the data and time series it is valid to assume that people use more content words during the working hours than during the evening and during the night. Foremost, it is interesting that a drill down into which words are used by whom and when could reveal interesting insights, but as mentioned not enough meta data were available and an exhaustive analysis of each detail is not possible within the scope of this dissertation.

While the content word time series progress is not too surprising, that of the stop words offers an interesting detail. If you have a look at the ratio (cf. figure 7.13 of *content words* vs *stop word*) it seems to be a fairly regular process

with lows of the ratio around 3 a.m. in the morning[14]. Not too surprising, because this only means people use less content words during the night time.

Drunk people?

But taking closer look at the two values (cf. figure 7.12), it is clearly visible that while the content word value is dropping on Friday, Saturday and Sunday morning around 1 a.m. (as it does during the working week), the stop word ratio is increasing, which contrasts its performance during the working week. Although the ratio in figure 7.13 conveys the notion of a fairly regular and "uninteresting" progress, the application of another type of graphing the values provides different picture for the weekends. Thus a hypothesis could be derived from this graphs that people going out and having a drink tend to use more stop words than usually. A potential way to verify this, would be to check the *spelling error* property.

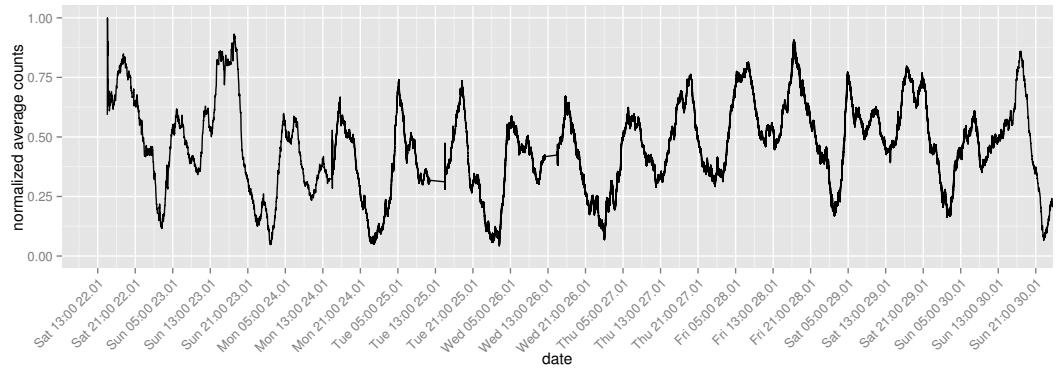

Figure 7.13: Normalised ratio of *content words* and *stop words* for TREC 2011

Events 2012 corpus

Figure 7.14 shows the time series for *content words* and *stop words* for the time span of 18 October 2012 0 p.m. PDT to 26 October 0 p.m. PDT. Again this graphs supports the observations made for the TREC 2011 corpus that *content words* increase during the day, while *stop words* do so during the night. The *weekend morning* observations for the *stop word* increase during the weekend morning hours is also confirmed.

Nevertheless, as this corpus has a higher event density it also shows a bit more differentiated view. This means the graph oscillates more than the TREC 2011 graphs do. Furthermore there are two abnormal *weekend morning* peaks during the day, on Monday 22 October 1 p.m. and Tuesday 23 October 1pm. This could be due to chance, but in any case this could be another good

[14] A low in this graph means fewer *content words* compared to *stop words*

starting point for an in-depth analysis.

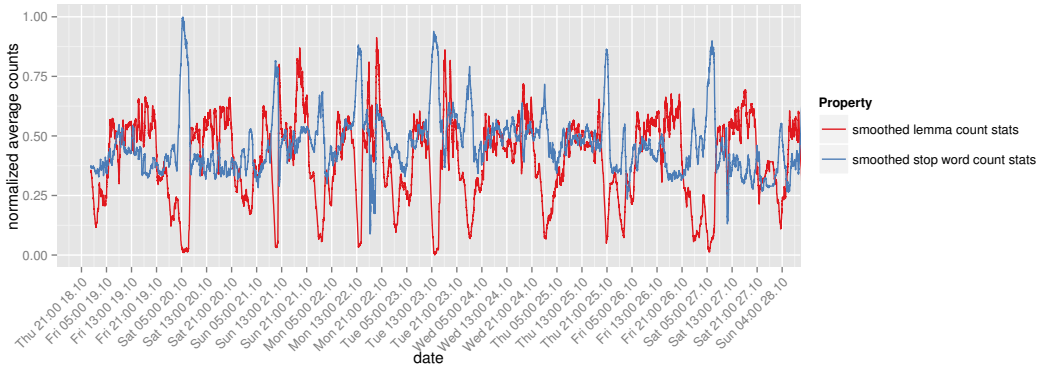

Figure 7.14: Normalised *content word* vs *stop word average count* for Events 2012 corpus

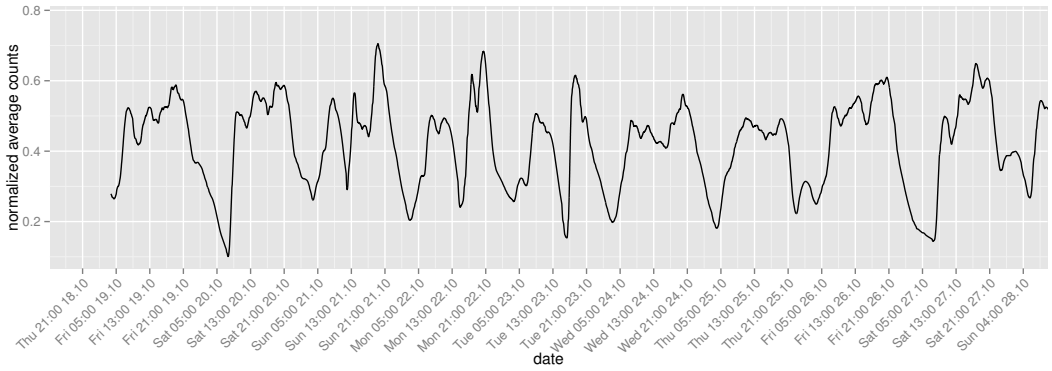

Figure 7.15: Normalised *content word / stop word ratio* for Events 2012 corpus

ICWSM 2011 corpus

Finally, the ICWSM 2011 corpus is studied in terms of the *document length* property. Again the event density for this corpus is much higher than that of the TREC 2011 corpus. Therefore the time series progress is much more uneven and oscillates more even though the values are smoothed by the length based event window[15].

Figure 7.16 shows the *stop word* and *content word* property over the used corpus period. The first observation is that a Facebook posts based corpus also follows the same day / night rhythm as the Twitter corpora do.

Another interesting observation is that the *stop word ratio* does not increase

[15]The horizontal lines as well as the few vertical outliers in the graphs are due to measuring errors

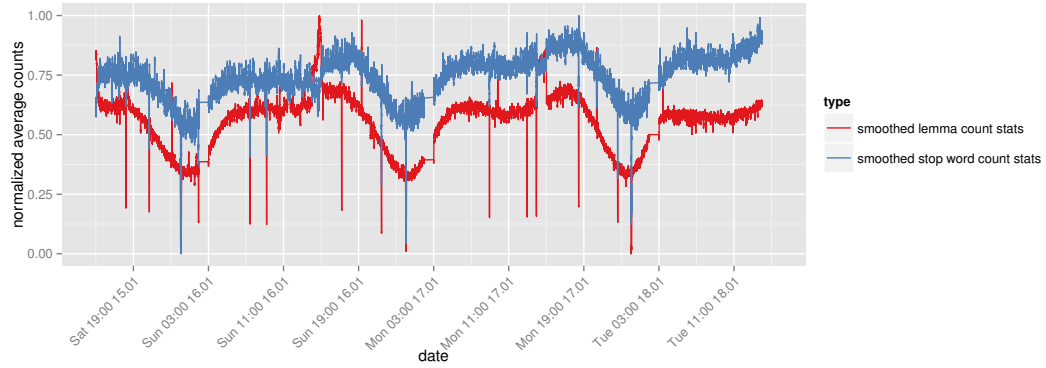

Figure 7.16: Normalised *content word* vs *stop word* average count for ICWSM 2011 corpus

during the morning hours like the Twitter corpora do. This could be an indicator of people on Facebook being more mindful of their tone, even in the morning. Yet another interesting starting point for a thesis with a more sociolinguistic focus.

The normalised ratio of *content words* and *stop words* (cf. figure 7.17 *only* supports the assumption of the day / night rhythm and also highlights the peaks on Sunday 16 January 2 p.m. and Monday 17 January 2011 2 pm.

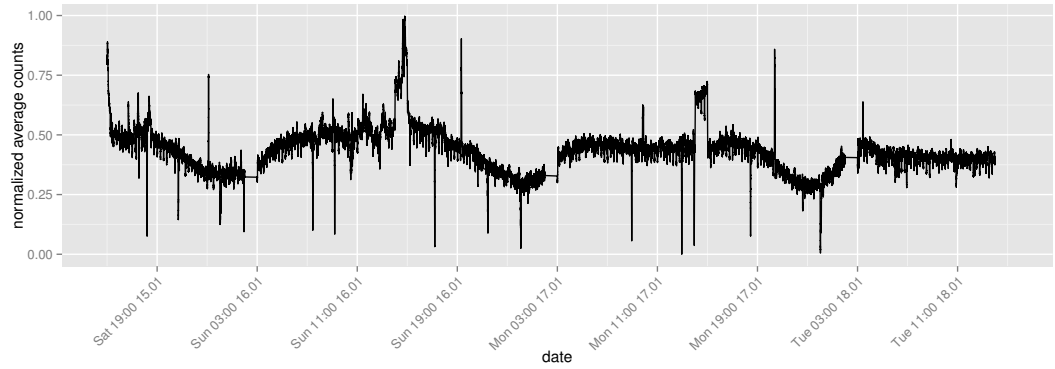

Figure 7.17: Normalised ratio of *content words* and *stop words* for ICWSM 2011

So to summarise, base on the ICWSM 2011 Facebook based corpus that it can be assumed that a Facebook corpus follows the same day / night rhythm as a *Twitter corpus*, but the people tend to better watch their language. It also seems that the Facebook is more used throughout the world than *Twitter*[16], because the highs and lows are not as significant as within the Twitter corpora. In order to get a clearer picture a location based filter would be required.

[16] At least for the time period when the corpora where collected

7.7.4 Character related properties

This section studies various character related properties are studied. In general character related properties are

- Overall character count
- Upper case count
- Lower case count
- Number count
- Special character count like *& % $ # _ { }*
- Spelling errors

For the analysis in this section *upper case* or *lower case* were used, because these properties form the whole content of a document (except for numbers and special characters). That is why it is interesting to investigate how these properties evolve over time and how their time series progress differ. This allows for deriving interesting conclusions from the underlying text stream.

Upper case and lower case

The case count properties reflect the number of upper case and lower case letters within a document. In English the majority of the words is in lower case. Upper case is only used for proper nouns or in social media or chats to express shouting. So if the rise of upper case statistics coincidences with the rise of proper nouns within the same time period, then nothing interesting has happened as more proper nouns consequently mean more upper cases. But in contrast if the increase does not coincide with an increase of occurrences of proper nouns or nouns in general, one could assume that people have started yelling and the tone within the stream can be considered "louder". If this happens at certain times of the day, assumptions can be made about the current state of a text stream.

But numbers can be also of interest. An increase in numbers could be due to an increased number of documents related to stock quotes, time schedules, addresses or geographical locations. If such an increase is spotted it is also an interesting starting point for an analysis.

TREC 2011 corpus

At first, the time series progress of *lower case* average counts is reviewed. In figure 7.18 these the are plotted over the period from 23 January to 30 January

2011. The first observation that can be made is that the lower case average counts reach a peak around 12 a.m. to 1 a.m. every day, then drop down to a lower level around 5 a.m., reside there for another eight hours till 1 p.m. and starts dropping to their daily low around 9 p.m. from where they rise back to the peak around 0 a.m. This happens regularly over the given time period and thus can be considered as a pattern.

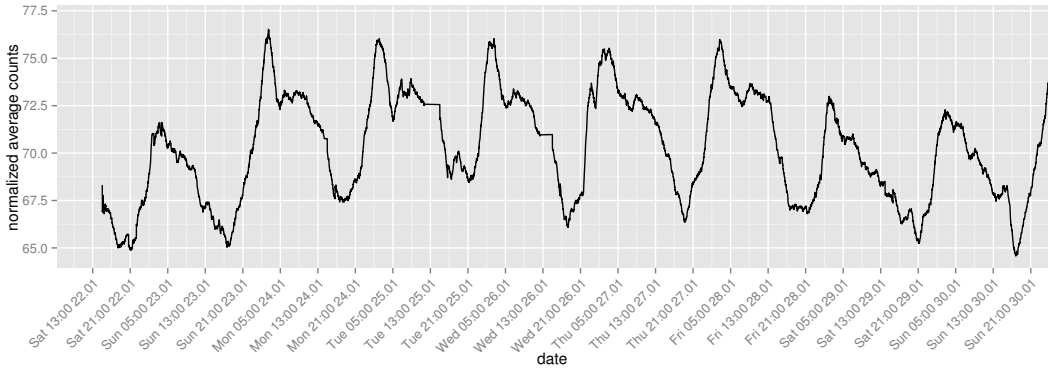

Figure 7.18: *Lower case* average count values for TREC 2011

If the lower case average count is contrasted with the previously studied content word property (cf. figure 7.19) it becomes apparent that during the day more but shorter content word are used, while during the night fewer but longer ones are used per Tweet. The observation is backed by the comparison of the lower case and the lemma length property[17]. This is an interesting finding, because it suggests that people tend to change their way of writing depending on the time of the day – at least for this specific corpus –, because otherwise the lower case counts would follow along the document length property.

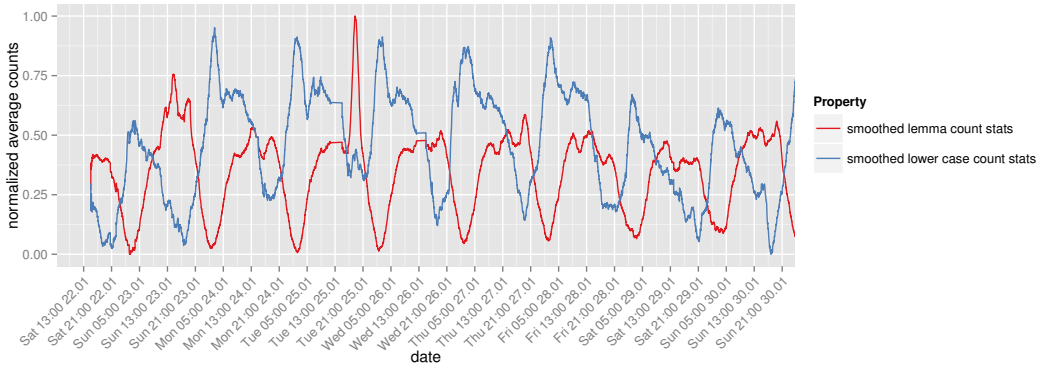

Figure 7.19: Normalised *lemma count* vs *lower case count* for TREC 2011

[17]The latter shows the average length per lemma over time

The upper case property does not follow such a regular progress (cf. figure 7.20). From an intra-day time series progress perspective this value is at a high level around 1p.m. and then drops off till 9 p.m., but there are many irregular ups and downs inbetween. As previously mentioned due to the semantic meaning of upper case within social media such irregular peaks, e.g. Wednesday 26 January 11 p.m. or Sunday 30 January 5 p.m., could be an interesting pattern for further investigation.

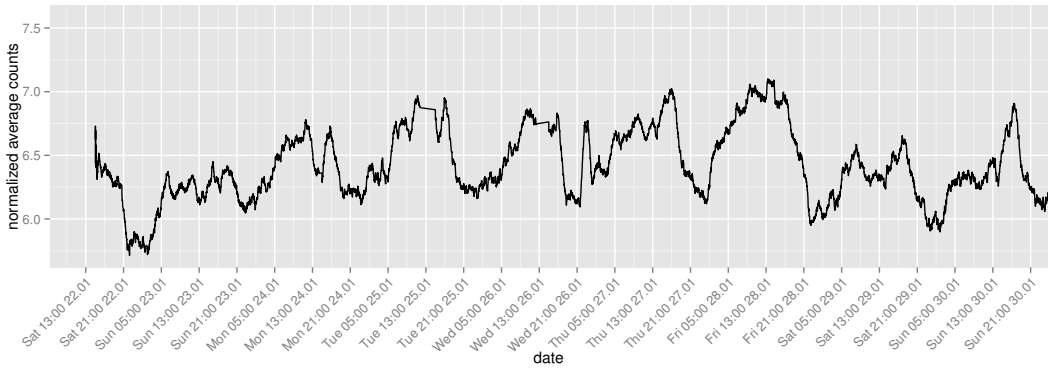

Figure 7.20: *Upper case* average count for TREC 2011 corpus.

Events 2012 corpus

Figure 7.21 shows the lower case average counts for the Events 2012 corpus for the time period from 22 October 2012 to 29 October 2012. This graph also supports the findings from the TREC 2011 corpus in terms of the lower case property. This means, the time series progress is roughly the same, i.e. it peaks around 12 a.m. to 1 a.m. drops to a lower level, resides there and then goes down to its low around 9 p.m. Furthermore, as this corpus has a higher event density it shows more abnormal outliers, e.g. Tuesday 25 October 12 a.m. or Thursday 25 October 8 pm. Again, this shows that the chosen approach is valid to detect patterns as well as to spot outliers as starting points for further in-depth analyses.

The upper case property shows more regular progress for the Events 2012 corpus than for the TREC 2011 corpus and follows quite smoothly along the lower case average counts (cf. figure 7.23). This maybe due to the higher event density.

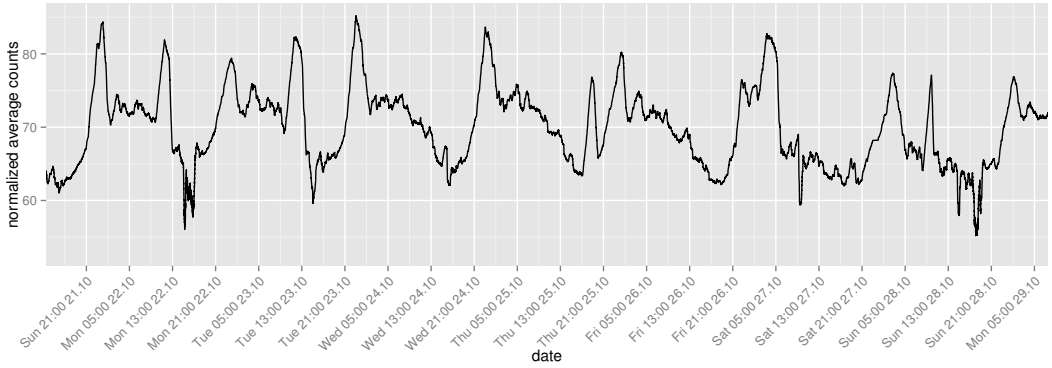

Figure 7.21: *Lower case* average count values for Events 2012 corpus.

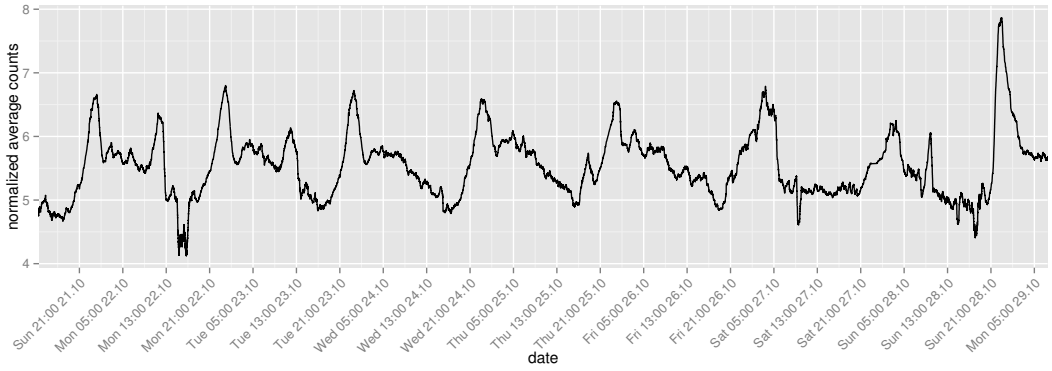

Figure 7.22: *Upper case* average count values for Events 2012 corpus.

ICWSM 2011 corpus

Figure 7.24 shows the average lower case counts for the ICWSM 2011 corpus. This corpus is Facebook based, and an interesting fact is that it shows a different time series progress than Twitter corpora. First, Facebook posts seem to be longer during the working days than during the weekend (at least Sunday). Furthermore the post length already starts increasing at 3 a.m. in the morning and peaks around 7 a.m. During Sunday 16 January 2011 the length recovers during the afternoon, while during the working days the length seems to decrease continuously to a low at 3 a.m. in the morning. Furthermore, it seems that people tend to send the longest Facebook posts during the morning probably before or while going to work, which is fairly interesting, because one could assume that people post longer documents after work or on the way home when they are about to start their free time.

The comparison of upper case and lower case average count does not show anything abnormal for this corpus. Both follow almost exactly the same time series progress. Thus it could be assumed that in contrast to the Twitter corpora, Facebook users do not change their tone over the day, because both

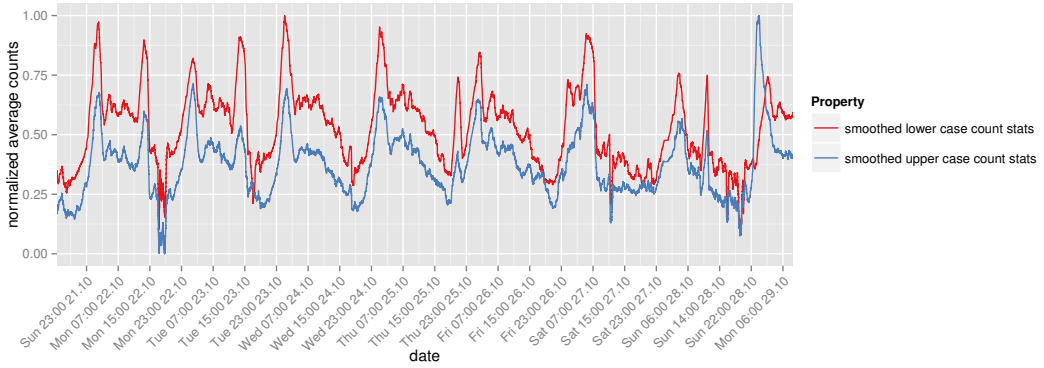

Figure 7.23: *Upper case / lower case* counts normalised for Events 2012 corpus.

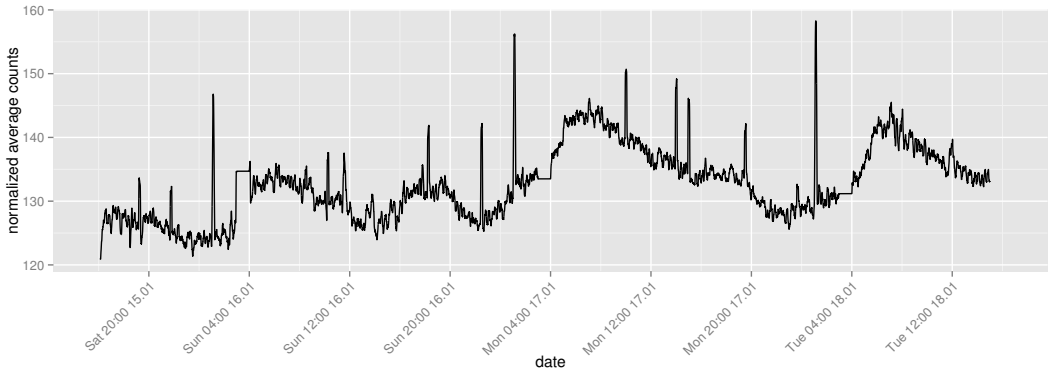

Figure 7.24: Lower case counts for ICWSM 2011 corpus.

property values stay constant over time and thus an increase in upper case counts is just a consequence of longer documents in general, because longer documents increase the probability that also more *upper case* characters occur. However, in order to be able to verify this claim, are location based analyses would be required, as it is possible that the worldwide use of Facebook dilutes the peaks and lows in the graph.

7.7.5 Property: parts of speech

Another interesting aspect is parts of speech within a corpus. *Parts of speech* (POS) are for instance nouns, verbs or pronouns, i.e. parts of speech are classes of words that share common grammatical properties and thereby form syntactic categories (Kroeger, 2005, p. 26). The analysis of parts of speech can reveal characteristics about the corpus in terms of style used for writing a document. E.g. using more verbs than nouns can be considered as a more lively, active style, because they describe events and actions, while using more nouns can indicate a more formal style, because they represent

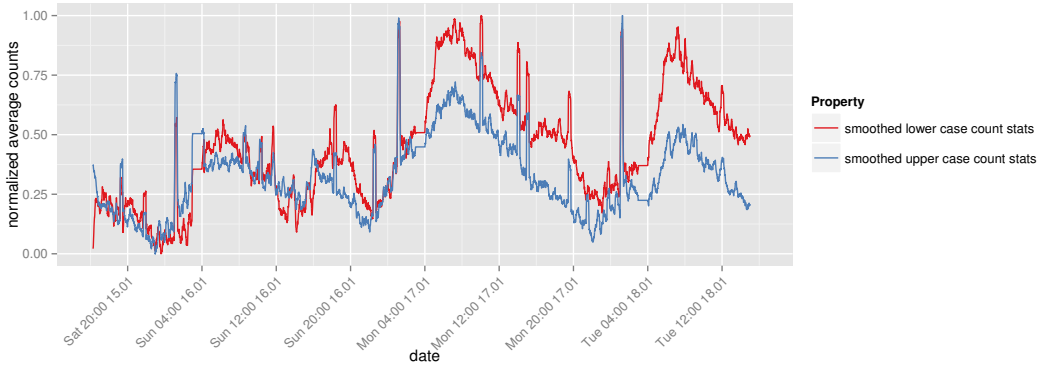

Figure 7.25: Normalised *upper case* and *lower case* average counts for ICWSM 2011.

persons, entities or locations (Kroeger, 2005, p. 33).

All three corpora were annotated using the *POS tagger* presented in (Gimpel et al., 2011). Not all of the available features were used, but only the ones that accounted for the majority of tags. Notably, these were nouns, verbs, pronouns, pre- or postpositions, adjectives and *miscellaneous*[18].

TREC 2011

Figure 7.26 shows all parts of speech properties logged for this corpus on a normalised scale that were. This, at first sight, dense graph (six time series) illustrates – over the time period of 22 January 2011 to 30 January 2011 – that the ratios of the parts of speech do not differ from each other and all follow the same time series progress. This means the semantic structure of Tweets is consistent over time. The ratio of parts of speech does not vary and this is interesting, because one could assume that the semantic structure differs during the day.

Figure 7.27 provides a more detailed look at the 27 January 2011, which it underlines that the ratios drop and rise in the same fashion[19]. There might be some interesting issues regarding the miscellaneous parts of speech as they oscillate more than the other ones. But if a look on the overall volume is considered, this becomes negligible.

The exact alignment of the time series progress of the various parts of speech categories becomes more obvious when a look at the absolute smoothed count values is taken into account (cf. figure 7.28). The graph also shows that

[18] This includes abbreviations, foreign word, symbols or just garbage (Gimpel et al., 2011, p. 2)

[19] The offset of the misc parts of speech is due to small a peaks and lows over the complete course of the data.

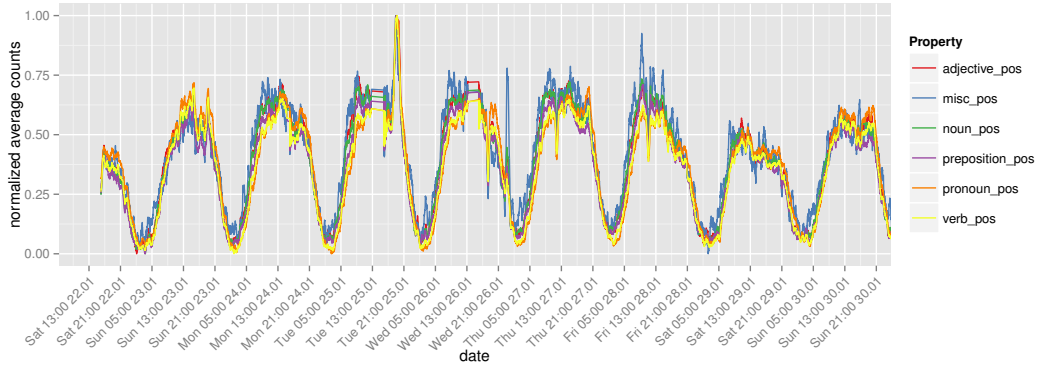

Figure 7.26: Normalised parts of speech average counts for the TREC 2011 corpus.

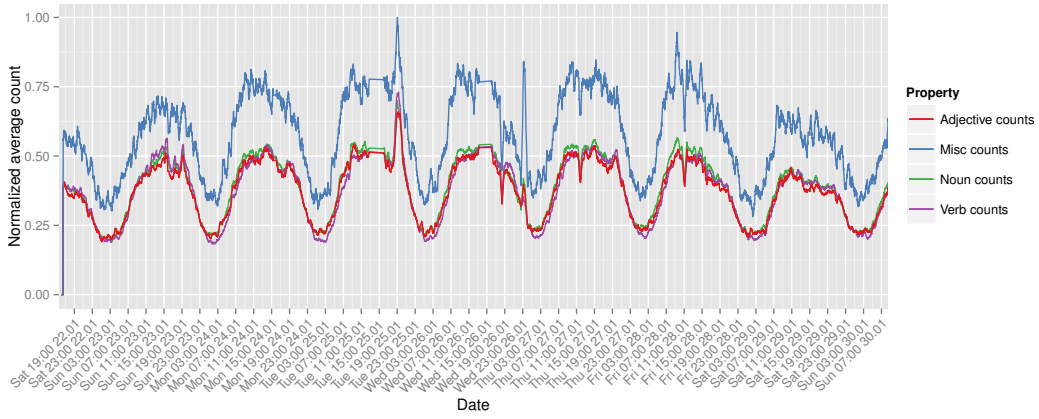

Figure 7.27: Parts-of-speech for a single day of the TREC 2011 corpus.

the corpus is dominated by nouns and verbs. Adjectives or miscellaneous parts of speech occur less frequently. Foremost the large distance between nouns, verbs and adjectives leads to the assumption that a more basic style of writing is used that rather focuses on subjects (nouns) and actions (verbs) than on describing a certain state in more detail using adjectives[20].

Events 2012

The observations from the TREC 2011 corpus apply to the Events 2012 corpus as well. A look at figures 7.29 and 7.30 underpins this observation.

ICWSM 2012

It is interesting to see that the observations from the Twitter corpora are also valid for the Facebook corpus. As Facebook posts are not limited in size

[20]The purpose of different parts of speech is described in (Kroeger, 2005, p. 33ff)

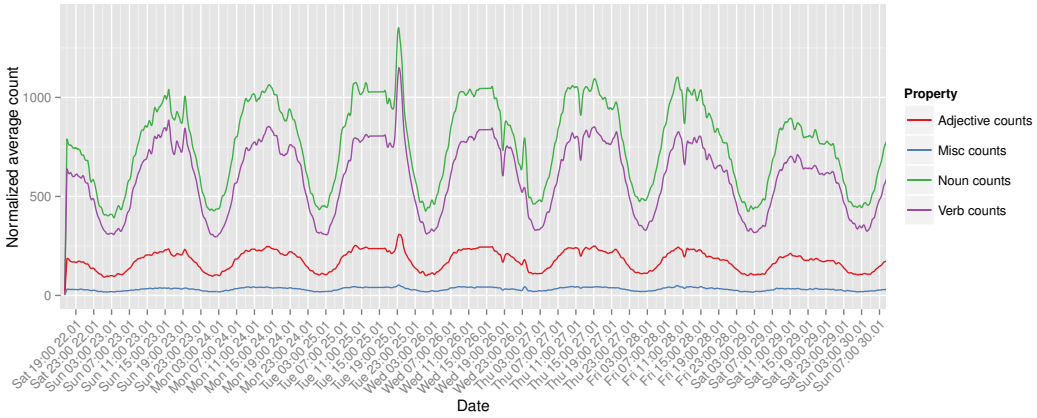

Figure 7.28: Absolute noun and verb counts for the TREC 2011 corpus.

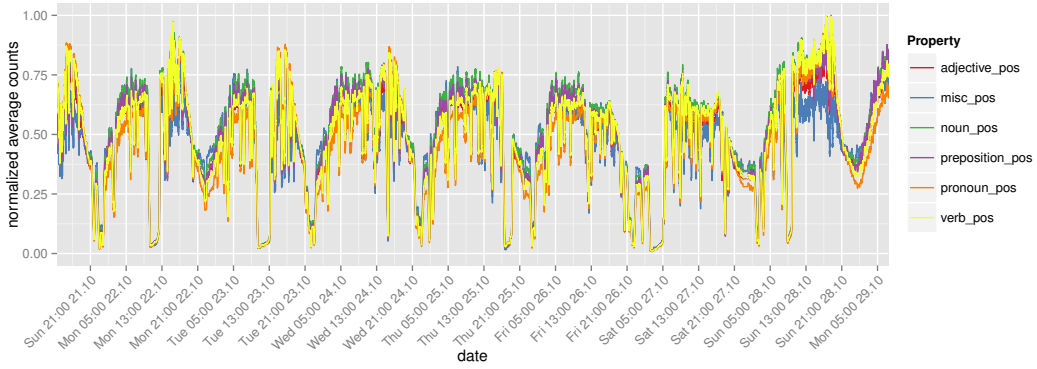

Figure 7.29: Normalised parts of speech for the Events 2012 corpus.

and therefore more explicit, an elaborate style would be possible that could favour adjectives or other auxiliary semantic parts-of-speech. Nevertheless, this does not seem to be the case, and Facebook users seem to maintain a focused and constant style of writing (cf. figure 7.31). Again the nouns are more important than the verbs, but the difference between nouns and verbs is much less significant than within the Twitter corpora. This could be due to the fact that there is not character limit and thus more verbs can be used if required. In Twitter it seems that people drop verbs instead of nouns, in order to express their intentions.

7.7.6 Property: hashtags

The next property that is studied are *hashtags*. Hashtags are a very interesting property of a text stream, because they are currently the common sense indicator for topics. Twitter revived the usage of hashtags that was used in the *Internet Relay Chat* IRC and made the concept popular. It is now used

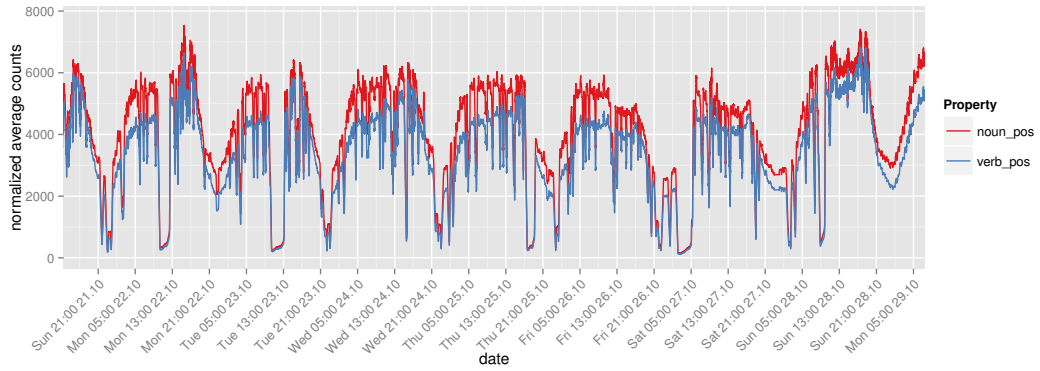

Figure 7.30: Absolute noun and verb counts for the Events 2012 corpus.

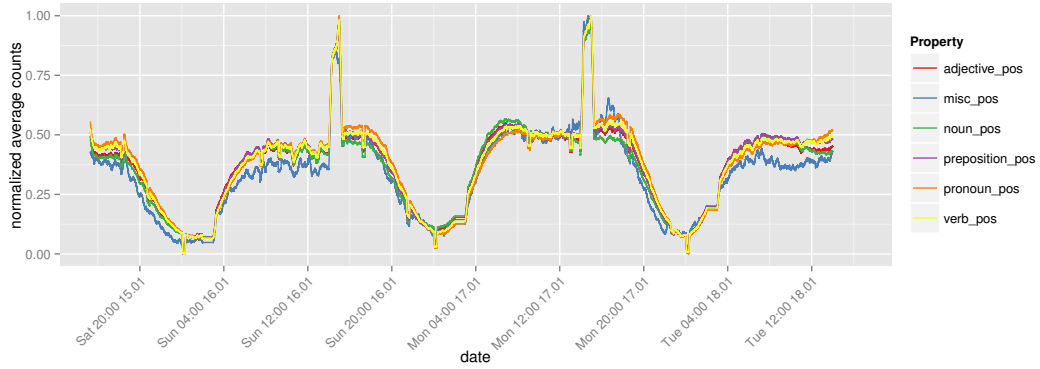

Figure 7.31: Normalised parts of speech for the ICWSM 2011 corpus.

widely by Facebook, Instagram or Google+ (Bennett, 2014). Thus monitoring hashtags is a good indicator of what people are talking about and an obvious as well as a straightforward way to monitor topics. The popularity spawned multifaceted research studies. Among others things, hashtags were used for conversation tagging (Huang et al., 2010), analysis of sport events (Blaszka et al., 2012) or sentiment analysis (X. Wang et al., 2011). This variety of research underlines the importance of hashtag analysis.

This section studies meta statistics about hashtags, i.e. this section studies overall counts and unique hashtag counts and does not focus on specific topics. In order to conduct studies on specific topics represented by hashtags the hashtags of interest can be used to define a context using a Context Filter. Thereby the focus of the stream is narrowed down and a detailed analysis can be conducted.

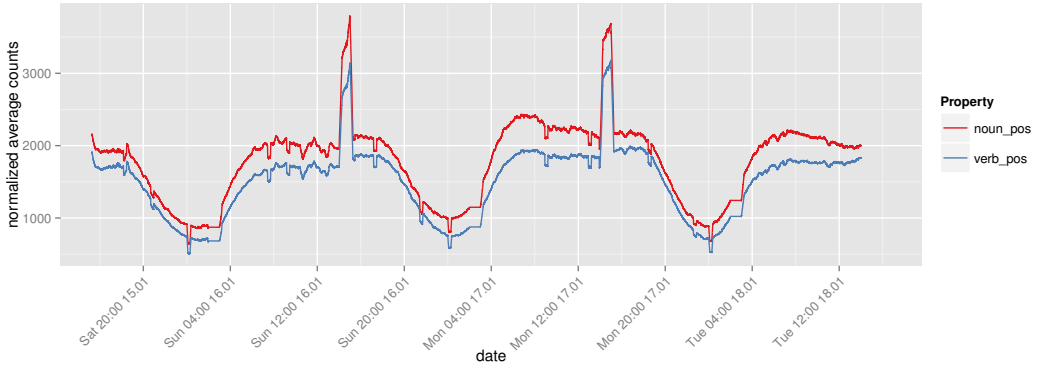

Figure 7.32: Absolute noun and verb counts for the ICWSM 2011 corpus.

TREC 2011

Figure 7.33 shows the normalised, smoothed figures of the hashtags statistics for the TREC 2011 corpus between 23 January and 30 January 2011. Again this time series follows the day / night rhythm that has been already observed for other properties like *document length*. It is Interesting to see that the trend during the day is not very regular and oscillates heavily even though the values have been smoothed before plotting. The State of the Union peak also becomes clearly visible.

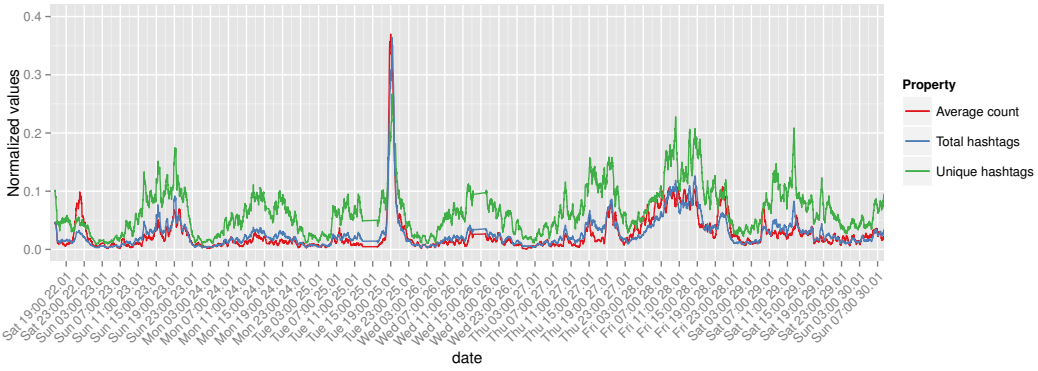

Figure 7.33: Normalised hashtag statistics for one week of the TREC 2011.

Figure 7.34 shows a detailed view of a three day range. As stated previously, the hashtag statistics oscillate more than for instance the unique lemma counts, which were plotted for comparison. In general it can be stated that the most interesting section for a drill-down are the spots where there are peaks or where the time series progress diverges, i.e. where one of the three properties (unique hashtags, total hashtags, average count) do not follow the same progress.

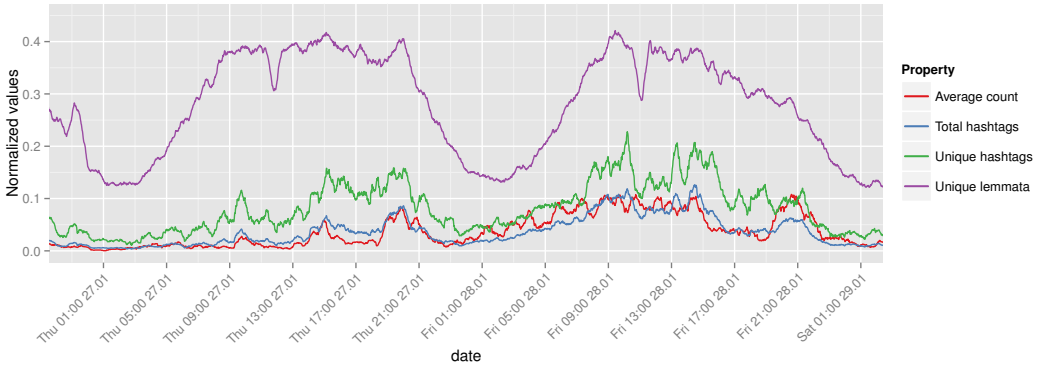

Figure 7.34: Normalised hashtag statistics for three days of the TREC 2011 corpus.

Events 2012

The time series progress of the Events 2012 corpus for the time period between 27 October 2012 and 31 October 2012 roughly support the observations from the TREC 2011 corpus. Again it shows more details due to its higher event density, but nevertheless this corpus also affirms the notion that hashtags are proportionally more often used during the day than during the night. Additionally, two more interesting events for this specific corpus can be spotted. First, the peaks for the hashtags are around 1 p.m. for these three days, at least for Sunday, Monday and Tuesday. Surprisingly, the peak on Sunday is different from the ones on Monday and Tuesday, because the total amount of hashtag usage goes up, but the unique hashtag count does not. A brief analysis of the used hashtags shows that two tags dominate the peaking period on Sunday: #IfIWasInvisibleIWould and #sandy. The latter is of particular interest, because hurricane Sandy hit the East Coast of the USA on that day. The second interesting date is 31 October. The hashtag counts go down remarkably on Wednesday 31 October afternoon. This day is Halloween and it seems to indicate that people are more busy within *trick'n'treating* than with tweeting. As Twitter is not that popular with teenagers (Lenhart, n.d.), it is probable that the parents have to watch their children and thus have less time to Twitter. But as mentioned this is only an assumption which would require more detailed data and analysis.

ICWSM 2011

Finally, at look at the Facebook corpus is taken. Figure 7.36 shows the smoothed absolute values of unique hashtags for the IWCSM 2011 corpus. Although hashtags have been a genuine Twitter specific concept, it becomes clear that hashtags are also used frequently on Facebook. Although the time series

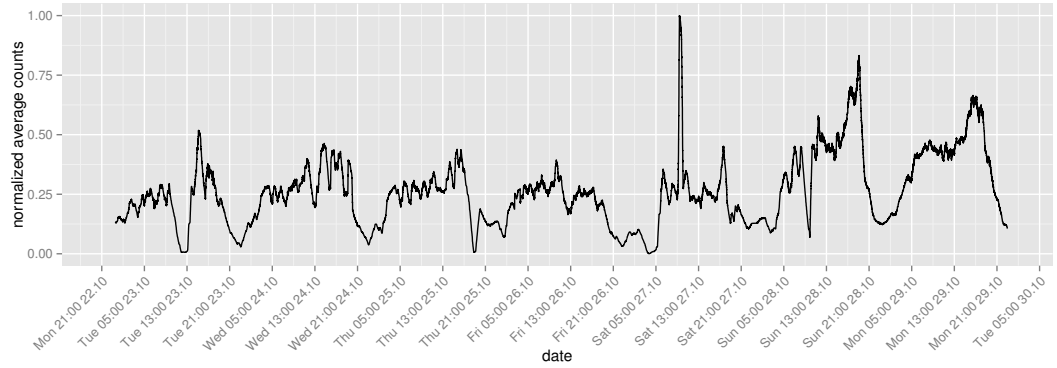

Figure 7.35: Normalised hashtag statistics for the Events 2012 corpus.

progress is much more irregular, it can be seen that the same holds true for the Facebook corpus; users tend to use more hashtags during the day than during the evening. This is interesting, because it would be also reasonable the other way round; i.e. that people use more hashtags during the event, in other words after work when they supposedly have more time to post interesting topics.

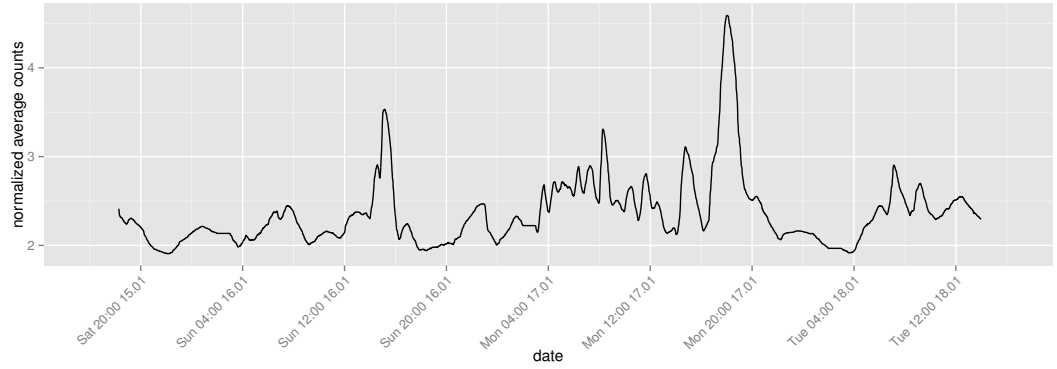

Figure 7.36: Normalised unique hashtag counts for the ICWSM 2011 corpus.

7.7.7 Studying sentiments in dynamic corpora

Finally, a lexicon-based sentiment analysis is presented. Sentiment analysis is a topic of high interest, as it allows gaining insight about the personal feelings that are expressed within a document and thus making assumptions about the overall emotional trend within the text stream. This aspect is important for many kinds of applications such as customer satisfaction analyses or the analysis of the political atmosphere before elections. The seminal paper in this area was presented by (Pang et al., 2002).

Nowadays, sentiment analysis still poses many open questions. Natural language is very ambiguous, therefore it is difficult for machines to understand the real sense of an utterance and that is why sophisticated machine learning algorithms are often applied to this task. Nevertheless, lexicon-based approaches have also been shown to perform well (cf. (Taboada et al., 2011) or (Ounis et al., 2008, p. 8)). Therefore, the appearance of lexicon-based sentiment terms within social media streams can be seen as an expression of sentiment within the stream.

The three types of sentiments – positive, negative and neutral – are studied in comparison, i.e. all three types are plotted within the same chart in order to compare the progress of sentiments over time. Additionally, the sentiments are studied under the aspect of splitting the week into working days and weekend as well as morning, noon, afternoon, evening and night, in order to spot changes in the mood of the text stream.

TREC 2011

Figure 7.37 shows the time series progress for the positive and negative average counts of lexicon-based sentiment tokens in the TREC 2011 corpus. It shows that the mood of the stream rises already at 4 a.m. in the morning, while the negative utterances reach their daily low at the same point. One could assume that people are in a bad mood in the morning, but it seems that the positive thoughts outweigh negative ones even in the early morning. In contrast the peaks for the negative mood are always in the evening around 8p.m. to 12 a.m.

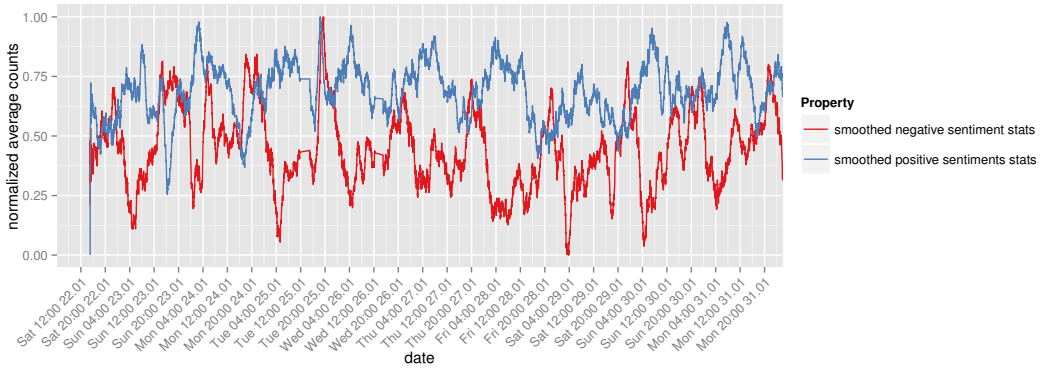

Figure 7.37: Normalised positive and negative sentiment count average for the TREC 2011 corpus.

In general the text stream of this specific corpus is more positive than negative. This is shown in figure 7.38. It shows that the average use of positive

utterances is always higher than the negative ones.

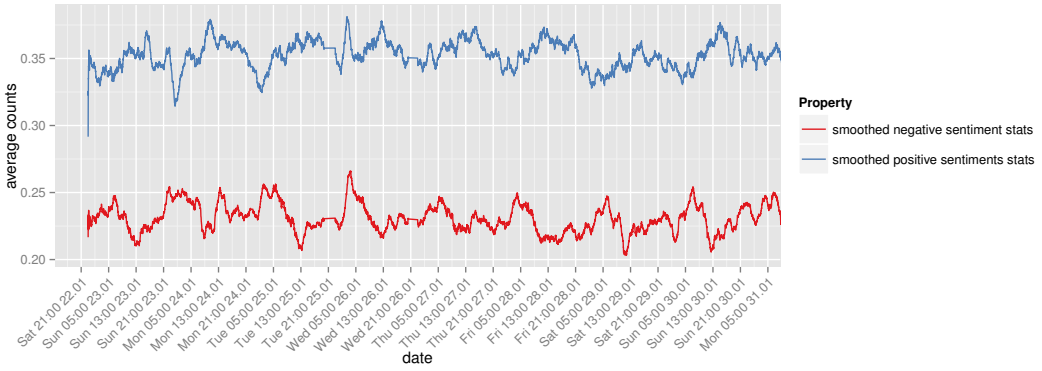

Figure 7.38: Absolute positive and negative sentiment count average for the TREC 2011corpus.

Events 2012

Figure 7.39 also shows that the average text stream of the Events 2012 corpus is more positive than negative. A difference to the TREC 2011 corpus can be seen when the normalised values (cf. figure 7.40) are analysed. The peaks of the positive sentiments are always around 9 p.m. in the evening while the TREC 2011 corpus has its peaks during the day. Reasons for this could be a general shift of the sentiment usage within the Twitter corpus or due to the sampling of the corpus. Again, more detailed data could reveal interesting aspects.

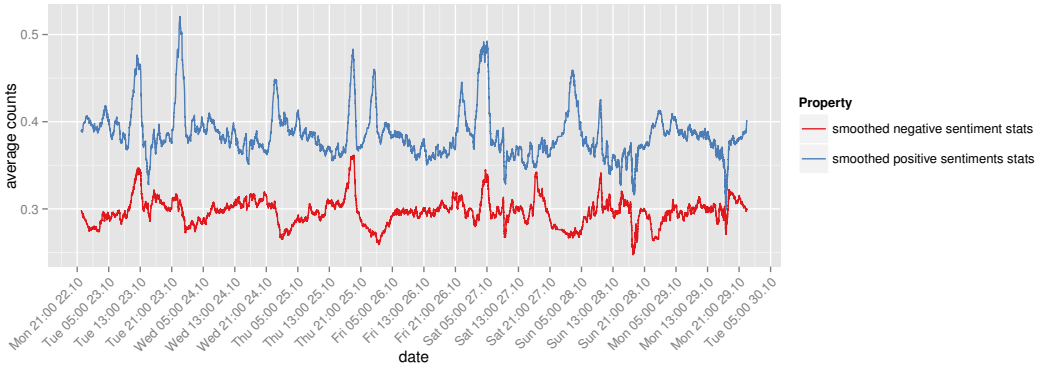

Figure 7.39: Absolute values of positive and negative sentiments average counts Events 2012 corpus.

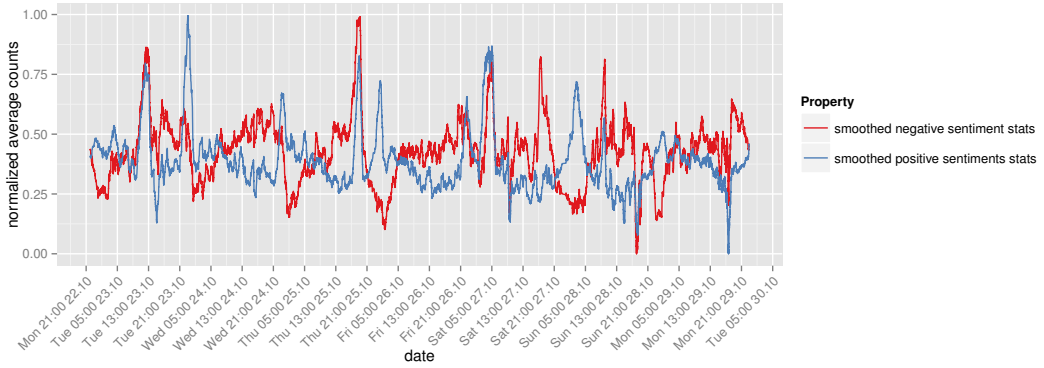

Figure 7.40: Normalised values of positive and negative sentiments average counts Events 2012

ICWSM 2011

For the Facebook based corpus it can also be shown that the overall mood is more positive than negative (cf. 7.41). Figure 7.42 demonstrates that people on Facebook – at least for this specific corpus – seem to be in the best mood before work and in the evening. At least they tend to use proportionally the most positive utterances from 4 a.m. to 8 a.m. and 4 p.m. to 8 p.m.

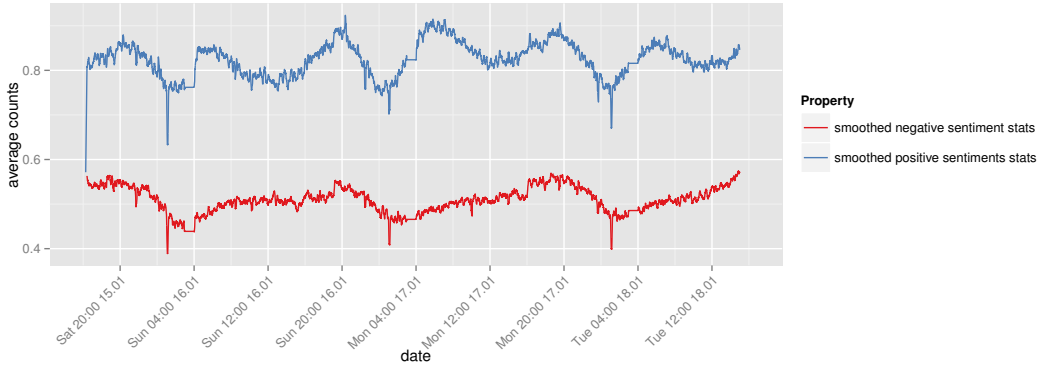

Figure 7.41: Absolute values of positive and negative sentiments average counts ICWSM 2011 corpus.

It is difficult to tell if the aforementioned assumptions are really true without detailed demographic, geographic or further meta data. Nevertheless, the application of lexicographic sentiment analysis using event processing shows that a real-time mood monitoring is feasible. If the data were enriched with demographic and location data this could lead to interesting monitoring results.

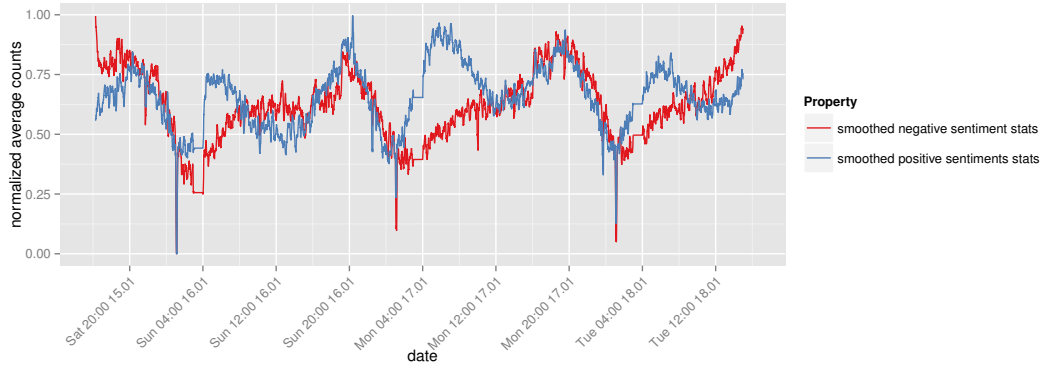

Figure 7.42: Normalised values of positive and negative sentiments average counts ICWSM 2011 corpus.

7.8 Summary

This chapter studied the structure of dynamic corpora that are based on sliding windows. First, several sliding windows of various sizes were used to study how the structure of the dynamic corpora vary. A visual comparison of the time series graphs was also conducted, as well as several statistical checks to assess the similarity of various sliding window sizes. Dynamic time warping was used to measure the similarity of the time series graphs and to justify the visual evaluation. It could be shown for all three social media corpora that large windows can be approximated by smaller windows, and thus the smaller ones can be used as resource efficient alternatives.

Nevertheless, a lower bound or at least a minimum number of documents must be present, but this depends on the event density of the corpus. For a low event density corpus like the TREC 2011 corpus, which only provided four events per second, a window of approx. 90 seconds already showed the same performance in terms of the information filtering quality as a large, 2-hour sliding window. For higher density corpora like the Events 2012 and the ICWSM 2011 corpus, which provided both around 30 events per second, sliding windows of 5 to 10 seconds provided the same structural quality as windows of a few minutes.

A static upfront definition of a sliding window size would require an upfront analysis of a subset of a text stream. As text streams can change rapidly, the upfront could be outdated before its results are applied. Therefore, methods to adaptively and automatically determine a reasonable window size were studied. One approach was to monitor the average index entropy using a value change detector. It could be shown that index entropy yields good results in terms of precision, but the recall values are significantly worse than

for a static window. A second approach was based on determining the window size by keeping a minimum amount of documents within the sliding window. The number of documents was determined using a statistical estimator for sample sizes and the calculated sample size was as the target function of the window adjusting mechanism, which adapted the sliding window size to keep the desired amount of documents in the sliding window. Both approaches yielded similar precision values as the static approach. Nevertheless, the recall values were worse. In consequence this means that for precision oriented tasks adaptive windows can work as well as static windows while, having the advantage of not being tested upfront.

Finally, a visual analysis of different text stream properties was presented as a real-word example of the proposed reference architecture and event reference model. By monitoring various text stream properties it was demonstrated that "internet linguistics" (cf. (Zappavigna, 2012, p. 192f)) can be performed successfully using dynamic corpora and the proposed event reference model. While the results only represented assumptions about the underlying text streams – for an exact analysis more meta data would have been required – it could be illustrated that monitoring several text stream properties can be done in real-time. Such analysis could also yield interesting insights and appealing starting points of socio-linguistic investigations. The comparison, combination and contrasting of various text stream features, for instance document length vs. spelling errors during the morning hours on the weekends, yielded interesting findings that can be subject to further investigation.

The next chapter studies information filtering aspects in detail. This comprises the study of four essential information filtering components, *viz.* filter policies, term weights, thresholds and query expansion.

Chapter 8

Studying event-driven filtering and text stream analysis components

The previous chapter showed how to determine a suitable lower bound for a sliding window as basis for a dynamic text corpus. Furthermore, various properties of three social media corpora were studied using *StreamFI*. That chapter helped to verify that the chosen dynamic corpus approach is suitable to dissect and analyse text streams.

This chapter focuses on information filtering and four essential components. Overall two main subjects are addressed: first to further verify that an event based information filtering system is valuable for text stream processing and, second, to evaluate well-known information filtering principles in the context of dynamic text corpora and high velocity text streams[1].

The following four core components are studied due to their relevance to the reference processes presented in 4.2.5:

1. Filter policies
2. Threshold methods
3. Term weighting
4. Query expansion

The choice of the components was also based on their relationship to event processing, as described in section 4.2.1. Furthermore, the selection contains the essential and fundamental components of information filtering. Of course, there are more aspects that influence the performance of an event

[1]The work presented here is similar, in terms of its purpose, to the work presented in (Kürsten, 2012), who conducted an extensive study on static corpora and information retrieval methods.

processing based filtering system. However, these are exclusively related to information retrieval without any relationship to event processing, e.g. thesaurus creation, or they are more specialised topics that can be subject to further research like special similarity measures adapted to the temporal nature of text streams. The evaluation and analysis is done using *StreamFI*.

Chapter structure

The chapter starts with a short discussion about the methods that are used. Subsequently, different temporal types of search profiles are introduced that are used throughout the evaluation. Afterwards, the first component is studied: filter policies. Filter policies are rules that try to generate a preliminary set of potentially relevant documents by eliminating as many as *obviously* irrelevant events as possible, e.g. if you are only interested in English Tweets than any Spanish Tweet can be ignored, i.e. every Tweet that does not have the language attribute set to English is removed. Thereby, filter policies fulfil an important task, because they act as a gatekeeper to the later stages of the filtering system that involve more computational resources. This helps to decrease the load on the filtering algorithms and reduces the danger of too many false positives. Three different ways of filter policies are also studied.

The second component consists of term weighting schemes and scoring. A multitude of term weighting algorithms such as *tf.idf* or *Okapi BM25* are used in social media analysis, but there is no exhaustive comparison and final conclusion as to which term weighting schemes works best in an high velocity text stream scenario with dynamic corpora. Therefore, several seminal term weighting schemes are compared in terms of their influence on the filtering performance. A few new methods are studied as well.

The third component studied is *thresholds*. In terms of information filtering these are essential, as they are used to decide if a document is relevant or not. Therefore, several approaches on how to calculate threshold values based on the scores of the documents are studied. Their performance is compared in terms of the common performance indicators like precision or recall. Furthermore, these approaches are compared to methods that use machine learning to classify documents. The goal is to show how both approaches performed in a scenario where only dynamic corpora and a limited set of information are available[2].

The final component that is studied is *query expansion*, to which search profiles are closely related. A search profile reflects and describes the information need of the user and is therefore important for the filtering quality of

[2]It must be noted that the focus here is on approaches using limited memory, i.e. the subject is a rather low cost approach. Large scale machine learning using massive amounts of computing units that have large amount of training, test and evaluation data available is not subject of this thesis.

the system. A search profile in its simplest form can consists of only a fixed list of search terms, or, in its most distinguished form, of dynamic adaptation rules that reflect the users interests, social network, mood or location. In the context of this dissertation, the focus is only on term-related search profile properties due to the lack of user or ambient profile information in the used corpora. Therefore, methods on how to exploit the temporal nature of a text stream to generate new terms for expanding the initial query are studied.

8.1 Introduction

8.1.1 General notes on used methods

Some general remarks must be made, in order to correctly judge the presented results.

First only textual and linguistic features are considered. The social graph of users, their popularity, their number of followers, etc. is not used. This is, first because it is just not scope of this thesis and second the required data was not fully available (TREC corpus) or not easy to recreate (Events corpus).

Second, no external evidence in form of links was used. Tweets are limited in their size and thus link shortener services are used to embed links to external websites. This information was not taken into account, because the resolution and processing of URLs within less than a second for several hundreds of links did not seem feasible within the given technical possibilities, and second it was already shown that external evidence improves the quality significantly (for Twitter cf. (Albakour et al., 2013), (Liang et al., 2012) and for short documents in general cf. (Sahami & Heilman, 2006)).

Third, the results of the evaluations depend heavily on the available relevance judgements. As already stated in the corpus description, both the quality of the TREC corpus relevance feedback as well as of the Events corpus is not to be taken without critical assessment. The relevance feedback for the TREC corpus was indeed crafted by domain experts, but a few samples revealed some not very convincing relevance assessments. The Event corpus relevance feedback was partly generated automatically and thus this also has to be used carefully. More remarks are provided in section 8.1.6.

Fourth, the selection of the methods that were used for comparison had to fulfil at least one of two requirements. One was that the method was an accepted state of the art method like e.g. Rocchio or Okapi BM25. The other was that the method could be implemented based on the available publications.

Corpus	MRF-FI	BM25	MRF-SD	MRF-BM25	MRF-FS
AP	0.2077	0.2149	0.2128	0.2210	0.2261
WSJ	0.3258	0.3332	0.3429	0.3512	0.3553
ROBUST04	0.2920	0.2892	0.3092	0.3101	0.3079
WT10G	0.1861	0.1948	0.2140	0.2129	0.2129
GOV2	0.2984	0.2971	0.3360	0.3476	0.3398

Table 8.1: Comparison of test set mean average precision for language modelling (MRF-FI), BM25, MRF model using language modelling weighting (MRF-SD), MRF model using BM25 weighting (MRF-BM25), and MRF learned using our proposed feature selection algorithm (MRF-FS) taken from (D. A. Metzler, 2007, p. 144)

In general, the study attempted to used state-of-the-art methods. First they are usually used by other authors as well in terms of presenting their results. Thus, it is possible to compare the results to their research work without having to know the implementation details of their new algorithms, as the comparison is performed via the seminal algorithms. Second, a plethora of approaches to e. g. term scoring, thresholds, etc. exists that makes it impossible to re-implement and compare to all those approaches. Therefore, the use of state-of-the-art algorithms represents a legitimate shortcut to provide a traceable and sound comparison with other research.

Expected improvements

Finally, a note on the dimensions of improvements that can be expected. At first sight the differences between various methods and algorithms, which are presented afterwards, may seem insignificant, but the figures of improvements or degradations are within the common scale of information retrieval research. For instance, table 8.1 is taken from the PhD thesis of Donald Metzler, a renowned information retrieval researcher: his *Markov Random Field* approach to document scoring is compared to different BM25 variations[3].

Effect, not overall performance

It is also important to mention that the goal of this analysis is to show the effect of different methods, not to design and implement the best possible filtering system for short text documents. This is due to four reasons. First more data with more meta-data would be required, in order to build optimal features for scoring or learning. Second, the use of external evidence would be required to improve the performance to a level of a competitive commercial system. But this is not possible, because the resolution and analysis of URLs, for instance, is not possible within the boundaries of near real-time processing requirement due to network latencies. Third, in order to achieve

[3]For details on Markov Random Fields cf. (D. A. Metzler, 2007).

the best possible filtering results more components must be included and fine-tuned. And fourth, each studied component contains parameters that have effects on the performance, e.g. the number of used feedback terms, the exponential decay value λ,, the number k for the top k frequent pattern mining algorithm, etc. The parameters used in this dissertation were obtained by evaluating a various evaluation runs.

Thus, the focus is to highlight the improvements or degradation of methods in contrast to a predefined baseline. Moreover, the order of the studied components does not imply a gradual improvement of the presented performance indicator. This is because the order of the system's components depends on the goal of the system, i.e. for instance is it more precision or more recall oriented; and this depends on the particular use cases the system targets.

8.1.2 Different temporal categories of search profiles

Foundation for later evaluation

In (R. Jones & Diaz, 2007) a temporal classification of queries was presented. This classification can be used to divide the queries, which are used in this chapter for evaluation purposes, into the three classes: atemporal, temporally unambiguous and temporal ambiguous queries. Atemporal queries are "relatively time-invariant" (p. 14), i.e. they represent a constant information need and the documents that are supposed to be detected occur regularly. Temporally unambiguous queries are quite specific to a certain point in time. This is the most common type of queries in social media where breaking news are of major interest (p. 14f). Temporally ambiguous queries, in contrast, look like the previous type but cannot be assigned exclusively to a certain point in time, e.g. the query "soccer world championship" can refer to a current championship – if there is one ongoing at the time the query is issued – or to any previous or forthcoming one (p. 15).

Empirical categorisation

This section briefly summarises to which type of profile each of the used search profiles belongs according to (R. Jones & Diaz, 2007, p. 14). The categorisation is based on a survey of five people that were asked to classify the topics from the TREC and the Events corpus into the aforementioned categories. The inter-rater agreement was calculated using *Fleiss' kappa*[4] and yielded a value of *0.553* for the *TREC Corpus* and *0.482* for the *Events Corpus*[5]. These values indicate a "fair" agreement (cf. (Landis & Koch, 1977, p. 165))

[4]Fleiss' kappa is used to assess the inter-rater agreement of a certain number of raters (>2) (Fleiss, 1971, p. 378ff).

[5]The *p-value* for both values was less than .001.

Corpus	Atemporal profiles	Temporally unambiguous profiles	Temporally ambiguous profiles
TREC 2011	4, 5, 13, 23, 24, 25, 27, 29, 32, 33, 45, 47, 49	2, 3, 7, 8, 12, 14, 15, 17, 18, 19, 22, 34, 35, 37, 43, 48	9, 10, 11, 20, 21, 28, 30, 31, 38, 39, 40, 41, 42, 44
Events 2012	15, 50, 154, 162, 163, 170, 454	42, 43, 46, 51, 53, 56, 74, 164, 167, 187, 249, 254, 256, 434, 436, 456, 484	14, 49, 54, 55, 146, 155, 169, 486, 488

Table 8.2: Categorisation of search profiles

on the chosen classification[6].

8.1.3 Filtering methodology

The filtering methodology adheres to the task definition provided in the 2012 TREC real-time filtering pilot task (Macdonald et al., 2012). The reason for this is, the TREC guidelines provide a standardised and comparable task description for information filtering in a scientific environment. The methodology is defined as follows:

> The task will follow the adaptive filtering methodology from the TREC filtering track (see [2]). At the starting point for a given topic, systems are allowed to use the entire corpus prior to queryTweettime as background training data, and the query and queryTweet as a positive training example. (Note that queryTweets are not guaranteed to be good examples!)
>
> Following this, the system must emit a score as well as a retrieval decision for each Tweet from queryTweettime to querynewestTweet. Any unreported Tweet will be assumed to have a score of negative infinity with a negative retrieval decision. Systems must process the Tweets in Tweet ID order and output a decision before processing any further Tweets.
>
> If for a specific Tweet, the system emits a positive retrieval decision, the system is allowed to know the relevance value of that Tweet (or 0 for an unjudged Tweet). This simulates (exceedingly

[6]The detailed list on the temporal profile evaluation can be found in the online repository.

> prompt) feedback with which the system is allowed to incorporate in future retrieval decisions. If the system emits a negative retrieval decision for a Tweet, it does not get access to any relevance information for that Tweet, although the Tweet may be used otherwise (for example for background models or IDF distributions).
>
> Participants may use topics MBx | x mod 5 == 1, that is, MB1, MB6, MB11, MB16, MB21, MB26, MB31, MB36, MB41, and MB46, along with all their relevance data, as development topics to tune their systems. Results should only be submitted for the remaining 39 topics. (Macdonald et al., 2012, paragraph 32-35)

8.1.4 Topics

The *TREC 2011* corpus and *Events 2012* corpus both provided relevance assessment information. The *TREC 2011* corpus provided approx. 50,000 relevance judgements for 49 topics. These relevance judgements were assessed by human evaluators.

The *Events 2012* contained approx. 150,000 relevance judgements for 502 topics. These were created using semi-automated methods. Therefore, this corpus contains more relevance judgements, but also suffers from the problem of each automated pseudo-relevance approach that the relevance judgement must not be necessarily one any human evaluator would make. They also can be just plain wrong. Furthermore the topics of the *Events 2012* corpus are not described in 2-4 term sized search strings, but are represented by a long, verbal descriptions often taken from a news article that was judged as being most relevant to the topic. Experimentations showed that using the complete paragraph wise description yielded a recall value of approx. 80%, but also led to long processing times and many false positives. Therefore, in order to leave space for the algorithms to do their work a more human like approach was taken and 2-5 term long search strings were derived and used for the experimentations.

From the 49 available *TREC 2011* topics, 36 were used. Ten of the 49 topics were reserved for training purposes by the *TREC* authorities. As training topics none of these would ever be included in any performance report using this corpus. Therefore, in order to preserve comparability to other research works that use this corpus, these topics were also excluded. In addition three more topics (15, 18, 49) were excluded by the author, because only one or two positive relevance judgements were available for these topics. The was done, because if these topics were included and the one or two relevant documents

were retrieved by an algorithm, the recall value for this topic would be 1,0 and thus yield a biased increase in the average recall value. The average recall value is often reported by researchers using this corpus, in order to show the effectiveness of their algorithms. It is not known if other researcher also excluded this topics, but nevertheless they are excluded here, in order to avoid any unfair influence on performance values.

From the *Events 2012* topics 16 topics were chosen. The selection was made based on the amount of the available relevance judgements with in the chosen subset of the corpus. In order to keep the runtime reasonable for a single evaluation run only 16 million out of 150 million available documents were used. The time period was 28 October 2012 to 2 November 2012.

An overview of the search queries that were used to study the two corpora can be found in appendix B.

8.1.5 Baseline Run

In order to be able to quantify the effect of the parametrization of the studied components it is necessary to define a baseline against which the different runs can be compared. The baseline is gathered using this default configuration and always displayed during the respective analysis:

- Term weighting: *Okapi BM25*.
- Relevance feedback: No relevance feedback.
- Threshold: Average score[7].
- Filter policy: *static* and at least 2 search term must be matched[8].
- Window size: 360 seconds for TREC 2011 and 20 seconds for Events 2012 corpus[9].

The abbreviations that are used afterwards are defined as follows:

- *TP*: True positives: the system and the assessor agree on the classification of a document as being relevant.

[7] A Tweet is considered as relevant if the score of the Tweet exceeds the average score of the the scored Tweets within the defined sliding window.

[8] There were no search string within TREC or Events corpus that contained less than three filter terms. Thus two terms as a minimum match seemed reasonable.

[9] Even though it was shown that adaptive windows provide a decent approximation, static windows are used for the evaluation, in order to make it easier to reproduce the results and compare the various evaluation runs.

- *FP*: False positive: the system considers a document as relevant that the assessor did not classify as relevant or that was not evaluated at all. The latter is a reason for large number of false positives within the evaluation runs.[10].
- *TN*: True negative: the system and the assessor agree on the classification of a document as being not relevant.
- *FN*: False negative: the system classifies a document as irrelevant while being judged as relevant by the assessor.
- *Precision*: Macro precision value. Average precision over all topics, i.e. the single precision values per topic are summed and divided by the total number of topics[11].
- *Recall*: Macro average. Average recall over all topics, i.e. the single recall values per topic are summed and divided by the total number of topics
- *T11SU*: Macro-averaged T11SU value.

8.1.6 Remarks on corpus and feedback quality

A previously mentioned, the absolute values of the performance indicators like precision or recall depend heavily on the available relevance assessment. Thus, these values depend highly on the agreement or disagreement of the system and some assessor on the classification of a document. If both agree the values increase; if not they decrease. This obvious fact is important to remember when the absolute figures of the studied methods are evaluated, because if the assessment of the assessors, against which the systems performs or from which the system tries to learn from, are poor or inaccurate, the results of the system are probably also inferior to evaluations where the so called *gold standard*, i.e. the assessments of the assessors, are well crafted and reviewed.

Assessment quality

This problem is pointed out here, because the two used Twitter corpora on the hand provide decent amount of relevance assessment, but on the other hand the assessments do not seem to have the best quality, if the relevance assessments are looked at in detail. As it is impossible to re-evaluate the

[10]The should be kept in mind when studying the presented figures because due to the effort of creating relevance feedback judgements many relevant documents are not judged at all which are probably returned by the system. But it is good style to report the false positives rate this way. Also the TREC guidelines support this notion.

[11]In contrast to this, micro-averages sum up the amount of tp, fp, tn, fn to calculate the performance indicators. This approach is prone to being dominated by single topics with large amount of documents(Schütze et al., 2009, p. 281) and thus would lead to a biased result. Therefore, micro-averages are ignored in this evaluation.

more than 50,000 relevance judgements of the *Twitter* corpus and more than 150,000 of the Events corpus, a few samples should illustrate the author's assumption. Table 8.4 contain a few examples to illustrate the claim[12].

Amount of feedback per topic

It is also important to keep in mind how many relevance assessments are available per topic. Discerning a good threshold, or to learn a decent machine learning model usually requires a large volume of high-quality data. But for the two corpora the relevance assessments were limited and also unevenly distributed. For the TREC corpus in average 60 positive relevant feedback documents were available with a standard deviation of 50. For the Events corpus the average was 202 with a standard deviation of 605. There, many topics only had one relevant document, but there were also with more than 10,000. Also due to the runtime for the evaluation of the Events corpus, only a subset was available which limited the amount of relevance feedback. Together with the biased quality of the relevance feedback assessments in general, not the absolute performance is important, but the relative performance.

8.2 Studying the effects of filter policies

The first component studied is *filter policies*. Filter policies represent a preliminary filtering facility that tries to efficiently remove as many "obviously" irrelevant documents from the stream as possible, in order to prevent the later filtering stages from being flooded by such documents. The means there are many documents that do not match basic requirements, like language or location, and can thus be filtered out easily. Filter policies are thereby only *true negative* focused. Two general types of filtering processes are studied in this section: *heuristic* and *machine learning based* filter policies.

This section is structured as follows: First related research is presented. Then, a formal definition of filter policies is provided. Third, different types of filter policies are introduced and described in detail. Fourth, the introduced filter policy approaches are evaluated. This includes an algorithm that automatically derives filter policies using a *frequent pattern* algorithm. Then, the effect of combining machine learning with the heuristic filter policies is investigated. Finally, the approaches are compared in terms of their effect on standard information filtering performance measures like precision, recall, sensitivity, etc. The evaluation is performed on a *macro average* level and also in terms of the previously presented profile categorisation.

[12]For the Events corpus there was no relevant category for "non relevant" documents. Thus only samples for classified as "relevant" are shown.

Cor.	ID	Query	Example	Classified as
TREC	9	Toyota recall	”Toyota faces further recalls: Toyota has recalled a further 1.7 million vehicles worldwide following safety fear... http://bit.ly/hiCtWJ”	Not relevant
TREC	20	Taco Bell filling lawsuit	“Taco Bell, This Is What Really Hides In Their Beef gizmodo.com/5742413/ #MustRead”	Not relevant
TREC	22	Health care law unconstitutional	“Fla. judge rules health care law unconstitutional http://ow.ly/1b7hph”	Not relevant
TREC	32	State of the Union and jobs	“State of the Union speech to focus on jobs: Obama http://goo.gl/fb/iT5er”	Not relevant
TREC	13	Oprah half-sister	“I am Oprah’s half sister.”	Relevant
TREC	28	Detroit auto show	“QuelvinJ Show!!”, “OCReggie another ped ax: RT Anaheim Trauma Alert Ball/Euclid FD onscn of 3 patients down Auto vs ped”	Relevant
Events	14	Giants win world series	“END IT TONIGHT, GIANTS !!!!!!”, “Sergio Romo’s beard!”, “I’m going to go streaking if we win tonight!”	Relevant
Events	15	Pride of Britain Awards	“I still got money from Britain idk wtf to do with it doe.”, “Britain gets first 4G mobile services http://f24.my/T1J4nKÂ ”	Relevant
Events	54	wins Chelsea Manchester United	‘beta chelsea..controllers of the man utd‘”	Relevant
Events	436	Sandusky child abuse	‘Dressing as Jerry Sandusky tonight #HelloKids” ,“It’s me, Jerry Sandusky, and the pope in the club”	Relevant

Table 8.4: Doubtful relevance assessment for Events 2012 corpus.

8.2.1 Related Research

The term *filter policy* has not been used in information filtering research as a scientific term, yet. Therefore, there is no directly related research available., but the principles of filter policies are based on accepted information retrieval research topics, which are briefly discussed now.

The first area is query formulation.This research area comprises how an query is posed against a search or filtering system. In its basic form, a filter policy is *just* a Boolean query and thus well known concept. But filter policy tend to go beyond Boolean term filters. This means while a Boolean filter is usually used as a term filter that restricts the retrieved document set[13] and filter policy tries to integrate more aspects, like linguistic, semantic or geographical concepts.

Also related to the approach of defining filter policies is the field of *event processing with uncertain data*. This topic was discussed by (Renners, 2013) or by (Wasserkrug et al., 2008). (Renners, 2013) studied how to use uncertain event data in a health care scenario. The authors presented methods on how to derive *Bayesian Networks* based on a given event model for health applications. Furthermore, they used *fuzzy logic* in order to match various domain rules. Their research is different, because it does not represent a stacked filtering processing for textual information, but a way to classify uncertain events into different categories. Such an approach could be integrated into an extended version of *StreamFI*, in order to improve the overall filtering performance. But this is not in scope.

8.2.2 Definition of filter policies

There is no general definition of the term *filter policy* in the area of information filtering. Therefore a *filter policy* is defined as follows:

> A filter policy is a set of rules that efficiently blocks the majority of the "obviously" irrelevant documents from a text stream for a given search profile. Filter policies are *true negative* focused only and can be chained as required.

In detail, a filter policy acts as a gatekeeper to the later filter stages. The majority of incoming documents is not relevant at all for a certain topic, and this

[13] As previously mentioned, Boolean term filters are the very first filter policy in the multi-level filtering processing

can be discerned using rules(=filter policies). Such rules are often straightforward and provide the basic requirements a document must fulfil to be potentially relevant. For instance, filter policies can be defined using one or more of the following criteria:

- How many (if at all) query matches does document contain?
- How many feedback matches are contained?
- Has the document occurred within a certain time period?
- Was the document created at a specific location or within a specific spatial range?
- Does the document have the right language?
- etc.

Such filter policies are usually created by domain experts and are based on the domain expert's judgement of which criteria a document requires to fulfil an information need. Additionally, there are alternatives to domain expert filter policies, which are also studied in the next sections.

8.2.3 Three types of filter policies

In this section, three filter policy approaches are compared. The first one – as just introduced – is based on the application of heuristics to generate filter policies. This means the policy rules are defined upfront by a human domain expert. This type of filter policies is static.

The second filter policy approach is based on *frequent pattern mining*. This means that a document is converted into a set of discrete text stream property ids that are fed into a *frequent pattern mining* algorithm. The algorithm generates sets of frequent text stream property ids that appear jointly. These sets can be used as filter policies.

The last approach is based on *machine learning classification algorithms*. Text stream properties from relevant and irrelevant documents are gathered, a model is trained and then applied to incoming documents. In contrast to a final classification of a document, in this stage the classifier is set to a low threshold so that only really absolutely irrelevant documents are filtered out.

Filter policies can be chained, i.e. it is not necessary to select a single approach, but they can be combined to form a sequence of filter policies. In this dissertation, the second and the third approach are intended to be an extension to the first approach, i.e. basic heuristic filter policies are always in place for every evaluation run, but are extended by one of the proposed approaches.

The two heuristic filter policy rules that are always in place during the evaluation runs are, first a Boolean term filter, i.e. each document must at least contain one of the terms in the search profile and, second, a temporal filter. For the TREC corpus a time range was provided by the TREC guidelines, during which a Tweet could be considered relevant. For the Events corpus only a lower bound was defined, i.e. the timestamp of the first relevant available Tweet in the corpus was used.

8.2.3.1 Heuristic filter policies

Wikipedia defines *heuristic* as "any approach to problem solving, learning, or discovery that employs a practical methodology not guaranteed to be optimal or perfect, but sufficient for the immediate goals" (Wikipedia, 2015c).

Thus the approach to design heuristic filter policies is to discard the majority of the documents based on simple, but efficient rules that can be derived by a common user or a domain expert. The process is to define properties, which should be fulfilled by documents, so the filter policy lets them pass. These criteria are often straightforward and obvious to understand. Usually heuristic filter policies comprise terms, location, time, language or geographical restrictions, i.e. such filter policies can filter a vast majority of the documents before they enter the more sophisticated and computationally more expensive stages of the filter processing.

Heuristic filter policies can be based on restricting – among others – the following aspects:

- Term
- Location
- Language
- Time
- Structure

Furthermore, it must be noted that the heuristic filter policies that are used in this dissertation are all in *disjunctive normal form*, i.e. the components of each single rule are concatenated via a logical *AND*, while all rules are combined using a logical *OR*. Each component of a single rule is a binary variable that can either be set to true of false.

8.2.3.2 Filter policies based on frequent patterns

Heuristic filter policies are based on the experience and the personal beliefs of the person defining the filter policy. Thus, this kind of filter policies is rather

subjective and often lacks sound justification. Even though such policies can yield a good filtering performance[14] it is desirable to have an approach at hand that allows generating filter policies automatically and foremost based on the incoming text data.

A new approach

Therefore, a new approach based on *frequent pattern mining* is presented to automatically derive filter policies from the incoming text stream. The challenge of frequent pattern mining was defined in (Aggarwal & Han, 2014, p. 1) as follows:

> Given a database D with transactions $T_1 ... T_N$, determine all patterns P that are present in at least a fraction s of the transactions. ... Each transaction can be considered a sparse binary vector, or as a set of discrete values representing the identifiers of the binary attributes that are instantiated to the value of 1.

In turn, this means the outcome of a frequent pattern mining algorithm is a set of patterns, where each pattern also consists of binary attributes. Comparing these patterns with heuristic filter policies as defined in the previous section, it becomes apparent that both are of the same form, i.e. heuristic and frequent pattern mining based algorithms yield filter policies of the same structure. Consequently, this means that the results from a frequent pattern mining approach are still understandable by human beings.

Now the algorithm itself is described. The idea is to discretize the feature values of an incoming document using a non-parametric method, i.e. the percentiles are used in order to classify each feature into a three categories: *low*,*regular*,*high*[15]. The classification of features is based on their membership to the *lower* or *upper* quantile or the 50% inbetween. Thus, each stream property s belongs to a class C with $C := \{c \in [low = 1, regular = 2, high = 3]\}$, except those, whose value is countable to 9, e.g. query matches. In order to create a unique id per stream feature type, each stream feature type is mapped onto a numeric class as well that is based on values using *base 10*, e.g. *number of query matches = 10*, *number of feedback matches = 20*, *document length = 30*, etc. Thereby, the feature class and the feature category can simply be added and a unique id for the feature is created, e.g. for an incoming document containing a number of query matches that belong to the upper quartile, the id for instance would be $3 = high + 10 = querymatches = 13$.

Concrete implementation

By repeating this process for every features, each incoming document was

[14] The evaluation of heuristic and automated filter policies will highlight this

[15] A non-parametric method has the advantage of not requiring information about the underlying distribution of the text stream feature.

Algorithm 2 Convert document into transaction

1: **function** CONVERT($document_d$)
2: **for each** stream feature sf of the $document_i$ **do**
3: **if** $sf_{value} \leq 9$ **then** $unit\_position \leftarrow count(sf_{value})$
4: **else** $unit\_position \leftarrow percentile(sf_{value}) \in [low = 1, regular = 2, high = 3]$
5: **end if** $tens\_digit \leftarrow sf_{class} \in [document\_length = 10, ...]$
6: $id = tens\_digit + unit\_position$
7: $transaction.add(id)$
8: **end for**
9: **end function**

converted into a *transaction* consisting a various discrete values. This transaction was then used as the input to a *frequent pattern mining* algorithm. A *frequent pattern mining* algorithm allows to identify frequent *itemsets* from a transaction database. The transaction database in this case in a continuous list of the transactions created by the algorithm 2. The critical point in *frequent pattern mining* is to select the optimal values for the *minimum support*, which is usually identified by *trial-and-error*. The volatile nature of a text stream prevents such an approach, thus an algorithm from the data mining library *SPMF*[16] was used that allows to generate a list of *top-k* frequent patterns from a transaction database. This means, instead of defining a minimum support upfront, the algorithm produces k patterns that are used as a filter policy.

Each *Scored Event* that surpassed its associated threshold was selected from the stream, converted into a transaction, and fed into the algorithm. The *FP-growth* algorithm that was provided by SPMF then generated rules from the given transactions based on the *antecedents* $\{11, 12, 13\}$, i.e. 1, 2 or 3 query matches[17]. Antecedents are preconditions of a rule, i.e. the precondition must be contained within a transaction before it is used by the algorithm (Witten & Frank, 2005, p. 65). The rationale for this is that a document must be related to at least one query match, in order to be potentially relevant.

For the evaluation runs a set of patterns was derived upfront that were used as a basic set of rules for the filter policy. Table 8.5 gives an example what *transaction ids* of a rule look like.

This leads to the following filter policy:

$$fp := 12 \wedge 23 \wedge 31 \wedge 62 \wedge 83 \wedge 122 \tag{8.1}$$

[16] `http://www.philippe-fournier-viger.com/spmf/`

[17] The class id for query match was chosen to be 10, thus one query match yields 11, two query matches 12 and three query matches 13. The maximum size of search string was three terms, thus only these three antecedents were required for the tests.

Feature	Class Id	Value	Transaction Id
Query matches	10	2	12
Feedback matches	20	3	23
Hashtags	30	1	31
Named entity	60	2	62
Nouns (upper percentile)	80	3	83
Verbs (regular percentile)	120	52	122

Table 8.5: Example of Frequent Pattern Mining based filter policy rules

Rule	Interpretation
{11,33,83,120,160}	11=One query match 33=Upper percentile document length 83=Upper percentile noun count 120=Zero numbers 160=Zero named entities
{12,20,40,63,83}	12=Two query matches 20=Zero feedback matches 40=No hashtags 63=Upper percentile spelling errors 83=Upper percentile noun count

Table 8.6: A few real-world examples of filter policies based on frequent patterns.

Finally, a brief extract from the generated rules is shown in table 8.6, in order to illustrate the results of the presented approach[18].

8.2.3.3 Preliminary document candidate selection based on weak classifiers

The third approach to implement a filter policy employs machine learning techniques. As shown in Chapter 7 it is possible to continuously gather various feature from a text stream. The analysis of the features also showed that these features are capable of reflecting the nature of the stream. Thus, these text features can be used as input for a machine learning algorithm. Machine

[18]Explanation of ids: query matches count= 10, feedback matches count= 20, documentlength= 30, hashtag count= 40, link count= 50, spelling errors= 60, stop word count= 70, noun count= 80, verb count= 90, lower case= 100, upper case= 110, number count= 120, positive sentiment count= 130, neutral sentiment count= 140, negative sentiment count= 150, named entity count= 160

learning and especially classification algorithms can be used to assign a class label to a document and thus provide a relevance decision for a document. But in this stage of the filter process a final attribution of a class label would be too early, because the score value, which is an important feature, has not been calculated yet. The importance of the score as a feature is due to the fact that it encodes the textual information of a document in terms of its relevancy based on a given scoring function.

Features and thresholds

Nevertheless, the remaining features already encode information about the document and the idea is that relevant documents share commonalities in their structure that a classifier can learn, although a strong feature like the document score is not yet available. Without such a strong feature, it is obvious that a classifier does not have the same discriminative power as a classifier, which includes these features. But a weak version can also be beneficial, in order to remove obviously irrelevant documents. The idea is to set the threshold of the classifier very low, so only the absolutely not relevant documents are filtered out by the classifier. For this evaluation the threshold, at which a document was considered "potentially" relevant, was determined dynamically using a non-parametric method. This means, the class probabilities of the relevant document class, which were generated by the learning algorithm, were collected and a histogram of the values was derived from the values in memory. The threshold was then determined by selecting the threshold at the $k - th$ percentile of the histogram. For lower frequency corpora like the TREC corpus, the value was set empirically to 0.1, while the value for the higher event density Events corpus was set to 0.5. Thereby the threshold adapted naturally over time.

For the purpose of the evaluation a *Random Forest* classifier[19] was chosen, because "[i]nstead of examining all features at every node to find the feature that gives the greatest gain ratio, only a subset of features is evaluated for each tree" (Janert, 2010, p. 419). This means each decision tree of the *random forest* represents a different interaction of features and therefore this type of algorithm seemed reasonable. Also *random forest* are supposed to model feature interaction well (Owen et al., 2011, p. 376), which was a desirable feature in this scenario, because thereby many different facets of a document can be captured by the learning algorithm.

Details on how the model was built and the model performance in general are discussed in the next section.

[19] In the courseof this dissertation the *JSAT* library `http://jsatml.blogspot.de/` was used. The random forest was initialized with 700 trees.

8.2.4 Evaluation of filter policy strategies

After introducing the various filter policy approaches, this section presents a detailed evaluation of the performance of each approach. The goal is to show the effect of each filter policy approach in detail and derive recommendations at the end of the section. The results will be discussed in terms of their over all performance, as well as on profile category and query level.

Remarks on the model building process and the experimental setup

Before the evaluation a short note on the model building process of the machine learning approaches is required, in order to clarify how the results were achieved and which factors influenced the performance.

In general, a major problem for any learning algorithms is the quality and the amount of the available data. If the training data are biased or of low quality, then the trained model and generated rules are also affected. Also, more training data for a machine learning algorithm is better. In this dissertation both aspects were problematic. First the relevance feedback assessments were sometimes doubtful (cf. 8.1.6) and also the volume of training data was limited. Nevertheless, both aspects were addressed as well as possible, in order to allow a general statement how well such approaches are suited for the filter policy stage. The quality aspect was addressed by using the relevance assessments from the TREC training for both TREC and Events corpora and both pattern mining and classifier-based approaches. This was because the assessments of the TREC corpus were carried out by domain experts while the events assessment was done semi-automatically.The limited training data size was addressed by applying thorough data pre-processing and feature engineering as proposed in (Kotsiantis & Kanellopoulos, 2006). Thereby it was assured that the quality of the model was as high as possible. The process included an exploratory analysis of the features including box plots and histograms as well as the removal of superfluous features by using correlation analysis and *principal component analysis* PCA. This process left ten features, which were used for building the classifier model[20]. The advice given (Domingos, 2012) was also taken into account, e.g. avoid overfitting by *k-fold* cross-validation, thorough feature engineering, or the use of ensemble methods like random forests.

Feature engineering

Additionally, it must be noted that all performance measures that are presented later on depend on the parameter settings that are used. For the top-k frequent pattern mining approach the value of k was set to 50. The thresh-

[20]The analysis script can be found in the online repository.

old that a document in the filter policy stage using the classifier had to exceed was – as mentioned in the previous section – based on non-parametric threshold selection method, i.e. the thresholds of all classified events were monitored and a document had to be among the 90% percentile of the scores. The intention of this non-parametric threshold is, first to be adaptable, and second to filter the documents that are judged as absolutely bad by the classifier. It must also be noted that other values, would lead to different results. However, the goal is not find the optimum parameter setting, but to verify the applicability and the performance of these approaches.

Experimental setup explained

Finally, the experimental setup was as follows. As baseline a simple heuristic filter was used. This filter policy required a document to match at least the number of search terms minus 1, e.g. for the topic "2022 fifa soccer" a document must have matched at least two terms. Then for comparison, the frequent pattern mining approach was added once as an additional filter policy and once the classifier-based policy. This means that the results that are shown in the next section, show how both non-heuristic approaches improve or degrade the performance in contrast to the baseline.

8.2.4.1 Overall level evaluation of TREC and Events corpus

In this section the results on a overall level for the TREC corpus and the Events corpus are presented. This means the results are not split up by profile category or by topic.

The results from the runs are summarised in table 8.7 for the TREC corpus and in table 8.8 for the Events corpus. Figure 8.1 and figure 8.2 show the results as *Kiviat graphs*. This type of graph allows plotting multivariate data within a single graph.

At the beginning, the absolute figures of true positive, false positive, etc. are discussed and then the summarising performance indicators like precision, recall or f1-measure. First a look at the *true positive* rates are taken. These rates are consistent in terms of their trend for both corpora, i.e. the baseline/heuristic filter policy yielded the most true positives for both corpora, followed by the frequent pattern approach and the classifier-enhanced policies. This was to be expected as the heuristic baseline is a very loose filter policy that aims only at removing the least amount of relevant documents by only applying a simple term filter and thus letting pass many documents to the later filtering stages.

If the percentage of change between the approaches is studied the picture changes. The frequent pattern mining approach yields 28.78 % less true positives for the TREC corpus and 38.91% for the Events corpus. While this is

Filter Policy	Baseline	Frequent pattern	Classifier
Recall	0.3765	0.2699	0.1169
Precision	0.2459	0.2621	0.3108
F1	0.2401	0.2436	0.2040
T11SU	0.1873	0.2045	0.2786
Specificity	0.4478	0.4544	0.3494
Sensitivity	0.6617	0.6564	0.7703
True Positive	702	507	223
True Negatives	10384	7345	1698
False Positive	8629	6252	1333
False Negative	480	352	86

Table 8.7: High-level evaluation of filter policies applied on TREC 2011 corpus.

Filter Policy	Baseline	Frequent pattern	Classifier
Recall	0.3661	0.2562	0.1634
Precision	0.1245	0.1108	0.1419
T11SU	0.0816	0.0754	0.1473
Sensitivity	0.6609	0.6300	0.7033
F1	0.1208	0.1033	0.1125
True Positive	3115	1903	1518
False Positive	89853	82328	26114
False Negative	1804	1389	825

Table 8.8: High-level evaluation of filter policies applied on Events 2012 corpus.

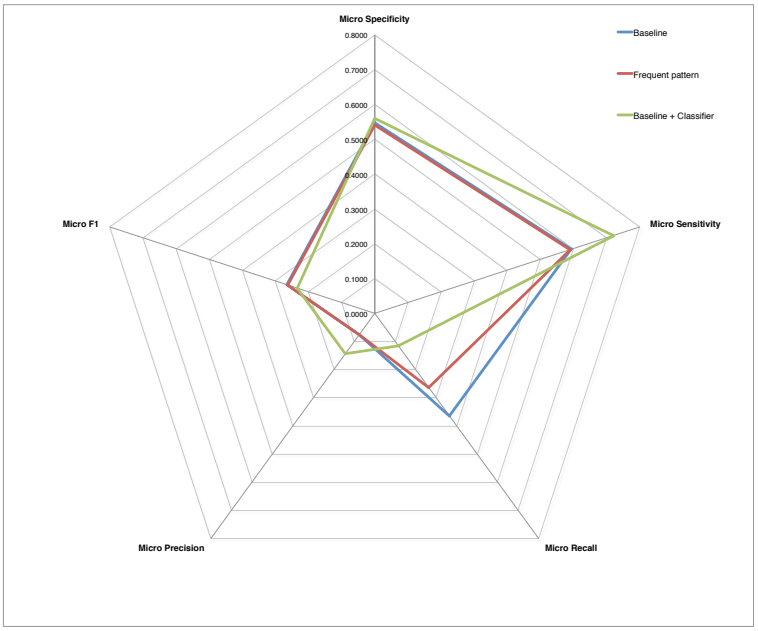

Figure 8.1: Kiviat graph summarising the performance measures of different filtering policies for the TREC corpus.

a 10 percent difference this seems acceptable, as both approaches have left a decent number of true positives. More interesting is the the reduction of true positives by the classifier-enhanced approach. For the TREC the classifier-enhanced filter policy resulted in 68.23% less true positives than the heuristic baseline. For the Events corpus the classifier approach returned 51,77% fewer true positives. It seems that in a text stream with higher event density, such as the Events 2012 corpus compared to the TREC corpus is, the classifier-based approach seems to let pass a reasonable amount of true positives. For low-density text streams this approach seems be too restrictive.

True negatives were only available for the TREC corpus, as the Events corpus did not provide any negative relevance judgements. The frequent pattern approach reduced the number of true negatives by 29-27%, which corresponded to the rate of the true positives reduction. It is interesting to see that the true negative rate for the classifier approach dropped by 83.65%. This figure becomes interesting when studied with the changes of *false positives* and *false negatives*. The classifier-based approach also decreases both numbers by around 82-83%. The effect for the Events corpus is not as significant, but the values are decreased by 71,94% for the false positives and 54.27% for the false negatives. The interesting fact is the over-proportional decrease of false positives in contrast to the reduction of true negatives, i.e. the classifier approach removes more irrelevant documents from the stream than relevant ones. This is not true for the frequent pattern approach. The reduction of documents for

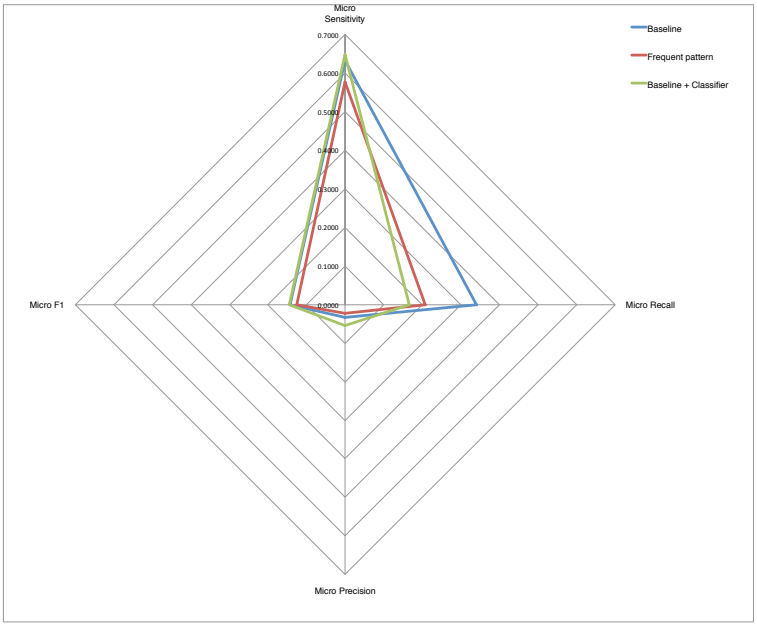

Figure 8.2: Kiviat graph summarising the performance measures of different filtering policies for the TREC corpus.

each of the categories has the same proportion (around 30%), while for the Events corpus the reduction of true positives is over-proportional in contrast to the reduction of false positives. This is not a desirable results.

This findings support the notion that heuristic and frequent pattern approaches work equally well for low-density text stream, while the classifier-based approach removes too many documents. For a high-density text stream the classifier approach seems more promising.

Next, the performance indicators precision, recall, f-measure, sensitivity and specificity on a macro-level are reviewed. For the TREC corpus precision raised by 6.58% with the frequent pattern mining approach and by 26,38% with the classifier approach. However, recall dropped by 28.32% for the frequent pattern approach and by 68.95% for the classifier approach. As pointed out earlier precision is more important than recall, as there are usually enough relevant Tweets and documents. Sensitivity, being based on true positives and false positives, reflects the previously described results. The classifier-enhanced approach had the highest sensitivity values. The frequent pattern mining approach had approximately the same performance as the heuristic approach. Specificity is reported only for reasons of completeness, but in practice this value is not very useful as the number of true negatives is always very high and thus the value always close to 1 (Schütze et al., 2009, p. 162).

In summary, the heuristic approach and the frequent pattern mining ap- Summary,

proach are well suited in a low-density text stream. The heuristic filter policy can be replaced by a frequent pattern based filter policy in case it is not feasible to maintain the policies manually. For high-density text streams, the classifier-based approach seems preferable as it efficiently removes false positives while still letting true positives pass. This is another reason to use a classifier-based filter policies.

8.2.4.2 Evaluation on temporal profile level

The overall figures gave a first impression as well as a general advice on when to use which filter policy. But this level is too coarse-grained, because it does not reveal how each approach performs on the different profile categories. This is important to investigate, because one approach can be the most effective for a specific profile category while not being in the first place in the overall rating, i.e. it is necessary to determine which approach fits which scenario.

This section discusses the TREC corpus results. The results for the Events corpus can be looked up in C. First table 8.9 summarises the result of the relevant performance indicators for all three profile types.

Atemporal search profiles

Table 8.10 shows the result for the atemporal, category 1 topics. Again the classifier-enhanced baseline has the highest precision values, but only by 3 to 4 % compared to the other two approaches. But the *false positives* as well as the *false negative* values are far better than the ones from the other approaches. Again the overall impression is supported that the classifier approach is the most precise, but yet it is also too strict, while the frequent pattern mining approach yields the same results as the heuristic approach and thus both are equally suited as the filter policy to use for the further stages of this analysis.

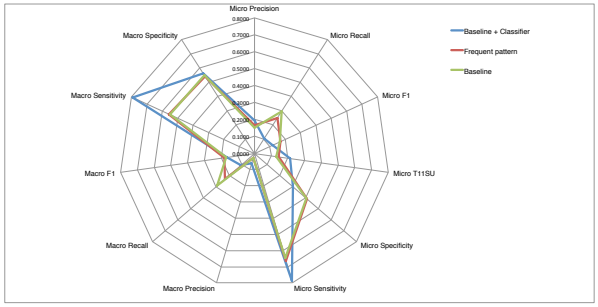

Figure 8.3: Kiviat graph summarising performance for filter policies on atemporal topics.

Temporally unambiguous queries

Table 8.11 summarises the results for the temporally unambiguous, category 2 topics. It is interesting to see that all approaches were able to identify many more relevant documents from the stream than for the category 1 topics. Also

	Atemporal	Temporally unambiguous	Temporally ambiguous
Precision			
Classifier	0.1956	0.3853	0.3953
Frequent Pattern	0.1672	0.3553	0.2880
Baseline	0.1535	0.3509	0.2523
Recall			
Classifier	0.1039	0.1378	0.1255
Frequent Pattern	0.2480	0.2448	0.2852
Baseline	0.2964	0.4197	0.3872
F1-Measure			
Classifier	0.1172	0.2578	0.2289
Frequent Pattern	0.1697	0.3021	0.2608
Baseline	0.1641	0.3051	0.2484
Sensitivity			
Classifier	0.7877	0.7724	0.7480
Frequent Pattern	0.6639	0.6750	0.5896
Baseline	0.6460	0.7029	0.5874
Specificity			
Classifier	0.3011	0.3855	0.4340
Frequent Pattern	0.4118	0.4836	0.4779
Baseline	0.4062	0.4865	0.4557

Table 8.9: Evaluation of filter policy performance on temporal profile level for the TREC 201 corpus.

Policy	Classifier	Frequent pattern	Baseline
Topic Count	11	12	12
True Positive	69	150	188
False Positive	1050	4894	6441
False Negative	18	120	153
True Negative	1357	5815	7844
Precision	0.1956	0.1672	0.1535
Recall	0.1039	0.2480	0.2964
F1	0.1172	0.1697	0.1641
T11SU	0.2147	0.1439	0.1304
Specificity	0.3011	0.4118	0.4062
Sensitivity	0.7877	0.6639	0.6460
Total Relevant	638	647	647
Total Non-relevant	10502	11523	11523

Table 8.10: Performance measures for atemporal topics of the TREC corpus.

the precision values are all closer than for category 1 (and category 3), which leads to the assumption that for this type of topics all three approaches are equally suited when precision is the performance measure that should be optimized. The recall values for the machine learning approaches were also only slightly higher than for the category 1 topics while the heuristic approach yielded a significantly higher recall value. The reason for the good precision values is probably due to the fact that microblogs like Twitter are often used to discuss recent issues and events. Thus the feature density is much higher than for profile types one and three and thus it becomes easier for the filter policies, because many more potentially relevant documents than non-relevant are available while a topic is discussed. This assumption is supported by the significant increase of precision from profile one to profile two.

Temporally ambiguous queries

Finally, table 8.12 shows the result for the category 3 topics, the temporally ambiguous ones. First the results in term of macro averages are studied. Here the gap in terms of precision values is again larger. The gap between recall values for the frequent pattern mining approach and the heuristic approach decreased compared to category 2 topics. On the contrary, the spread of the recall values is also large. This time the baseline approach consequently

Policy	Classifier	Frequent pattern	Baseline
Profile Category	2	2	2
Topic Count	14	14	14
True Positive	88	197	305
False Positive	190	873	1279
False Negative	41	118	171
True Negative	224	981	1492
Precision	0.3853	0.3553	0.3509
Recall	0.1378	0.2448	0.4197
F1	0.2578	0.3021	0.3051
T11SU	0.3247	0.2680	0.2619
Specificity	0.3855	0.4836	0.4865
Sensitivity	0.7724	0.6750	0.7029
Total Relevant	810	810	810
Total Non-relevant	11225	11225	11225

Table 8.11: Performance measures for temporally unambiguous topics of the TREC corpus.

yielded the best recall values, as it is the least restrictive approach. These large recall values also had enough influence on the f1-measure to turn the baseline approach into the most effective approach in terms of precision/recall trade-off.

Summary

In summary, the impression from the overall evaluation is supported, because the classifier-enhanced approach yielded the most precise results while being the most restrictive one that removed too many documents for the later stages. It also showed that for temporally unambiguous topics the classifier-based policy approach has lesser influence in terms of precision and recall than on the other two topic categories. For these two, the difference in performance values comparison between the classifier-enhanced and the other two filter policies approaches differ significantly.

8.2.5 Conclusion

This section studied how different filter policies performed in terms of standard information filtering performance measures in various levels of detail. The baseline run used a heuristic filter policy. Then frequent pattern mining and a classifier-based approach were added. The frequent pattern min-

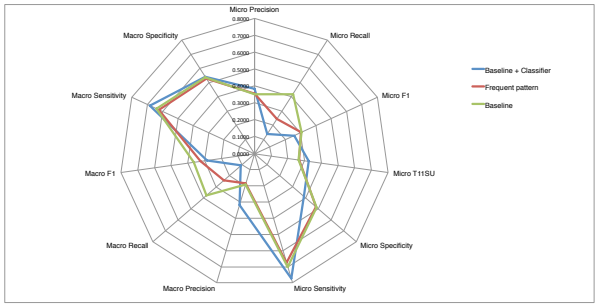

Figure 8.4: Kiviat graph summarising performance for filter policies on temporally unambiguous topics.

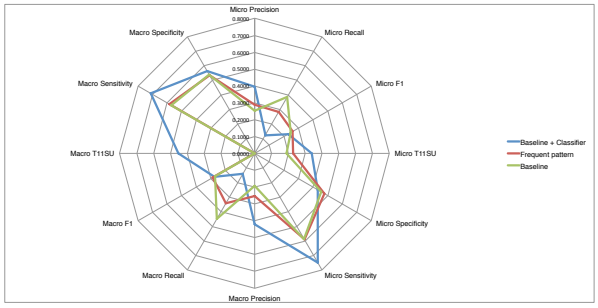

Figure 8.5: Kiviat graph summarising performance for filter policies on temporally ambiguous topics.

ing approach slightly increased the precision, but decreased the recall value remarkably. The classifier-based approach also decreased the recall value, however it significantly increased the precision value.

The frequent pattern mining approach seems promising in terms of the rules created by the algorithm, i.e. they are human intelligible, take into account many facets of a document and could be therefore a sound base to improve or enrich heuristic filter policies. But they suffer from a performance problem compared to the heuristic and the classifier-based approach. This is probably due to the available training data. Moreover cleaner training data would be required to study the real potential of this approach. The weak classifier approach using an adaptive threshold yielded good results in terms of precision. But the recall values were low for the TREC corpus. For the Events corpus, this was not the case, and the classifier-based approach yielded the best recall values along the best precision values. It seems that the higher event density was beneficial in this case, as there were more documents that could pass the classifier-based filter policy.

In summary, the classifier-based approach is obviously the best suited approach in high-density text stream scenarios. In text streams with less event density, it probably removes too many documents.

Policy	Classifier	Frequent pattern	Baseline
Profile Category	3	3	3
Topic Count	10	10	10
True Positive	66	159	208
False Positive	91	467	875
False Negative	27	114	156
True Negative	117	537	1027
Precision	0.3953	0.2880	0.2523
Recall	0.1255	0.2852	0.3872
F1	0.2289	0.2608	0.2484
T11SU	0.3403	0.2292	0.1885
Specificity	0.4340	0.4779	0.4557
Sensitivity	0.7480	0.5896	0.5874
Total Relevant	465	465	465
Total Non-relevant	7738	7738	7738

Table 8.12: Performance measures for temporally ambiguous topics of the TREC corpus.

8.3 Scoring in dynamic text streams

The next component that is studied is scoring and term weighting. Term weighting is essential, because the attributed weight of a term, or in general a document component – like links for instance –, reflects the importance of a term or component and thereby influences the overall score of the document. A document score is calculated by using a scoring function that builds an overall score, using and combining the score value of the single components in a specific way. Often this combination defaults to the generally accepted approach of linear feature-based models, i.e. the "linear combination of features" (D. Metzler & Bruce Croft, 2007, p. 257). Features can be term weights or weights of other document components, which are combined with a parameter vector Θ. (D. Metzler & Bruce Croft, 2007) studied how to find the optimal parameter setting for Θ. This is not the goal here, i.e. the parameters are not optimized as the goal is not build an optimal filtering system, but to evaluate the construction and combination of features for the linear scoring function and to measure their influence on standard evaluation indicators. Furthermore, it must be noted that the scoring functions that are studied here, all produce monotonically increasing score values, i.e. the scores

produced by a given term weight are non-negative, a higher score indicates higher importance and all scores are summed up to form a final score. This is also important to remember for the study of thresholds in the next section.

As discussed in section 3.3.2 various approaches exist that can be used for term weighting. But many of these have been studied mainly in the context of static information retrieval. There are also a few weighting schemes that are directly focused on dynamic text stream data (cf. 3.3.2.5), but a comprehensive study of various term weighting schemes for information filtering in high-volume text streams is missing.

Four new term weighting schemes

Therefore, the goal of this section is to study a variety of different term weighting schemes in the context of dynamic sub-corpora, and to investigate which term weight scheme works best. In addition, four term weighting schemes are proposed and investigated, which intend to leverage stream characteristics that can be monitored using the proposed reference model and event based architecture. This section will show how these term weighting schemes perform. Furthermore, the term weighting schemes are studied again in terms of the macro averaged performance measures as well as the three introduced profile types.

The case for the evaluation

As mentioned previously, only few of the discussed term weighting schemes have been studied in the context of high-volume dynamic text streams. The dynamic sub-corpus approach using sliding windows is of particular interest, as it has a major effect on the corpus statistics. Furthermore, it is interesting that many term weighting schemes rely on a local factor, which is in general some variation of the *within-document term frequency* (cf. 3.3.2). But as short documents like Twitter or headlines usually only contain terms once, this factor in fact becomes constant for all terms. Nevertheless, in the context of short document term weighting and foremost Twitter many studies "just" take the approaches from information retrieval without modification; these are then applied one-to-one to dynamic text streams (cf. (Elsawy et al., 2014, p. 2043), (Albakour et al., 2013, p. 421) or (Shraer et al., 2013, p. 386)), even though the expected term frequency is one and thus constant (cf. (Naveed et al., 2011, p. 2)).

In summary, the main goals of this section are :

1. Apply the most common term weighting schemes within the proposed dynamic sub-corpus context.

2. Compare the various term weighting schemes in terms of their performance on a macro and profile type level.

3. Investigate and compare the performance of four new term weighting approaches that try to leverage the streaming and event processing setting.

8.3.1 Overview of studied term weighting schemes

The goal of this section is to investigate how different term weighting schemes work in a high velocity text stream scenario using limited memory. For this, a number of popular term weighting schemes was evaluated. Additionally, four new approaches are introduced that leverage characteristics of the text stream or combine features in a new way. These are also evaluated in terms of their performance on the corpora. A comparison with a static search engine was also made. For this the documents of the two corpora were indexed using *Apache Lucene*[21] and scored in a classic information retrieval manner using the standard Lucene scoring facilities. This means, the incoming documents were scored against the complete available corpus without any temporal limitations. The goal of this approach was to compare the performance of dynamic sub-corpora to a static corpus that contained the complete available document information. If the performance of the static, Lucene based approach is similar to the one of the dynamic sub-corpora, this can be seen as another indicator that the dynamic sub-corpus approach is valid.

The following term weighting schemes for scoring were studied:

- tf.idf
- idf.only
- df.idf
- Okapi BM25
- Language models
- Divergence from randomness[22] (PL2 version)
- Divergence from randomness (Bose)
- Divergence from randomness (Lnl2)
- *BursT*
- Boolean
- Apache Lucene based term weights
- Context based term weights (new)
- Poisson based term weights (new)

[21] `https://lucene.apache.org`

[22] Details on the divergence of randomness framework as well as the used models can be found at `http://terrier.org/docs/v3.5/dfr_description.html`. Last accessed on 13.11.2015

- Harmonic mean meta score-based term weights (new)[23]
- Arithmetic mean meta score-based term weights (new)
- Geometric mean meta score-based term weights (new)
- Harmonic mean document score-based term weights (new)[24]

8.3.2 New term weighting algorithms for text streams

This section presents new approaches to term weighting for short documents in high-volume text streams. As mentioned, the problem is often the local factor of a document due to the brevity of Tweets, comments, commit logs, short message or posts on social media platforms. Therefore the following four approaches try to overcome this limitation using different approaches.

8.3.2.1 Poisson distribution based term weighting

The idea for this term weighting scheme is to replace the local part of a scoring function, which usually derived from the local term frequency, with a value that is based on the *Poisson distribution*. The Poisson distribution is used to model the probability of an event e to occur k times given a certain expectation λ that e really occurs x times. For instance, the distribution can used to calculate the probability that n customers enter a shop within a certain period of time (Schaich et al., 1993, p. 126ff). This example already illustrates why this distribution is interesting in an event-driven scenario. The customer example can be easily applied on terms within a text stream and then the Poisson distribution can be used to calculate the probability of a term occurring x times. Formula 8.2 shows the general formula to calculate the probability of an event to occur x times given the expected occurrence rate λ.

$$P_{Poisson}(x;\lambda) = \begin{cases} \frac{\lambda^x}{x!}e^{-\lambda} & \text{, if } x = 0,1,2,\ldots;\lambda > 0 \\ 0 & \text{, otherwise} \end{cases} \tag{8.2}$$

The two required parameters, x and λ, are easily obtainable using the event-driven approach. A stream statics event processing agents monitors the average arrival time of each term. Another event processing agent, which is responsible for updating the index of the dynamic corpus, can provide the concrete count for each term. Using the cumulative distribution function (CDF) it is possible to calculate the probability that a term t occurred at most

[23] ID in the evaluation tables: HarmonicMeanScorer
[24] ID in the evaluation tables: HarmonicMeanScorer2

k times. For instance, if a term occurred 3.4 times per average and in the current sliding window it occurs $five$ times, the CDF for the Poisson distribution that term t occurs five times or less is 0.8705.

For the local part of the term weighting scheme the value $P_{Poisson}$ is used as follows:

$$\text{local\_term\_weight} = \exp^{-\log_{10}(1-P_{Poisson})} \tag{8.3}$$

As a probability always takes a value between one and zero, the logarithm is also always negative, therefore the negative logarithm is used in the formula to make the value for the exponent positive. The logarithm with base 10 is used, in order to avoid extraordinary large values for very rare events, because these would spoil other statistics. Furthermore, the value $P_{Poisson}$ is subtracted from 1, which yields the probability of terms occurring more than k times. Finally, this values is used as the exponent to base e. e is taken as it is a well-studied number in mathematics and is often used to describe exponential growth in nature; however, another base would work also. Finally, an example should illustrate the numbers that are calculated using this method. For instance, if the $P_{Poisson}$ is 0.75 the formula yields 1.8224, for $P_{Poisson} = 0.90$ the formula yields 3.659 and for $P_{Poisson}$ the value already becomes 7.344.

8.3.2.2 Context augmenting stream feature weighting

This weighting scheme tries to address the sparsity of content words by using not only the score of the query matches, but also the information about links, hashtags, spelling errors and the ratio of content and stop words. The idea is, again, that high quality documents do not only use the same terminology, but also share common document properties. In fact, this term weighting is an untrained version of a regression model, i.e. there is no parameter vector v that was learnt by using training data and adjusting the components. A further extension of this model would be to learn a parameter vector, but the goal here is to show, if the context parameters themselves already improve the scoring over a baseline. This comparison – in general the comparison of all term weighting schemes – helps to identify the term weighting schemes that yield the best scores to be used in machine learning models. The formula used is a follows

$$context(d) = \sum score(querymatches) + \log(IDF_{hashtags}) + \log(IDF_{links}) + \log(\frac{(contentword + 1)}{stopword + 1}) + \log(\frac{(characters + 1)}{spellingerrors + 1}) \quad (8.4)$$

The log is made in order to accommodate outliers. A similar approach to scoring using context features was used in (J. Zhang et al., 2012), but they used an upfront trained model and different context features. But most of all, they did not rely on the statistics from the surrounding sub-corpus, but used a static corpus. In some ways this can be also considered as a rather simplistic version of Google's approach to incorporating many different "signals" into the scoring scheme (B. Dean, 2015) and it would be interesting if more contextual parameters could be attributed to inflowing documents without delaying the processing[25]. The evaluation in the next section shows how well the proposed Context Scorer works.

8.3.2.3 Meta-scoring using means

This section studies two approaches to document scoring that use the harmonic mean as a meta-algorithm for calculating the final score. The harmonic mean "is defined as the reciprocal of the arithmetic mean of the reciprocals of the values" (Kenney & Keeping, 1954, p. 57) and the formula to calculate the harmonic mean is:

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \quad (8.5)$$

The main rationale for the usage of the harmonic means is twofold. The first is that it is not prone to outliers (Matuszak, n.d.) and, second, "it does not suffer from the flaw of misweighting the data points" (Matthews, 2004, p. 217). These properties are used in two ways. First is a intra-document based scoring using this mean, i.e. all terms of the document are scored and their scores are averaged using the harmonic mean. This could be beneficial, because by taking all terms of a document into account, other important terms – in terms of their score – are also incorporated, but do not dominate the overall score for the document due to the harmonic mean. Second, a inter-scoring-

[25] As mentioned earlier evidence, in form of URLs improves the performance significantly. An upfront modelling of the search profile and its constituting search profiles using e.g. *WordNet* can also have positive aspects. But yet again the goal is to study the single components in their purest form, in order to obtain a starting point for further improvements.

function value is calculated. This is in ways similar to ensemble methods in machine learning. This approach “refers to a set of techniques for improving accuracy by combining the results of individual or "base" classifiers” (Janert, 2010, p. 419), i.e. several classifiers run independently and by combining the classifiers the errors of each classifier should cancel out (p. 419). The same approach is taken here. Several of the mentioned classifiers are run in parallel and the final score is calculated by using the harmonic mean of the single scoring results. When this scoring scheme is applied, it must be taken into account that this affect the performance, because several scoring loops are required.

8.3.3 Performance evaluation of term weighting schemes

This section evaluates the reference term weighting schemes as well as the proposed ones in the known manner. First the overall performance is shown, followed by a more detailed look and analysis of the performance, where a term weighting scheme yields significantly different results from the other schemes.

For the evaluation *StreamFI* was configured to use the same threshold strategy (average score) as for the filter policy runs. Furthermore, the heuristic baseline filter policy was used, as it does not remove too many documents. This is reasonable for this purpose, because the number of relevant documents is limited per topic and otherwise an evaluation with too few relevant documents would not be meaningful. Also this policy is static and easier to reproduce. The results for the events corpus are in appendix D.

8.3.4 Overall evaluation of term weighting schemes

In this section an overview of the results is presented, in order to get an impression of the performance of various term weighting schemes. All results were collected using the same system conditions, thus they are comparable. In the next few sections the results are discussed in more detail. Table 8.13 shows the results for investigated term weighting schemes sorted by the F1 measure.

Detailed analysis of term weighting schemes

This section discusses the overall results on different levels of detail. First the summary performance indicators like precision and recall are discussed. Then, a closer look on the absolute figures of true positives, false positive,

Term weighting	TP	FP	FN	TN	Precision	Recall	F1	T11SU
Harmonic mean scoring functions	581	6669	463	10198	0.3295	0.3011	0.2913	0.2687
Terrier Pl2	655	7069	389	9798	0.3230	0.3316	0.2929	0.2571
Regular Okapi	653	7038	391	9829	0.3229	0.3307	0.2929	0.2570
Harmonic mean terms	513	7485	531	9382	0.3136	0.2624	0.2653	0.2444
tf.idf	540	6227	504	10640	0.3005	0.2751	0.2672	0.2507
TerrierInL2	444	8742	600	8125	0.2916	0.2207	0.2445	0.2648
IDFOnly	705	6423	339	10444	0.2886	0.3529	0.2875	0.2559
Lucene Okapi	727	7518	317	9349	0.2763	0.3659	0.2841	0.2432
Geometric mean scorer	401	8012	643	8855	0.2643	0.2051	0.2465	0.2800
Language model	469	6173	575	10694	0.2636	0.2257	0.2483	0.2470
Terrier DFR Bose	701	6561	341	10305	0.2633	0.3117	0.2802	0.2434
Boolean	291	3526	753	13341	0.2632	0.1505	0.2672	0.2952
Context scorer	691	10874	353	5993	0.2576	0.3676	0.2643	0.2280
Arithmetic mean	299	7642	745	9225	0.2226	0.1566	0.1933	0.2744
df.idf	308	7881	736	8986	0.2041	0.1549	0.1813	0.2659
Probability based	273	8088	682	8582	0.1522	0.1424	0.1522	0.2456
Burst	138	2101	230	4569	0.1504	0.0921	0.1318	0.1977
Poisson	219	7660	736	9010	0.0813	0.1180	0.1348	0.2560

Table 8.13: Overall evaluation of term weighting schemes applied on TREC 2011 corpus.

true negatives and false negatives is taken, in order to get a better view of how each term weighting schemes performs in each scenario. Then, a look on the performance on a temporal profile level is taken and finally the best performing method is compared with the standard Okapi BM25 model. The results for the Events corpus can be found in appendix D.

Precision

First, the precision is investigated. This is an essential performance indicator, because it shows, how "precise" the result are and this is important in a scenario of high-volume text streams. There are three term weighting schemes that perform nearly the same in terms of precision: the new proposed Harmonic Mean Scorer that averages several individual scoring schemes, the divergence from randomness PL2 from the Terrier framework[26] and the renowned Okapi BM25 mechanism. All these schemes score nearly the same. Close behind is the second version of the harmonic mean scorer that averages all document components. This harmonic mean scorer version uses the Okapi BM25 as the base scorer, so the results show that using all the components of the document does not yield any superior results in an overall evaluation, but requires a closer look at the performance within the different temporal profiles.

Corpus wide scoring performs worse

Then several term weighting schemes yielded precision values between 0.3 and 0.25. It is notable that the geometric mean scorer does remarkably worse than the harmonic mean scorer, which underlines that the harmonic mean is the better choice in this setting. Also notable is that the precision of the Lucene Okapi scoring scheme is significantly worse than the performance of the leading four scoring schemes. This underlines the claim that sub-corpora contain sufficient information to perform information filtering task. Furthermore, it is remarkable that the IFDOnly approach also performs significantly worse than the leading term weighting schemes. As mentioned, the local term frequency, which is used by the local term weighting component of the leading formulae, was replaced by a sub-corpus wide term frequency due to the sparsity of terms within short documents. But this sub-corpus wide term frequency is also contained in the idf value that forms the global term weighting component. Thus, the transformations that are applied by the TerrierPL2 or Okapi BM25, such as document normalisation, play a crucial role in improving the quality of a scorer. This becomes more significant when their values are compared to the df.idf term weighting scheme, which only yielded a precision of 0.2041. The proposed Context Scorer was also among

[26] `http://terrier.org/`

the moderate performing term weighting schemes. It yielded an overall performance of 0.2576, but the evaluation of the remaining performance indicators showed other qualities of this term weighting scheme, foremost in terms of recall.

The final four term weighting schemes yielded precisions values between 0.2 and 0.08. Notable in this scenario is that the idea of using the Poisson distribution as a replacement for the local term weighting component failed completely. Also the *BursT* algorithm, which was explicitly designed to address the dynamic nature of text streams, also failed. And finally, a plain probabilistic classifier, i.e. only adding the log probability of each term, also performed badly.

In summary, from an overall perspective the Okapi BM25, the PL2 and the harmonic mean of scoring functions performed best. A closer look at the remaining performance indicators as well on the temporal profiles can provide more insight into the performance and help to select the most appropriate weighting scheme.

Recall

When discussing the results from the recall point of view, the order of the best performing term weighting schemes changes. Now the Context Scorer, Lucene Okapi and IFDOnly performed best with recall values above 0.35. Here, the knowledge of the complete corpus was beneficial for the Lucene Okapi scheme, but the sub-corpus based Context Scorer and the IFDOnly approach yielded the same performance. Thus, this is again an argument for the validity of dynamic sub-corpora. The next five term weighting schemes that yielded a recall value over 0.3, included the four leading schemes from the precision evaluation. This illustrates that the high precision term weighting schemes also provided a decent recall value performance. Several term weighting schemes yielded recall values between 0.3 and 0.2, among these renowned schemes like tf.idf or language models. And yet again the Poisson based approach, the probability based approach and BursT based schemes performed worst.

8.3.5 Evaluation of temporal profile level

In this section, the various term weighting schemes are compared in terms of their performance on the three previously introduced temporal profiles. The goal is to show which term weighting scheme works best for which search profile type.

Atemporal search profiles

Precision

Table 8.14 shows the precision values for the studied term weighting schemes. The most obvious observation was that the precision performance for all term weighting scores was not good. The best performing term weighting scheme, Boolean, only yielded a precision of 0.1791, and the first ten of the list were within a range of 0.1791 to 0.1552. This showed that none of the term weighting schemes performed significantly better than the rest for the atemporal search profiles. The reason for this is probably due to the nature of atemporal search profiles. Relevant documents do not arrive in bursts but are spread evenly over time. If a burst occurs than it is probable not comparable to a *real* burst of for instance a breaking news event. Thus, a temporal component is not present that could help to distinguish the quality of terms by providing higher term scores due to higher term frequencies or denser term arrival. A look at the score distribution, which is presented later on, helps to understand the reasons for this.

Statistical details

In terms of recall, the picture changed. Table 8.14 contains the recall values. Context Scorer, IDFOnly and Lucene Okapi achieved a precision of greater than 0.3. Afterwards the recall values dropped steeply. The context information used by the Context Scorer seemed to add some more information, as it yielded slightly better results than the subsequent two ones. In general, this coarse grained summary did not allow identifying why one term weighting scheme yielded a higher recall value than another one, and more specifically whether the difference between the best performing and the worst performing was due to a real trend or caused by a few outliers with very high recall values. But before a more detailed analysis is performed to figure this out, a brief look at the absolute values of true positives, false positives, etc. is taken. Table 8.15 shows these values. The first four term weighting schemes are very similar in terms of their performance. Only the Context Scorer produced a large number of false positives and yielded less true negatives. The values for the remaining three top performing term weighting schemes is very similar in terms of absolute values. This also supports the assumption that the differing precision and recall values were due to outliers and happened more by chance.

Box plots

In order to verify this the spread of the score, precision and recall values of the atemporal search profiles was analysed using box plots[27]. "Box plots combine many different measures of a distribution into a single, compact graph" (Janert, 2010, p. 37). The box is called the *inter-quartile* range and

[27] A similar approach to compare multiple performance measures but applied to information retrieval components was used in (Kürsten, 2012).

Term weighting scheme	Precision	Recall	F1
ArithmeticMeanScorer	0.1493	0.1131	0.0661
Boolean	0.1791	0.1344	0.1808
Burst	0.0879	0.0947	0.0846
ContextScorer	0.1382	0.3114	0.1479
DfIdf	0.1171	0.1254	0.0686
GeometricMeanScorer	0.1352	0.1469	0.1075
HarmonicMeanScorer	0.1764	0.2367	0.1687
HarmonicMeanScorer2	0.1721	0.2196	0.1603
IDFOnly	0.1552	0.3056	0.1683
LanguageModel	0.1583	0.1668	0.1320
LuceneOkapi	0.1626	0.3055	0.1757
Poisson	0.0575	0.0953	0.0528
ProbScorer	0.0940	0.1381	0.0776
RegularOkapi	0.1672	0.2769	0.1691
TerrierInL2	0.1555	0.1660	0.1214
TerrierDFRBose	0.1357	0.2393	0.1498
TerrierPL2	0.1673	0.2784	0.1692
TFIDF	0.1683	0.2298	0.1616

Table 8.14: Precision values of different term weighting schemes applied on the TREC 2011 corpus for atemporal search profiles.

Scoring function	TP	FP	FN	TN
LuceneOkapi	204	5924	118	8113
ContextScorer	203	9034	119	5003
TerrierDFRBose	200	4940	122	9096
IDFOnly	198	4854	124	9183
TerrierPL2	179	5655	143	8382
RegularOkapi	178	5630	144	8407
HarmonicMeanScorer	149	5460	173	8577
TFIDF	141	5024	181	9013
HarmonicMeanScorer2	139	6177	183	7860
TerrierInL2	134	7731	188	6306
GeometricMeanScorer	119	7203	203	6834
LanguageModel	117	5109	205	8928
ProbScorer	115	7249	199	6763
DfIdf	103	7041	219	6996
Boolean	94	3190	228	10847
ArithmeticMeanScorer	91	6867	231	7170
Burst	78	1614	78	3939
Poisson	75	6915	239	7097

Table 8.15: TP, FP, TN, FN values of different term weighting schemes applied on the TREC 2011 corpus for atemporal search profiles.

spans 50% of the values. The marker in the box represents the median and the whisker at the top and at the bottom represent the tails of the distribution of values (p. 36), i.e. they span the upper 25% and the lower 25% of the values respectively. Dots outside of the whisker represent outliers. For this analysis the box plots were created using the *R* package *ggplot2*. The configuration of these box plots was defined in (Wickham, n.d., para. 4) as follows:

> The upper whisker extends from the hinge to the highest value that is within 1.5 * IQR of the hinge, where IQR is the inter-quartile range, or distance between the first and third quartiles. The lower whisker extends from the hinge to the lowest value within 1.5 * IQR of the hinge. Data beyond the end of the whiskers are outliers and plotted as points (as specified by Tukey).

Due to their compact representation they are well suited to compare different distributions, and as more than 10 term weighting schemes were to be compared, this representation was chosen. (Kürsten, 2012) also evaluated a multitude of term weighting schemes for static corpora by using this visual representation.

Figure 8.6 shows the box plots of the precision values for the single search profiles. It is apparent that all term weighting scheme fail to represent a clear trend for superior precision performance, because almost all term weighting schemes – except the ones performing badly - have a wide spread from 0.0 precision to around 0.3. So the only thing that really becomes apparent is that the Poisson approach, as well as the ProbScorer fail completely. BursT does also not perform well.

Figure 8.7 shows the recall values for the search profiles of the atemporal category using a box plot. Here, the picture is a bit clearer in the sense that the performance of each term weighting scheme varies stronger than the precision values. Reviewing the best three recall term weighting schemes, it becomes apparent that they all have long whiskers, which indicates a large spread of recall values over the various search profiles. Therefore, as with the precision values, the better performance of one term weighting scheme over the other, seems to be more based on chance than on the quality of the term weighting scheme itself. This becomes more apparent when looking at the overall score distribution.

Figure 8.8 shows the box plot for the scores. The scores are within relatively close ranges for all term weighting schemes; except, yet again, for the ones performing badly. This suggests that the scores are uniformly distributed and

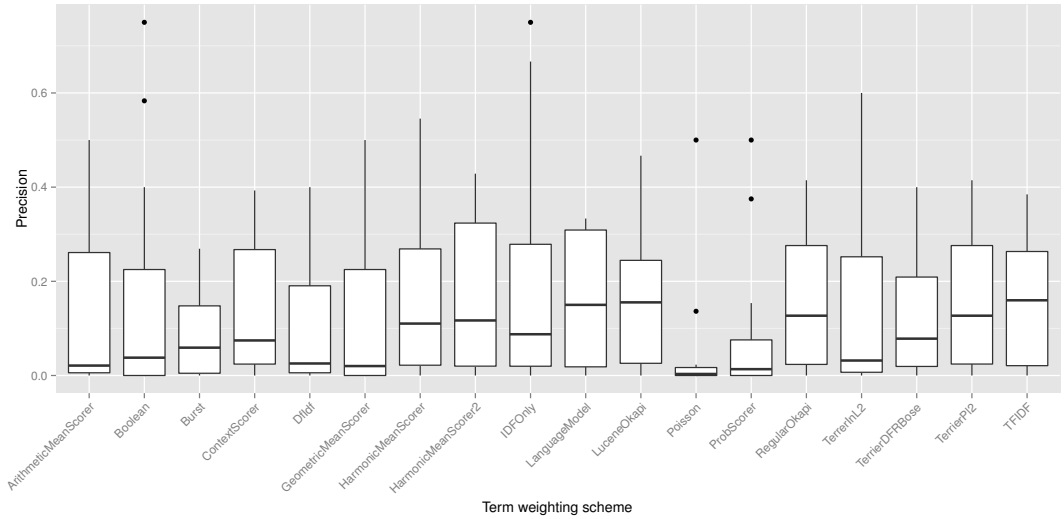

Figure 8.6: Box plot for precision values per term weighting schemes for atemporal search profiles in the TREC 2011 corpus.

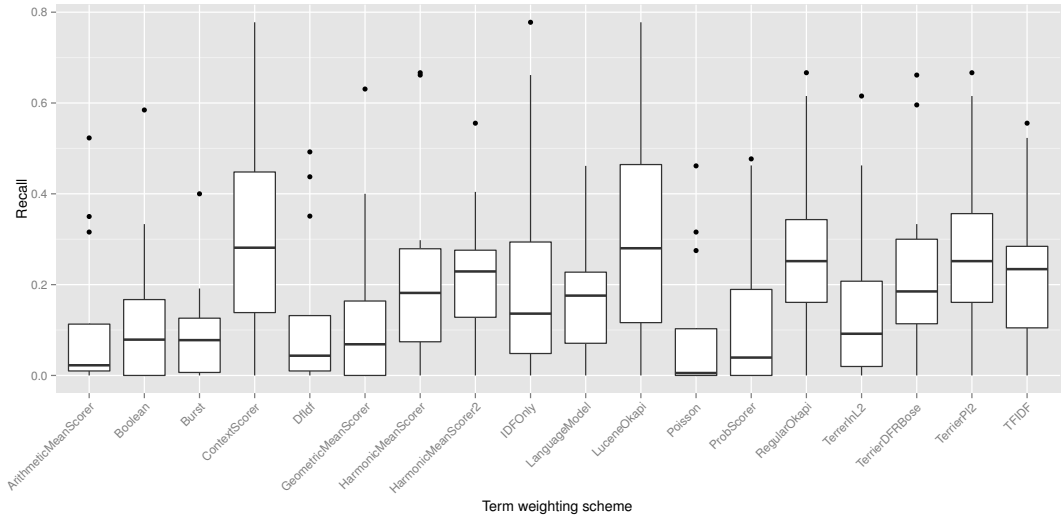

Figure 8.7: Box plot for recall values per term weighting schemes for atemporal search profiles in the TREC 2011 corpus.

support the assumption that documents for this type of profile arrive regularly and in low frequency, so it is difficult for the term weighting schemes to achieve a decisive and meaningful score.

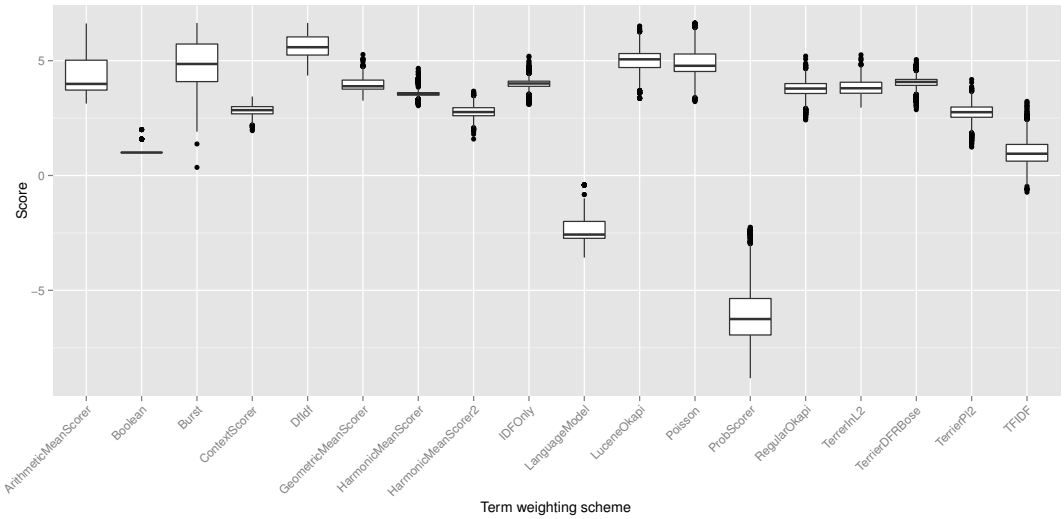

Figure 8.8: Box plot for score values per term weighting schemes for atemporal search profiles in the TREC 2011 corpus.

In summary, none of the term weighting schemes seems to be able to efficiently score documents for search profiles that belong to the atemporal profile class[28]. The precision and the recall values are rather low, and there is no clearly observable trend within the term weighting schemes. The differences between the several term weighting schemes for precision and recall is obviously not based on the different quality of a certain term weighting scheme, but due to outliers. This can be seen by the long whiskers on top of each of the term weighting schemes.

Thus, according to this configuration, none of the studied term weighting schemes are suited for filtering atemporal profiles, hence improvements are required. First, atemporal profiles must be detected from the text stream in a reliable manner. This could be done by monitoring the interarrival time of documents, for instance. Then, it is necessary to augment the scoring functions with external evidence, e.g. by resolution of the embedded links. In general, as the frequency of documents for such a profile is rather low, incorporating external information is acceptable from a performance point of view, as the high-volume text stream and efficient processing aspects do not

[28]The F1 measure, sensitivity and specificity were omitted here, in order to keep the chapter size within reasonable bounds. But details for these values can be checked in the online repository. The link is `https://bitbucket.org/streamfi/streamfi`, the user name is *streamfi* and the password is *BX0lzV7ZLcEi*.

count here.

Temporally unambiguous search profiles

This section studies temporally unambiguous search profiles. As defined by (R. Jones & Diaz, 2007, p. 14) this type of search profile is "relatively distinct with respect to the time dimension". This means documents are relevant only within a certain temporal context (p. 15), e.g. "traffic jams in Munich on 12 March 2015". Again, the performance of the term weighting scheme was studied in terms of precision, recall and absolute figures. The analysis with box plots is also presented.

Precision and recall

The first thing about precision that becomes apparent is that the performance values are rather high compared to the atemporal ones. While in the latter scenario all term weighting schemes failed, in the temporally unambiguous scenario several term weighting schemes perform well. The best one is the TerrierInL2 term weighting scheme, followed by the new Harmonic Mean Scorer and – most remarkably – by the Boolean Scorer. The latter means, a term weighting scheme that focuses only on term matches and nothing else, yields the same high precision values as term weighting schemes with more sophisticated calculation formulae. But the impression of equal performances changes if a look at the overall values in table 8.17 are taken into account. Here it becomes apparent that the precision values are rather high, but the overall values in terms of retrieved documents are less than a half of the top performing term weighting scheme. This shows that a Boolean term match is a good indicator for document relevancy, but it also shows that this limits whether the term weighting scheme performs well to a narrow context. Other term weighting schemes seem to "soften" the border where a document is relevant or not while not sacrificing precision.

This also becomes apparent when studying the recall values is taken. Here, Boolean Scorer performed badly. Interestingly, the LuceneOkapi performed best in terms of recall. This is probably because terms in documents that appear in temporally unambiguous search profiles occur in bursts and thus the term frequency for these terms is rather high compared to other terms and sum up over the complete corpus, which yields higher values in the used OkapiBM25 formula[29].

In general, eleven out of eighteen schemes yielded precision values greater than 0.4. This indicates that on the one hand, there are several approaches suitable for finding precise results. On the other hand it underlines the highly

[29] As a reminder, due to the local term frequency sparsity, the document.

Term weighting scheme	Precision	Recall	F1
ArithmeticMeanScorer	0.3384	0.1824	0.3143
Boolean	**0.4495**	0.1939	0.3448
Burst	0.3063	0.0912	0.2021
ContextScorer	0.3873	**0.4276**	0.3879
DfIdf	0.2791	0.1523	0.3035
GeometricMeanScorer	0.4044	0.2457	0.3776
HarmonicMeanScorer	**0.4609**	0.3645	**0.4259**
HarmonicMeanScorer2	0.4249	0.2891	0.3758
IDFOnly	0.4290	**0.4129**	0.4201
LanguageModel	0.4169	0.2592	0.3505
LuceneOkapi	0.4128	**0.4512**	**0.4196**
Poisson	0.0889	0.1188	0.2368
ProbScorer	0.2062	0.1250	0.2226
RegularOkapi	0.4447	0.3913	0.4195
TerrierInL2	**0.4876**	0.2487	0.3534
TerrierDFRBose	0.3654	0.3565	**0.4319**
TerrierPL2	0.4447	0.3916	0.4196
TFIDF	0.4370	0.3202	0.3830

Table 8.16: Precision, recall and f1 values of different term weighting schemes applied on the TREC 2011 corpus for temporally unambiguous search profiles. The best three approaches per metric are underlined.

temporal nature of the studied corpora and the importance of the temporal dimension for scoring. Table 8.14 summarises the precision values of the various term weighting schemes.

Recall

In terms of recall the values also improved considerably compared to the atemporal ones (cf. table 8.16). The best performing term weighting scheme in this scenario is the LuceneOkapi approach, followed by the ContextScorer and IDFOnly, all of which yielded recall values larger than 0.4. One possible conclusion was that a small dynamic sub-corpus does not always outperforms a large, more complete corpus[30]. However, there is no large difference between the first and the second place, and thus the dynamic sub-corpus approach still is viable.

[30] As a reminder, the LuceneOkapi scorer had access to the complete corpus and was thus not dynamic.

Scoring function	TP	FP	FN	TN
LuceneOkapi	281	601	101	378
ContextScorer	259	656	123	323
IDFOnly	239	503	143	476
TerrierPL2	239	462	143	517
RegularOkapi	239	461	143	518
TerrierDFRBose	230	513	150	466
HarmonicMeanScorer	217	370	165	609
TFIDF	194	383	188	596
TerrierInL2	178	354	204	625
GeometricMeanScorer	172	325	210	654
HarmonicMeanScorer2	170	427	212	552
LanguageModel	168	331	214	648
ArithmeticMeanScorer	151	324	231	655
DfIdf	138	330	244	649
Boolean	117	155	265	824
ProbScorer	116	331	196	548
Poisson	108	297	204	582
Burst	41	120	124	416

Table 8.17: TP, FP, TN, FN values of different term weighting schemes applied on the TREC 2011 corpus for temporally unambiguous search profiles.

The absolute values shown in table 8.17 underline the better performance of the LuceneOkapi approach. But even though a χ^2 test between the LuceneOkapi and the other approaches show a significantly better performance with a p-value less than 0.001 the absolute number of examples is too low to conclude an overall valid superiority of the LuceneOkapi approach over the others, i.e. the approach to use a large corpus instead of a small dynamic one. Additionally, there are significant differences between the term weighting schemes that use the dynamic sub-corpus approach. Starting from the TFIDF scorer, the number of retrieved documents for every category of true positives, etc. differs remarkably from the top performing ones. Consequently, the ContextScorer, IDFOnly and TerrierPL2 can be recommended if decent recall values are required. But a closer look at the box plots can help to understand the results better.

Figure 8.9 shows the box plots of the precision values. What becomes apparent is that almost all term weighting schemes have long whiskers, i.e. there is a wide range of precision values yielded by the term weighting schemes. So each term weighting scheme performed unambiguous well and worse at the same time for temporally queries. But as mentioned while discussing the feedback quality and the number of available feedback documents (cf. 8.1.6), there are several search topics with only few relevant documents and, for instance, filtering the only two relevant documents yields a precision value of 1.0. Nevertheless, the top performing term weighting schemes have a rather dense inter-quartile range.

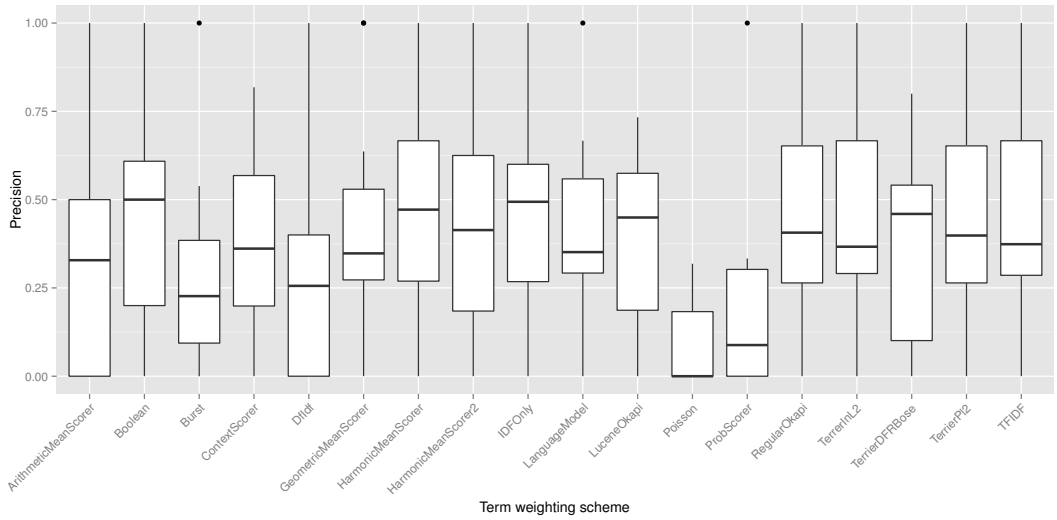

Figure 8.9: Box plot for precision values per term weighting schemes for temporally unambiguous search profiles in the TREC 2011 corpus.

The same applied to recall values. Figure 8.10 shows the box plot for the

recall values.

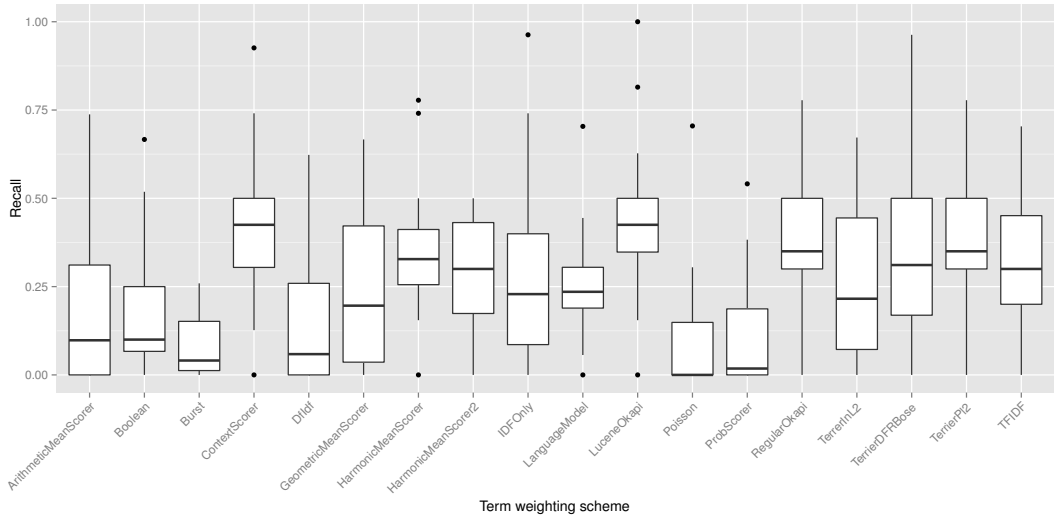

Figure 8.10: Box plot for recall values per term weighting schemes for temporally unambiguous search profiles in the TREC 2011 corpus.

Temporally ambiguous search profiles

Finally, the third profile category is reviewed, i.e. the temporally ambiguous ones. According to (R. Jones & Diaz, 2007, p. 15) profiles of this type "refer to the combination of several events and hence might be considered ambiguous if the user is looking for information about a specific event". This means, these queries are temporally sensitive, but not absolutely not focused on a single time period, e.g. the search profile "world cup soccer" can refer to the soccer world cup of 2014, 1990, 1974 or 1954. All queries are valid during the event, but which one is meant is not distinguishable from the query itself.

Table 8.18 shows the precision and recall values sorted by precision. This time RegularOkapi, TerrierPL2 and the two HarmonicMeanScorer performed significantly better than the rest. This is almost the same result as for the temporally unambiguous search profiles. Nevertheless, the absolute precision is worse by approximately 0.1 compared to the unambiguous profiles. However, this makes sense in a way, as temporally ambiguous search profiles are not as concise in terms of their temporal determinedness. Ambiguous queries can have different bursts and different periods where relevant documents can appear, while unambiguous ones are focused on a certain period in time. Precision

What is also interesting is that for temporally ambiguous queries, less term

Term weighting scheme	Precision	Recall	F1
ArithmeticMeanScorer	0.1825	0.1909	0.2014
Boolean	0.1482	0.1283	0.2348
Burst	0.0819	0.1041	0.1209
ContextScorer	0.2579	**0.3937**	0.2441
DfIdf	0.2313	0.2092	0.1858
GeometricMeanScorer	0.2635	0.2425	0.2215
HarmonicMeanScorer	0.3753	0.3259	0.2646
HarmonicMeanScorer2	0.3701	0.3054	0.2480
IDFOnly	0.2951	0.3671	0.2597
LanguageModel	0.2170	0.2755	0.2551
LuceneOkapi	0.2630	0.3639	0.2405
Poisson	0.1097	0.1563	0.1487
ProbScorer	0.1780	0.1811	0.1688
RegularOkapi	0.3837	0.3496	0.2770
TerrierInL2	0.2290	0.2720	0.2332
TerrierDFRBose	0.3098	0.3713	0.2589
TerrierPL2	**0.3837**	0.3508	**0.2771**
TFIDF	0.3119	0.2982	0.2416

Table 8.18: Precision, recall and f1 values of different term weighting schemes applied on the TREC 2011 corpus for temporally ambiguous search profiles.

weighting schemes yielded good results. While for the temporally unambiguous search profiles eleven out of eighteen term weighting schemes yielded high precision values (> 0.4), in this scenario only the four previously mentioned term weighting schemes performed in a satisfactory fashion. The fifth best term weighting scheme scarcely managed a precision value of 0.3.

Recall Regarding the recall values, again, the ContextScorer yields the best results. TerrierDFRBose from the divergence of randomness algorithm also performed well, as did the IFDOnly and the LuceneOkapi approach. Even though the *complete corpus* approach of LuceneOkapi performed well, it was again outperformed by the several dynamic sub-corpus based approaches.

Now the box plots are looked at. Figure 8.11 shows the distribution of the precision values over the used search profiles. The top four term weighting

schemes have the largest inter-quartile range[31], i.e. they have a broad range of precision values. This means that they have a larger number of search profiles compared to the other term weighting schemes for which the they yielded higher precision values. All term weighting schemes have their outliers, but their inter-quartile range is often rather limited. For instance, the fifth best term weighting scheme TFIDF has a limited range of precision values with only a few outliers.

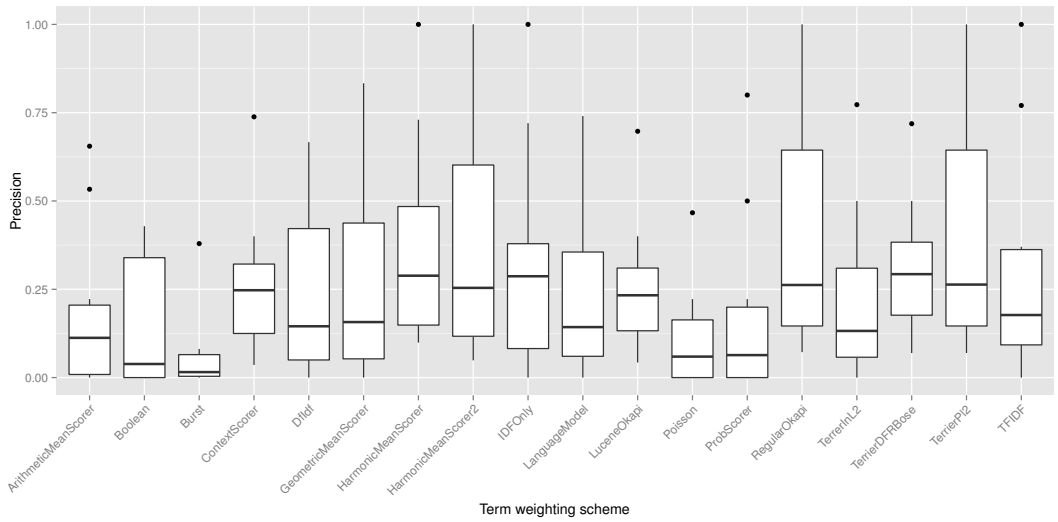

Figure 8.11: Box plot for precision values per term weighting schemes for temporally ambiguous search profiles in the TREC 2011 corpus.

Table 8.18 shows the recall values. This time the ContextScorer scheme yielded the highest recall value, followed by TerrierDFRBose and LuceneOkapi. So the two best performing term weighting schemes from the temporally unambiguous profiles – ContextScorer and LuceneOkapi – also performed best at the temporally ambiguous ones. TerrierDFRBose has a large whisker at the top. This indicates that the upper 25% of the recall values were situated there and thus the good position is due to outlier search profiles that raised the overall average. It is also notable that the recall values are not far worse than the recall values of the temporally unambiguous search profiles. Recall

8.3.5.1 Performance summary of term weighting schemes for TREC corpus

This section studies the results of the previous examination. Figure 8.13 shows a heatmap that visualizes the averaged rank each term weighting Overall ranking

[31] The inter-quartile range represents the quantiles between 25% and 75%.

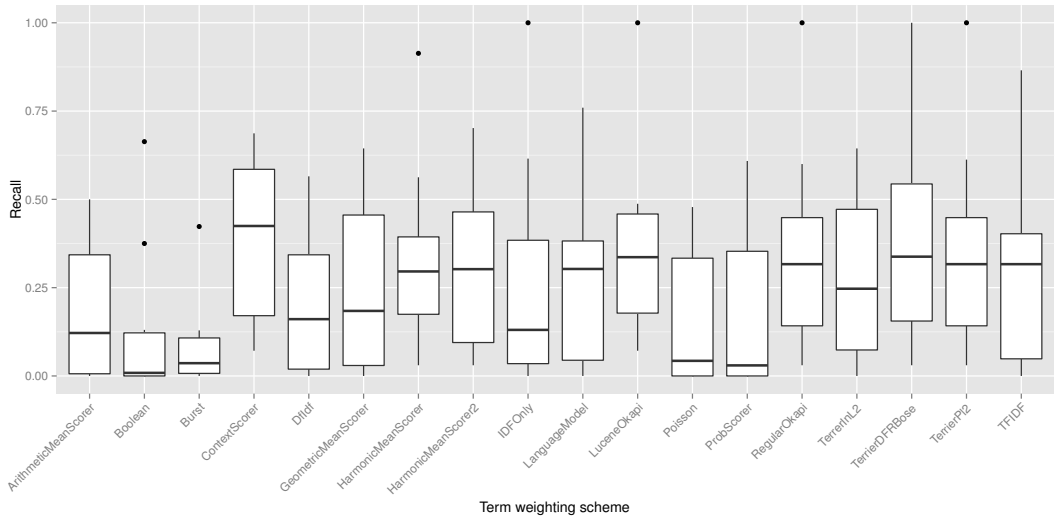

Figure 8.12: Box plot for recall values per term weighting schemes for temporally ambiguous search profiles in the TREC 2011 corpus.

scheme achieved in terms of precision, recall and f1-measure[32]. It becomes apparent that the best overall term weighting schemes were TerrierPL2, OkapiBM25 and the meta-scorer using the harmonic mean of various term weighting schemes. The rationale to use the harmonic mean also was also justified by the poor performance of the ArithmeticMeanScorer and the GeometricMeanScorer. It is also notable that the Poisson based approach performed worst. The new ContextScorer did not perform well in terms of precision, but yielded the best recall values. Thus, this scheme can be recommended for recall oriented filtering tasks. LuceneOkapi, the term weighting scheme using the static context, also performed well, but was not the best in class, which underlines that the dynamic sub-corpus approach is valid. Interestingly, the IFDOnly approach also performed, which can be seen as an indicator that the local component of a term weighting scheme is not necessarily required. The recall values were good, but the precision values fell short in comparison. But if a closer look is taken, it can be seen that this term weighting scheme performed well for atemporal and temporally unambiguous search profiles. The precision was only low for the temporally ambiguous profile category.

Score distributions

The score distributions of relevant and non-relevant documents generated by each term weighting scheme are shown in figure 8.14. This figures helps to understand, why some term weighting schemes performed better than the other by illustrating how the scores of relevant and non-relevant documents were distributed. In general, the larger the offset of the relevant document

[32]This means, e.g. if a term weighting schemes was the best in atemporal profiles and rank 4 and 7 in the other categories, it achieved a value of 4.

	Precision	Recall	F1
ArithmeticMeanScorer	13.00	14.67	15.00
Boolean	6.67	14.67	8.33
Burst	16.67	18.00	16.67
ContextScorer	11.33	1.33	8.33
DfIdf	14.00	14.33	15.33
GeometricMeanScorer	11.00	12.00	11.67
HarmonicMeanScorer	2.33	6.67	3.33
HarmonicMeanScorerDocument	5.00	8.67	8.33
IDFOnly	8.00	2.67	4.33
LanguageModel	10.00	10.00	9.67
LuceneOkapi	8.67	2.67	5.67
Poisson	17.67	16.67	17.00
ProbScorer	16.00	14.67	16.00
RegularOkapi	4.00	5.33	4.00
TerrerInL2	7.33	11.00	11.67
TerrierDFRBose	10.67	5.00	5.00
TerrierPl2	3.67	4.33	2.67
TFIDF	5.00	8.33	8.00

Figure 8.13: Summary heatmap of term weighting schemes applied on the TREC 2011 corpus.

score distribution (light blue) is from the non-relevant document distribution (light red), the better, because this means that the scores for the relevant documents are considerably larger than for the non-relevant and thus it is easier to draw threshold between the two document classes.

The best three schemes, Terrier PL2, OkapiBM25 and HarmonicMeanScorer, yielded higher scores for relevant than for non-relevant documents. This is shown by the light blue score density distribution of the relevant documents, which shifted to the right in comparison to the score density of the non-relevant documents. They do not have any outliers on the upper end of the score values, i.e. the non-relevant distribution does not overlap the relevant distribution. In the LanguageModel there are outliers in the upper range, which thus affect the performance of the term weighting scheme. This figure also shows why the Poisson Scorer completely failed. The scores for the non-relevant documents where always higher than those for the relevant documents and thus used threshold could not distinguish between relevant and non-relevant documents. The results for the Boolean Scorer are also interesting. This term weighting scheme only yields scores like $0, 1, 2, 3, \ldots$ ignoring any temporal aspect of the text stream, still this strategy produced good precision values. The overall values of returned documents were not high compared to the other term weighting schemes, but nevertheless the precision was good.

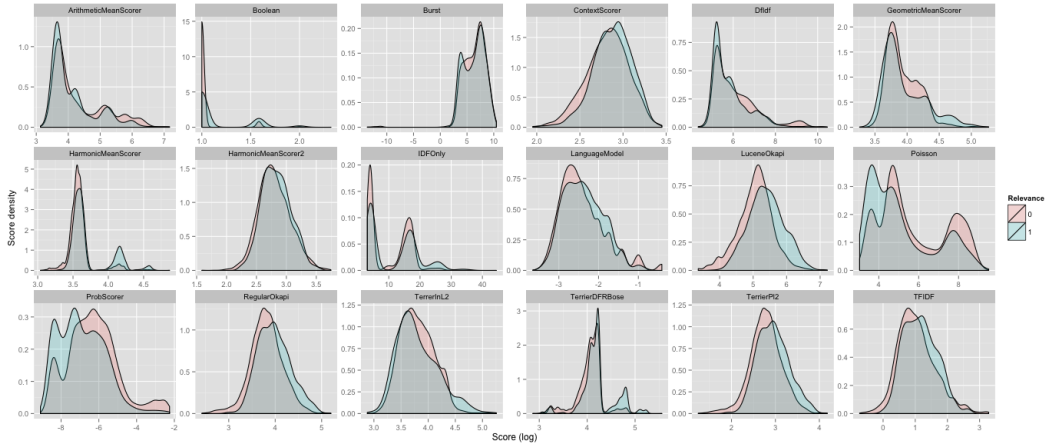

Figure 8.14: Distribution of scores for relevant and non-relevant scores of the studied term weighting schemes applied on the TREC corpus.

8.3.5.2 Evaluation of the Events corpus

Table 8.19 shows the overall results for the Events corpus and reveals various findings. First the precision and recall values are significantly lower than for the TREC corpus. The best f1 value is only 0.1684, which is only the half of the best of the TREC evaluation. Nevertheless, there are four term weighting schemes that perform remarkably better than the rest: Boolean Scorer, TerrierInL2, ArithmeticMeanScorer and GeometricMeanScorer. This is interesting, as the two mean scorers did not perform very well on the TREC corpus. Boolean Scorer and TerrierInL2 also performed well on the TREC corpus. In terms of recall TerrierPL2, TerrierDFRBose, RegularOkapi and the new ContextScorer performed best. IDFOnly did also well, the others performed rather badly. In general, the overall evaluation does not show any clear trend towards any specific term weighting scheme.

Figure 8.15 shows the score distribution values. Although several term weighting schemes show a clear offset between the score distribution of irrelevant and relevant documents, none of them succeeded in creating high precision values.

Figure 8.16 shows the box plots of the precision values for all search profiles. The first observation is that the median precision value is almost equally low for all term weighting schemes, i.e. that the precision values of 50% of the search profiles were below that line. A few have a larger inter-quartile range, like Boolean, TerrierInL2 or LanguageModel, but none of the shows a significant better performance from the visual examination. Furthermore, all term weighting schemes have outliers that help to yield a more or less equilibrated level of precision in comparison.

Term weighting	TP	FP	FN	TN	Precision	Recall	F1
Boolean	2837	17307	1895	0.1909	0.2753	0.1684	0.1812
TerrierInL2	3127	26065	1605	0.1799	0.3002	0.1497	0.1563
GeometricMeanScorer	3331	35950	1401	0.1605	0.3180	0.1412	0.1365
DfIdf	2544	29488	2188	0.1296	0.2514	0.1380	0.1810
ArithmeticMeanScorer	3115	34299	1617	0.1673	0.2869	0.1353	0.1647
HarmonicMeanScorer	3122	32468	1610	0.1396	0.3376	0.1348	0.1035
LanguageModel	3002	32192	1730	0.1356	0.3121	0.1346	0.1234
IDFOnly	3204	42328	1528	0.1220	0.3785	0.1315	0.0870
LuceneOkapi	2450	31116	2282	0.1240	0.2783	0.1314	0.0965
TerrierDFRBose	3281	42659	1448	0.1228	0.3852	0.1313	0.0866
TerrierPL2	3357	43403	1375	0.1297	0.3858	0.1302	0.0915
RegularOkapi	3342	43148	1390	0.1295	0.3830	0.1296	0.0918
TFIDF	3010	36705	1722	0.1393	0.3402	0.1244	0.0998
ContextScorer	3234	56933	1498	0.1137	0.3807	0.1231	0.0826
HarmonicMeanScorer2	2524	34336	2208	0.1242	0.2962	0.1200	0.1019
Burst	1176	10563	894	0.0816	0.2479	0.1039	0.1204
ProbScorer	2058	26595	1196	0.0841	0.1647	0.0961	0.1540
Poisson	2375	27827	879	0.0909	0.1838	0.0914	0.1526

Table 8.19: Overall evaluation of term weighting schemes applied on the Events 2012 corpus.

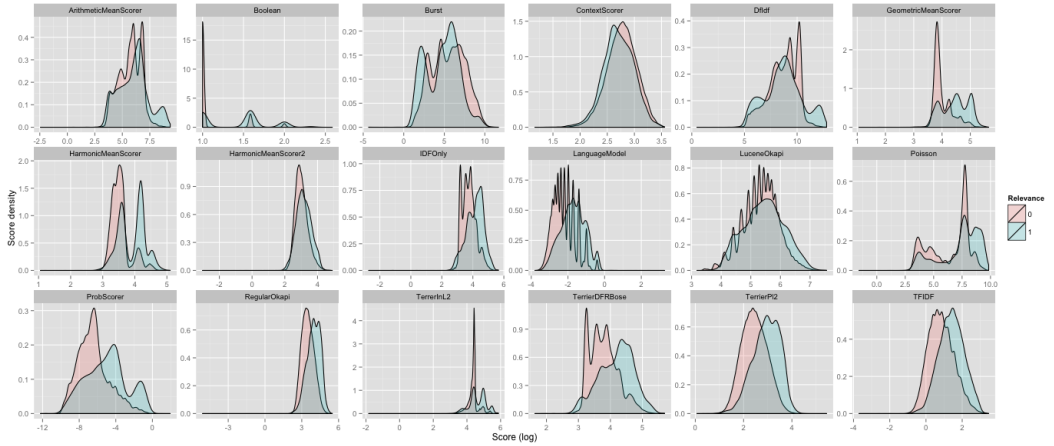

Figure 8.15: Score distributions generated by various term weighting schemes applied on the Events 2012 corpus.

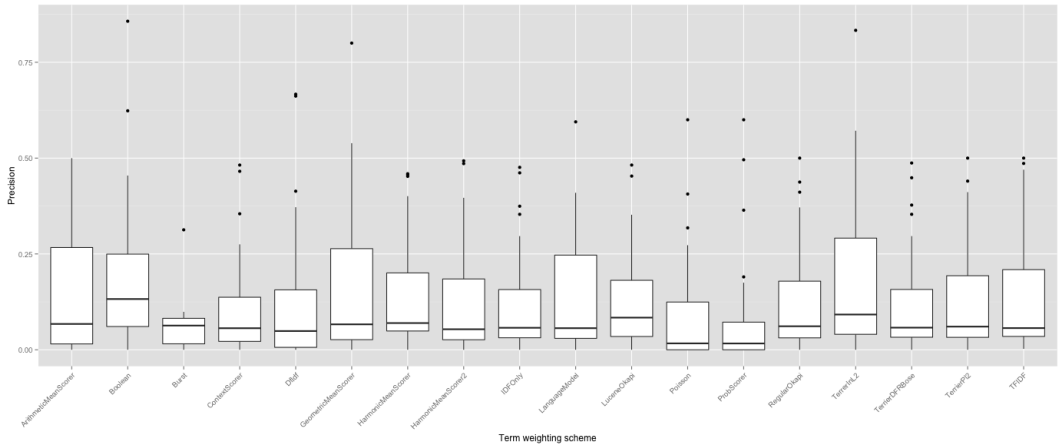

Figure 8.16: Box plots of the precision values for the term weighting schemes applied on the Events 2012 corpus.

No clear picture for Events corpus

In summary, no clear picture can be drawn from the data for the Events corpus, because of the many outliers for every term weighting scheme. This is probably due to the weak relevance judgements that are so biased or of poor quality, which is why no clear overall performance order cannot be devised. A further argument that the Events corpus' relevance judgements are not too good is that the BursT term weighting scheme – a term weighting scheme in fact designed for temporal text streams – yields tremendously good precision and recall values, .3129 and .6933 respectively, for the atemporal search profiles (!), but performs rather badly for the temporal ones (cf. appendix D for details). But as the BursT values are contradictory to the results of the TREC corpus, where BursT did not perform well on the atemporal profiles, either this supports the assumption of the biased quality of the Events 2012 corpus' relevance assessments.

Summary

In summary, the evaluation of the Events corpus can only provide limited recommendations. This is largely because the results, when compared to the TREC results on a temporal level, show no difference between the three profile types. While the TREC results fit the each temporal profile, i.e. the atemporal profile is hard to grasp, and leads to mediocre performance, the two temporal profiles yielded substantially better results, the Event results did not show that much difference between all three profiles. Consequently, the relevance assessments of the Events corpus can be considered as rather arbitrary.

Nevertheless, in terms of recall the ContextScorer again performed well, which indicates that taking more than one term features into account for document scoring can be beneficial to finding more relevant documents. In terms of precision, the Boolean Scorer can be recommended. This is information is helpful, as it shows that in a scenario where relevance assessments are scarce or of bad quality, the *good, old* heuristic approach to take a *simple* term match as a relevance indicator is still valid.

8.3.5.3 Comparison and summary

This section displays a brief comparison of the performance of the various term weighting schemes in both corpora is shown. Table 8.20 shows the averaged rank of each term weighting scheme per corpus. This helps to summarise the many tables and facts from the previous sections. The best results per column are shown in bold.

Term weighting	Precision		Recall		F1	
	Events	TREC	Events	TREC	Events	TREC
ArithmeticMeanScorer	3.67	13.00	7.33	14.67	7.33	15.00
Boolean	**1.67**	6.67	14.00	14.67	**2.33**	8.33
Burst	12.33	16.67	8.67	18.00	12.33	16.67
ContextScorer	14.00	11.33	5.67	**1.33**	11.67	8.33
DfIdf	7.67	14.00	10.67	14.33	7.33	15.33
GeometricMeanScorer	4.33	11.00	8.67	12.00	7.67	11.67
HarmonicMeanScorer	7.33	**2.33**	9.00	6.67	7.67	3.33
HarmonicMeanScorer2	10.67	5.00	10.00	8.67	13.00	8.33
IDFOnly	12.33	8.00	9.67	2.67	9.67	4.33
LanguageModel	7.67	10.00	6.33	10.00	8.33	9.67
LuceneOkapi	12.33	8.67	12.67	2.67	8.67	5.67
Poisson	16.00	17.67	17.00	16.67	14.33	17.00
ProbScorer	16.67	16.00	17.00	14.67	11.67	16.00
RegularOkapi	10.67	4.00	5.00	5.33	11.33	4.00
TerrierInL2	4.00	7.33	8.67	11.00	6.67	11.67
TerrierDFRBose	12.00	10.67	9.33	5.00	9.33	5.00
TerrierPL2	10.33	3.67	**4.33**	4.33	10.33	**2.67**
TFIDF	7.33	5.00	7.00	8.33	11.33	8.00

Table 8.20: Average rank over the three temporal search profiles of the various term weighting schemes on the TREC and the Events corpus side by side. Best performing scheme per column is bold.

8.3.5.4 Summary

This section evaluated eighteen term weighting schemes and their performance on two social media text stream corpora and three temporal profile categories. The Events corpus showed biased results due to the poor level of quality of the relevance assessments. The three main findings are first that the LuceneOkapi approach that used the complete corpus did not score best and was outperformed by other term weighting schemes. Second, Boolean Scorer in a setting with few or bad relevance assessment worked well and, third, that in terms of recall the ContextScorer was a good choice. The TREC corpus – based on a domain expert based assessment – also showed that the new proposed ContextScorer performs well in terms of precision and foremost recall, and the HarmonicMeanScorer does reasonably well. Seminal term weighting schemes like Okapi BM25 or term weighting schemes based on divergence from randomness are also suitable in a text stream scenario.

8.4 Threshold methods for text stream filtering

As stated in the previous section, the scoring functions used in this dissertation are linear feature-based models, i.e. the scoring functions sum up the score of the various components to a final document score. These kinds of functions always yields monotonically increasing score values; the rationale of such a function is that relevant documents are supposed to yield higher scores than non-relevant documents. Consequently, it is necessary to derive a threshold that separates the relevant from the irrelevant documents, i.e. a boundary is required to classify the incoming documents. The identification of this boundary is the task of classifiers. As here linear feature-based models are used, the here proposed thresholds can also be considered as *linear classifiers* (Schütze et al., 2009, p. 301)[33].

In this section two approaches to threshold determination are studied. The first approach is to monitor the scores of a classified text stream, i.e. the scores of documents are observed, and a threshold based on statistical moments like average or standard deviation is calculated. This is a straight-forward and fast method to determine thresholds. This approach does not involve any kind of machine learning, but only monitors of document scores and properties. Nevertheless, it can be beneficial to have a fast and easy way at hand to discern relevant from irrelevant documents. The dynamic subcorpus approach that constantly moves over the data and forgets old scores

[33]The basics and foundation of thresholds and classifiers were discussed in section 3.3.4

could also help to address the problem of noise documents (cf. 3.3.4).

This approach is compared to the second approach that involves machine learning. In the filter policy chapter a preliminary filter decision was made by applying weak classifiers that required a document to be only fairly relevant. In this section the classification is performed more rigidly. This is for two reasons. First the score value – as an important feature – is available and, second, the decision boundaries of the classifiers are not as weak as within the filter policy phase. In general, machine learning approaches are a promising field to solve the classification problem of documents and got a real boost since the beginning of the 2010s due to the rapidly increasing computer power. Thus, the future is definitely ruled by approaches that learn supreme models using peta-byte of training data. Nevertheless, not all datasets are tremendously large, and not always are large volumes of training data, therefore the approaches, which are studied in this dissertation, provide viable knowledge about efficient methods where fast text stream data are involved, but the training data are limited[34].

This section is structured as follows: First score-based approaches are introduced and an approach is presented on how to leverage the possibilities of an event-driven information filtering system to monitor a multitude of different aspects of a stream and derive highly specialised thresholds. A meta threshold algorithm using harmonic mean is also presented, similar to the one introduced in the term weighting section. Second, the quality of several machine learning algorithms in the context of high-volume text streams is evaluated. This means features are derived based on the dynamic sub-corpus approach, i.e. the features are only normalised using the values within a sliding window, and fed into several machine learning algorithms. The results are compared to the "easy" threshold approach. Finally, these approaches are compared to the standard approach to model a decision threshold as the intersection of the assumed distribution of the relevant and irrelevant documents, e.g. by using a *Two-Gaussian-mixture-model*, *Gaussian-Exponential-mixture-model*, etc. (cf. section 3.3.4).

8.4.1 Score based threshold strategies

In research literature, thresholds are often only determined by using the classes relevant vs. non-relevant (cf. section 3.3.4). However, this is too coarse

[34] As discussed in section 8.1.6 the quality of the available relevance assessment leaves space for improvements. Thus, the quality of the models is also limited. Nevertheless, this provides good insights into how classification algorithms work using features derived continuously from a text stream using a dynamic sub-corpus approach.

grained as these overall thresholds fail to address the specificities of documents within each of the two classes. For instance, a negative document which, by accident, has three query matches scores higher than a relevant document with two query matches. If the majority of relevant documents only have two query matches, it is probable that the average score for the positive document class is lower than the score for the non-relevant document with three query matches. The consequence would be that the three query matches document would be judged as relevant.

But if there was a facility that would monitor the scores for each query match combination (1, 2, 3... query matches), a query match specific threshold could be derived. E.g. the average score for the positive document with 1, 2 or 3 query matches would be monitored within different sub-streams and the scores for non-relevant document as well. Then only the thresholds and distributions of the relevant and non-relevant documents obtained using 1, 2, 3, etc. query matches would be compared.

The event-driven approach and the use of an event processing engine that continuously monitors the text stream allows to address this. It is easy to monitor different aspects and different decision criteria streams. This is done by segregating each scored document into a specific sub-stream and monitor the values accordingly.

The threshold grammar described in 8.1 illustrates the different kinds of thresholds that could be constructed. The *«aspects»* represent the stream property and the *«criterion»* the value type of the stream property that should be monitored. The flexibility allows constructing many different thresholds, for instance a sub-stream that monitors all documents that belong to class "positive", profile category "temporally unambiguous" that have two query matches and one feedback match.

⟨*threshold*⟩ ::= ⟨*aspect*⟩* ⟨*criterion*⟩
⟨*aspect*⟩ ::= [query matches | feedback matches | search profile | profile category | document class]
⟨*criterion*⟩ ::= [percentile | top-k | ⟨*mean*⟩ | *n* sigma]
⟨*mean*⟩ ::= [harmonic | geometric | arithmetic | quadratic | ...]

Grammar 8.1: Grammar for threshold construction.

For the purpose of this study the following thresholds were used:

- Average score over all Scored Events

- Gaussian Exponential mixture model
- Harmonic mean threshold (cf. next section)
- Top-k threshold (with k set to 20)
- Percentile based threshold (threshold set to top 10% of the scored documents)
- Percentile per query threshold
- Average score per each individual query
- $n \cdot \sigma$ threshold with n set to 2^{35}
- Average positive score, i.e. only the score of documents that passed a previous threshold and were thus considered as positive was monitored
- Average positive score per query
- Average score per total query matches

8.4.2 Harmonic mean threshold

This threshold strategy represented a specialised version of an average threshold approach. But instead of averaging the latest score values within a sliding window, this threshold averages the multiple threshold values within the sliding window. This is similar to the harmonic mean approach used for scoring in the previous section. Already there, the harmonic mean approach performed decently and outperformed other mean based strategies. Thus it seemed reasonable to study this approach for thresholds as well.

8.4.3 Machine learning based classification

Scored based methods provide a classification decision based solely on the score value. Nevertheless, other features as well, like spelling errors, document length or number of parts-of-speech, contain information that can be beneficial for classification. Therefore, the classification of documents using different machine learning algorithms is studied. The goal is to provide a comparison between algorithms and discover which performs best in the given context. The quality of features is also studied that were derived using context information from the dynamic sub-corpus. In general, features, which are fed into a machine learning algorithm, required some kind of preprocessing in form of normalisation or standardisation. This is because, features often have different units or magnitudes of values, and in order to avoid

[35] A $n\sigma$ filter is based on the idea of setting the threshold to average plus n times the standard deviation of the monitored scores.

the dominance of particular features the values are normalised to a certain range (Janert, 2010, p. 311).

In the context of this dissertation, the main goal is to show if features that are only normalised or standardised[36] using a temporally limited scope, i.e. a dynamic sub-corpus, are capable of providing decent classification results and which classification algorithms perform best using these features[37] And again it must be stated that the training data are too few to obtain really good classifier performance. In the times of Big Data where deep learning algorithms analyse datasets with millions or even billions of training items and thousands of features, this is a rather limited dataset and thus the performance cannot be overwhelming. Nevertheless, the comparison between scored based thresholds and machine learning classifiers is interesting, because it shows how well both perform using yet limited, but also high-volume text stream data. This gives good insights into which methods could be used in a context where text documents arrive at a continuous and high-pace, but the relevance information on each topic is rather limited.

8.4.3.1 Score distribution mixture models

The seminal approach to model relevant and non-relevant documents with different distributions was presented in (Swets, 1969, p. 11ff) The idea was to determine a decision threshold for the intersection of the two class distributions. In (Baumgarten, 1999) for instance the intersection of a Gaussian, representing the relevant document, and a *Gamma* distribution representing the non-relevant documents was used. In (Arampatzis, 2001), (Lanquillon, 2001) and (Y. Zhang & Callan, 2001b) an *Exponential distribution* was used to model the irrelevant documents. But there are other distributions that can be used (cf. (Swets, 1969), (S. Robertson et al., 2013)).

The post-hoc analysis of the scores for relevant and non-relevant documents in the used corpora supported the assumption to model relevant document classes with a Gaussian, and non-relevant with an exponential distribution. In order to be able to evaluate the performance of the distribution mixture threshold approach, the parameters that are required to specify both distribution were taken from the post-hoc analysis and are summarised in table 8.21.

[36] Normalising is to convert features to a scale between 0 and 1, while standardising means to scale the values to have a mean of zero and a standard deviation of 1

[37] It must be noted that building machine learning models is an iterative process that often requires intuition and creativity (Domingos, 2012), thus, only a proof-of-concept like evaluation can be provided. Commercial use would require more tuning.

Value	TREC	Events
Average relevant	11.59	16.93
Standard deviation relevant	3.52	6.08
Average non-relevant	10.35	12.01
Standard deviation non-relevant	2.60	4.38
Calculated threshold	12.47	15.67

Table 8.21: Corpus statistics used to model the distribution mixture model thresholds.

In a scenario where more relevance feedback is available the parameters could be easily monitored using the proposed system architecture and the monitored values could be applied to model the distribution. Due to the very low number of relevance feedback, this "short cut" was chosen, in order to be able to have viable parameters to derive the threshold.

Nevertheless, the "prophetic" use of these parameters provided an interesting case for comparison, because the mixture model threshold method thereby gained knowledge that the other approaches did not have. The other approaches only had access to the information that had arrived before a document arrived, while the mixture threshold had incorporated all information, i.e. all scores of relevant and non-relevant documents, in the corpora. Thus, it was logical to assume that this threshold might perform better than the other thresholds, because it contained more information about the stream. This assumption is also verified here.

8.4.4 Evaluation of threshold performance

The performance of the thresholds and classification algorithms is first presented on an overall level and then a look at the category level performance is taken. Table 8.22 shows the performance of different threshold strategies for the TREC 2011 corpus. In order to guarantee comparability, the configuration was the same as for the baseline runs, i.e. OkapiBM25 was used for scoring and no query expansion was applied.

8.4.4.1 Overall evaluation

In terms of precision, the best result was obtained using the harmonic mean strategy that calculated the decision threshold by using the harmonic mean of several other threshold strategies. The second best approach was the Average Score threshold. This threshold just calculates the average score of the

documents within a sliding window and if a document exceeds this value, it is considered as relevant. In general, all studied thresholds yielded precision values between 0.2707 and 0.3312 and the first six were between 0.3052 and 0.3312. Thus, precision alone does not tell to much about the over all quality and further indicators like recall and f1 should be considered.

TREC corpus

In terms of recall, there is a clearer picture. Again, AverageScorer yields the best recall result, followed by AverageScorePerQuery – the same as AverageScorer but on a query level –, GaussianExponentialMixture model and AveragePerTotalMatches – here the scores are grouped by the number of matching queries within the document. All these four yielded recall values between 0.3108 and 0.3229. Afterwards, there was a steep drop off in recall performance. Table 8.22 shows the detailed values over all applied score-based threshold algorithms. In summary, the straight forward Average Score threshold yielded the best results. The more specialised ones, as proposed in section 8.4.1, performed well, but not significantly better or worse.

Events corpus

In total, the Events corpus contains data than the TREC corpus, but this is not the case on a per search profile. Together with the fact that the relevance assessments are not too good – as shown in the previous section – the results for the thresholds are probably also biased. Table 8.23 shows the overall results[38]. Again the performance in terms of precision is rather poor. The best threshold in terms of precision is the TopKThreshold, but if the total number of true positives is considered, this approach does not yield reasonable results. The second best approach regarding precision, is AveragePositiveScoreStrategy, i.e. the average score of the documents that exceeded their threshold, which also performed acceptably in terms of recall. Nevertheless, the precision values are all rather low and range from 0.1003 to 0.1665, which is not too good. Again, this shows the poor quality of the relevance feedback, especially as compared to the TREC results. The recall values, especially for the top performing ones, are not too bad. Here again the AverageScorer performed best, followed by the GaussianExponentialThreshold and the AverageScorePerQuery. The other threshold strategies did not perform well and yielded results below 0.27. This is also true for the more specialised threshold strategies that took query matches into account.

Machine learning based classifiers

Now, a brief look at the performance using machine learning classifier is taken. For this five different classification algorithms from the *JSAT*[39] machine learning library were used and evaluated[40]. As described in the in-

[38] The true negative rate could not be reported as there were no negative relevance assessments avaible in the Events corpus.

[39] `https://github.com/EdwardRaff/JSAT`

[40] For details on the algorithms cf. `https://github.com/EdwardRaff/JSAT/wiki/`

Threshold	TP	FP	FN	TN	Precision	Recall	F1	T11SU
AveragePositiveScorePerQuery	397	2030	647	14837	0.2707	0.2283	0.2594	0.2291
AveragePositiveScoreStrategy	390	2666	654	14201	0.3151	0.1935	0.2601	0.2771
Average Score	653	7038	391	9829	0.3229	**0.3307**	**0.2929**	0.2570
AverageScorePerQuery	628	7649	416	9218	0.2771	0.3266	0.2737	0.2333
GaussianExponentialThreshold	594	9139	450	7728	0.2758	0.3264	0.2760	0.2350
HarmonicMeanThreshold	357	2290	687	14577	**0.3312**	0.1893	0.2614	0.2615
Percentile	306	1585	738	15282	0.3083	0.1492	0.2390	**0.2969**
PercentilePerQuery	262	1867	782	15000	0.3008	0.1421	0.2375	0.2545
SigmaPerMatchesPerQuery	418	3461	626	13406	0.2746	0.2355	0.2612	0.2280
SigmaPerQuery	427	3156	617	13711	0.2854	0.2350	0.2692	0.2338
TopKThreshold	228	645	816	16222	0.2960	0.1450	0.2548	0.2776
AveragePerTotalMatches	604	7648	440	9219	0.2938	0.3108	0.2732	0.2444
SigmaOverallScore	359	2314	685	14553	0.3101	0.1696	0.2462	0.2740
SigmaPerQueryMatches	299	2637	745	14230	0.3052	0.1482	0.2402	0.2632

Table 8.22: Evaluation of the examined threshold strategies applied on the TREC 2011 corpus.

Threshold	TP	FP	FN	Precision	Recall	F1
AveragePositiveScorePerQuery	1437	26339	3482	0.1065	0.2132	0.1029
AveragePositiveScoreStrategy	2318	35951	2601	0.1467	0.2633	0.1208
AverageScore	**3507**	84990	**1412**	0.1254	**0.3958**	**0.1255**
AverageScorePerQuery	3060	89845	1859	0.1222	0.3648	0.1192
GaussianExponentialThreshold	3175	148760	1744	0.1136	0.3869	0.1130
HarmonicMeanThreshold	1661	29942	3258	0.1218	0.2146	0.1177
Percentile	1588	21489	3331	0.1400	0.1809	0.1201
PercentilePerQuery	1143	23673	3776	0.1209	0.1532	0.1065
SigmaPerSourceMatches	1299	35319	3620	0.1003	0.1669	0.0937
SigmaPerSourceMatchesPerQuery	1636	46792	3283	0.1060	0.2409	0.1026
StddevOverallScore	2076	33409	2843	0.1429	0.2349	0.1210
StddevPerQuery	1794	40705	3125	0.1164	0.2471	0.1152
TopKThreshold	292	1594	4627	**0.1665**	0.0585	0.1186

Table 8.23: Evaluation of the examined threshold strategies applied on the Events 2011 corpus.

troduction, the goal here is to show how simple score-based mechanism perform compared to classifiers that learn a model from the text stream. It is also interesting to see how well these algorithms work, when features are used whose values are not standardised or normalised using a complete data set, but are only based on the dynamic sub-corpus approach. Table 8.24 shows the results for the TREC corpus. The precision values are in the range of the lower performing score-based threshold algorithms, i.e. a machine learning approach in the given setting is not superior to a simple score-based threshold based method. The picture changes in terms of recall. Here, RandomForest or NaïveBayes outperform the threshold based methods significantly. More and particularly better training data could change the picture here, but as mentioned, in a scenario where good training is scarce or hard to obtain, the threshold based methods work well.

Table 8.25 contains the performance values of the classifiers for the Events corpus. Here, the same picture emerges as for the score-based thresholds. The precision values are rather low and were consistent with the performance of the score-based threshold. The recall values varied dramatically. For instance, StochasticMultLogReg failed completely in terms of recall, even though this algorithm performed well on the TREC corpus. It is also interesting to see that the LinearSGD algorithms yielded a high recall value. RandomForest and the Plat SMO algorithm yielded an average result in terms of recall. In summary, this supports the notion that the Events corpus data and relevance assessments are too noisy, in order to produce substantially good performance values.

Thresholds per profile category

This section investigates the performance of the various threshold strategies in the context of the three defined profile categories. Again, the goal is to show, in detail, which methods works best for a given profile category. Table 8.27 provides a summary of the performance of different score-based threshold mechanisms.

Atemporal

The TREC results are consistent with the term weighting, i.e. atemporal profiles are the hardest profile category to address, which is demonstrated by the lower performance values. The best threshold in terms of precision is the StatsPerSourceMatches. This is a strategy based on a specialised sub-stream that monitors the document scores on a query match level and judges all documents as relevant that exceed the average plus σ standard deviations. In this case the value of σ was set to 1. In terms of recall the GuassianExponential

`Algorithms`. This wiki page also provides all references to the papers on which the algorithms are based. The choice of algorithms was based on their popularity. Random Forests, Support Vector Machines, NaïveBayes or stochastic regression are among the most popular classifier algorithms (Wu et al., 2007).

Algorithm	TP	FP	FN	TN	Precision	Recall	F1	
Passive Aggressive	427	8683	617	8184	0.2037	0.1971	0.2094	0.2047
Random Forest	886	16038	158	829	0.2549	0.4402	0.2689	0.2154
Plat SMO	842	16335	202	532	0.2505	0.4171	0.2608	0.2078
StochasticMultLogReg	811	11883	233	4984	0.2748	0.4029	0.2815	0.2390
LinearSGD	745	12328	299	4539	0.2627	0.3747	0.2650	0.2281
NaïveBayes	846	16372	198	495	0.2499	0.4142	0.2600	0.2087
kNearestNeighbour	503	10380	541	6487	0.2333	0.2508	0.2189	0.2022

Table 8.24: Evaluation of the examined machine learning algorithms applied on the TREC 2011 corpus

Algorithm	TP	FP	FN	TN	Precision	Recall	F1
Passive Aggressive	1691	41456	3041	0.1034	0.2175	0.1046	0.0871
Random Forest	2873	72106	1859	0.0987	0.3448	0.0990	0.0747
Plat SMO	3328	73089	1404	0.1067	0.3822	0.1074	0.0792
StochasticMultLogReg	117	2742	4615	0.0687	0.0214	0.0490	0.2221
LinearSGD	4451	76634	281	0.1210	0.5123	0.1239	0.0800

Table 8.25: Evaluation of the examined machine learning algorithms applied on the Events 2011 corpus

performed best, but was followed closely by the AverageScorePerQuery and the AverageScore approaches. In terms of F1-measure StatsPerSourceMatchesPerQuery did best, but all approaches are rather close together. Thus, here the same advice as with the term weighting scheme can be given. Depending on the focus, precision or recall, the appropriate threshold can be chosen from the table. In terms of recall the choice is hard, but the recall values give good advice on which strategy to choose.

Temporally unambiguous

The temporally unambiguous profiles again showed high precision and recall values. The percentile approach, where the score of a new document must be within the top 10% documents scored best. Recall was rather low compared to schemes, thus the F1 measure was not the best. Again, all term threshold schemes performed well here. The best F1 measure was yielded by the AverageScorer.

Temporally ambiguous

Finally, this profile category also showed the same pattern as for the term weights. The values of precision and recall are not as high as for the temporally unambiguous ones, but they are significantly higher than for the atemporal ones. The HarmonicMeanThreshold, closely followed by the AverageScorer, yielded the best performance in precision and F1 measure. Again the AverageScore approach on average yielded the best results for this profile category and is thus the recommended threshold, if score-based thresholds are used.

ML classifier

Finally, the performance of the classifiers per category for the TREC corpus is considered[41]. In terms of precision for the atemporal category, all classifiers yielded more or less the same values. In terms of recall, there were major differences, where the RandomForest approach yielded a recall value of 0.4178 compared to the Passive Aggressive classifier that yielded only 0.1948. But on the level of the absolute values, i.e. true positives, false positives, etc. the picture changes. Appendix E table E.5 shows that for the atemporal category the classifiers fail to learn a decent model, because all of the classifiers produce a large amount of false positives. This supports the notion that atemporal profiles are hard to address without special treatment.

For the temporally unambiguous category the picture is the same as for the score-based threshold. The values are rather good in terms of precision and recall. Also the false positive values are decent, especially when compared to the false positive rate for the atemporal category. Compared to the scored based threshold, the best classifier, StochasticMultinomialLogReg, performed as well as the best thresholds. This indicates that the classifiers can perform reasonably well with the limited volume of training data; with more training

[41] Details for the Events corpus can be found in appendix E.

Threshold	Atemporal			Unambiguous			Ambiguous		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
AveragePositiveScorePerQuery	0.1644	0.2100	0.1654	0.4112	0.2825	0.3672	0.2426	0.2028	0.2306
AveragePositiveScoreStrategy	0.1979	0.1496	0.1558	0.4457	0.2369	0.3586	0.3173	0.2093	0.2556
AverageScore	0.1672	0.2769	0.1691	0.4447	**0.3913**	**0.4195**	0.3837	0.3496	0.2770
AverageScorePerQuery	0.1615	0.2882	0.1691	0.4074	0.3655	0.3963	0.2743	**0.3548**	0.2418
GaussianExponentialThreshold	0.1598	**0.2942**	0.1698	0.4152	0.3850	0.4080	0.2619	0.3214	0.2349
HarmonicMeanThreshold	0.1700	0.1544	0.1483	0.4594	0.2182	0.3533	**0.3912**	0.2145	0.2780
Percentile	0.1770	0.0973	0.1158	**0.4975**	0.1813	0.3094	0.2507	0.1846	**0.2943**
PercentilePerQuery	0.1863	0.1336	0.1458	0.4445	0.1649	0.3294	0.2815	0.1370	0.2269
StatsPerSourceMatches	**0.2012**	0.1362	0.1452	0.4257	0.1620	0.3099	0.2761	0.2168	0.2420
StatsPerSourceMatchesPerQuery	0.1737	0.2109	**0.1717**	0.3880	0.2902	0.3579	0.3172	0.1876	0.2489
StddevOverallScore	0.1731	0.1230	0.1375	0.4550	0.2120	0.3350	0.2832	0.2209	0.2502
StddevPerQuery	0.1742	0.2158	0.1708	0.4123	0.2826	0.3743	0.2751	0.1640	0.2481
TopKThreshold	0.1653	0.0919	0.1478	0.4556	0.1905	0.3573	0.3040	0.1596	0.2703
AveragePerTotalMatches	0.1642	0.2785	0.1675	0.4154	0.3378	0.3821	0.3183	0.3257	0.2491

Table 8.26: Comparison of score-based threshold performance on temporal profile category level for the TREC 2011 corpus.

data this would probably improve further.

The performance of the classifiers on the temporally ambiguous category in terms of precision is also in the same range as the score-based thresholds. Yet again, the recall values are higher, but do also come with a larger false positive rate than the score-based thresholds.

In summary, it could be shown that the studied classifier algorithms were only able to learn a satisfactory model for the temporally unambiguous queries. For the other two categories the model produced too many false positives. This can either be addressed by more training data, or by learning a model for each profile category separately, as it is probable that the features for each profile category have different properties. Summary

8.4.5 Summary

In this section score-based threshold mechanisms and machine learning classifiers were studied in terms of their filtering performance. It could be shown that the AverageScorer, i.e. the scorer that only averages over all scored documents within a sliding window, performed well in terms of precision and recall, although it is the simplest approach. This could be shown for both studied corpora and for almost all profile categories. Nevertheless, the results from the Events corpus are of limited validity due to the poor quality of the relevance assessments, which made it hard to really judge the performance of the threshold mechanisms. The category level evaluation for the Events corpus did not as expected show any differences in performance, due to the different temporal nature of each profile category.

The specialised threshold strategies, i.e. the strategies that monitor statistical values of specialised score sub-streams, did not show a superior performance. But this is probably due to the scarce data. E.g. if a threshold is to monitor the average score of all documents with three query matches, two feedback matches and zero links, it is logical that a reasonable threshold needs several data points, which are not available if a search profile only filters a few documents at all. As the average number of relevant documents for both corpora per topic is between 60 for the TREC corpus and 202 for the Events corpus, this is logical. Nevertheless, the approach to use specialised thresholds worked well in some cases and it seems promising to investigate this approach in larger corpora.

Finally, the comparison with the machine learning approaches showed that in terms of precision the AverageScorer performed better. In terms of recall the machine learning approaches performed better, but if a look on the actual

Threshold	Atemporal			Unambiguous			Ambiguous		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
kNN	0.1468	0.3840	0.1629	0.3610	0.4166	0.3609	0.2541	0.4888	0.2457
LinearSGD	0.1529	0.3324	0.1616	0.3735	0.4172	0.3728	0.2767	0.4076	0.2495
NaïveBayes	0.1368	0.2137	0.1325	0.3438	0.2647	0.3088	0.2287	0.3021	0.2060
Passive Aggressive	0.1492	0.1948	0.1574	0.2514	0.1721	0.2446	0.2274	0.2520	0.2281
Plat SMO	0.1464	0.3824	0.1622	0.3614	0.4205	0.3625	0.2564	0.4961	0.2475
Random Forest	0.1533	0.4178	0.1712	0.3755	0.4434	0.3824	0.2455	0.5069	0.2403
StochasticMultinomialLogReg	0.1678	0.3688	0.1815	0.4018	0.4320	0.4010	0.2656	0.4462	0.2480

Table 8.27: Comparison of classifier performance on a temporal profile category level for the TREC 2011 corpus.

figures of true positives and false positives are considered, it becomes obvious that the machine learning model produced too many false positives. This means the model were not capable of discerning relevant from non-relevant documents.

In summary, it could be shown that in a scenario where training data are scarce and relevance assessments are dubious or of poor quality, a straight forward average score threshold is sufficient and superior. When larger data volumes are available more specifically more reliable training data, the machine learning approaches will probably outperform the simpler score-based approaches. Thus, a repetition of this study would be interesting, when a corpus with several billion documents and several million well curated relevance assessments is available.

8.5 Relevance feedback and query expansion

8.5.1 Introduction

The principles of relevance feedback and query expansion were introduced in Chapter3.3.5. To summarise, the goal of relevance feedback in information retrieval is to adjust the initial scoring of documents by incorporating information provided by the user (real feedback) or the system itself (pseudo relevance feedback).

The special challenge in information filtering is that no ordered list of search results is available that could be re-ordered, but only the document itself, as well as some decision rules to decide whether the document is relevant or not. Therefore, the relevance feedback information can only be used to improve the initial scoring of the document and thereby help a potentially relevant document to cross the decision rule border. Thus, the goal of relevance feedback is to increase the precision values of a filtering system.

Query expansion is used to augment the initial search profile by adding new terms to the profile, thereby increasing the number of documents that can be subject to filtering. The challenge in a filtering scenario is that new documents arrive over time; therefore the algorithm that is used for query expansion should be able to adapt to this temporal aspect. The goal of query expansion is therefore to increase the recall of the system.

But both goals, precision and recall, are negatively correlated. This means if the quality of one factor increases the other one usually decreases. So if the user wants more high-quality results the filtering factors of the system are

stricter. If the user wants more varying results that are related to a topic, the restricting factors are loosened.

In general, information on the internet is abundant and many sources report on the same topics, hence there is usually no shortage of relevant documents. The problem is rather to find the relevant ones within the plethora of documents that are continuously being generated. Therefore, an information filtering system should primarily be focused on precision rather than on recall (Carpineto & Romano, 2012, p. 2). Nevertheless, there are several occasions where a good recall value is desirable. For instance, if a search profile only yields few results, the initial query must be widened. Also, in legal cases it can be required to find as many documents that are relevant as possible, in order to cover every aspect of the subject matter. Furthermore, intelligence services also want to cover every potentially relevant document, if possible. In the case of this dissertation, the first case is the most relevant.

Various feedback algorithms for information retrieval exist, but as with the term weights and thresholds, they need to be re-evaluated in the context of highly dynamic sub-corpora. Additionally, a new feedback algorithm is presented in this section that explicitly tries to address this goal. In order to verify its validity, its performance in filtering relevant documents is compared to various state-of-the-art relevance feedback algorithms.

8.5.2 The ReNove feedback algorithm

In high-volume text streams a relevance feedback algorithm should address novelty and recency as well as the re-occurrence of terms. Therefore an algorithm called *ReNove*, which addresses *Repetitiveness* as well as *Novelty* of feedback terms, is introduced that explicitly models the demands of high-volume text streams. The goal of this algorithm is to address the various types of search profiles that were presented in (R. Jones & Diaz, 2007, p. 14f) and where search profiles are categorized as either *atemporal*, *temporally unambiguous* or *temporally ambiguous*.

The rationale behind ReNove A relevance feedback algorithm in a highly dynamic and time sensitive area like text streams should first take recency and the temporal relation to the timestamp of the query issuing into account. Second, the burst of events, i.e. the cumulative occurrences of terms within a short period of time, should also be incorporated. Finally it must be considered that an information filtering query is a standing query that expresses a long term information need and therefore a feedback algorithm should also integrate rare or recurring terms.

The first component to address a temporal relation to the timestamp of the issuing of the information need, is the *query decay value*. In the ReNove formula this is an exponential decay value calculated by using the temporal distance between the timestamp of the query, when it was created, and the currently evaluated document scaled by a factor λ. This corresponds to other *query decay-based* methods presented in (Li & Croft, 2003), (Hong et al., 2013), (Miyanishi et al., 2013, p. 4) and (Efron & Golovchinsky, 2011).

The second component factor is the *feedback term interarrival time decay*. This is another exponential decay-based component using the temporal distance between the occurrence of a certain feedback term fb as the exponent again scaled by the factor λ[42]. This approach is somehow similar to the content *nutrition/energy* concept presented in (Cataldi et al., 2010) in terms of "re-energizing" the importance of a term, if repeatedly encountered by the system. But their approach focuses on the calculation of a term score-based on the squared difference of the product of a user authority weight and a local term weight component at two distinct points in time.

This component can also be referred to as the *hotness* of a topic, because if a topic is *hot*, i.e. a topic is frequently discussed using certain terms within a time slot, the interarrival time of these terms is also very short and the value remains relatively high and drops off, if the burst is over. A method called *BursT* presented in (Lee et al., 2011) uses the average of the term interarrival times to calculate the burst/hotness score of a document. But their method represents a moving average over the interarrival times, while the component here uses an exponential decay between two timestamps. The advantage of the latter is first the easy calculation and second the high sensitivity to temporal changes, because a value based on exponential decay is easily "re-energized", i.e. the score increases fast, if the interarrival time is short, and drops off fast if the burst is over.

The third component addresses the long-term information need of a query in information filtering. After the *hot, bursty* phase(s) is/are over certain terms may still occur regularly, but not frequently or in an accumulated way. Nevertheless these terms are of interest, as they usually represent a long-term, inherent aspect of a topic and this is important to information filtering. Therefore the last component uses a sliding window to keep a moving average of the *variances* of *interarrival times*. The *variance* describes the spread of data. The lower the variance, the closer this value is to the mean. Thus a low variance means a regular spread around the mean of a value, in this case the interarrival time of a term. An even spread of the interarrival time under-

[42]The hyper-parameter λ has the same value for both components. During the experiments the value of .00001 was found to deliver the best results.

lines a regular occurrence of a term over a period of time and yields a higher score for such recurring terms. Hence, this component is suitable to address the long-term information need represented by the query terms of a search profile.

Two version of ReNove are studied. One, *Extended ReNove*, that incorporates all three mentioned components, whereas the other, called *Regular ReNove*, only incorporates the interarrival time and the variance component. The latter is due to the fact that during the experimentation the ReNove version with the two components often outperformed the version that contained all three components and it was considered as relevant to report this result as well, This is interesting, because at first sight it may seem logical that an algorithm that addresses all relevant aspects of relevance feedback in a text stream should do better than one with only two components, But it seems that the *query decay factor* introduces an initial bias, which keeps terms that are no longer in the top k feedback terms that may not be very useful in the end. This prevents other terms that may be relevant in end, but not right at the time of issuing of the query, from making it to the top k feedback results.

Thus the *Extended ReNove* formula is

$$score(ER) = score(QD) + score(NOV) + score(REP) \tag{8.6}$$

and the regular ReNove

$$score(ReNove) = score(REP) + score(NOV) \tag{8.7}$$

with the query decay score $score(QD)$

$$score(QD) = \exp^{-\lambda \cdot (t_Q - t_i)} \tag{8.8}$$

where t_Q is the time stamp when query was issued and t_i the time stamp of the currently evaluated term. λ was empirically to be chosen as 0.00001, because with this value a gap of eight hours is as low as 0.0561 and hence quite marginal. The $score(novelty)$ is calculated as

$$score(NOV) = \exp^{-\lambda \cdot (t_i - t_{i-1})} \tag{8.9}$$

where t_i is again the timestamp of the currently evaluated term and t_{i-1} the timestamp of the penultimate occurrence of the same term. The value of λ is the same as for the *query decay* score.

Finally the *repetitiveness* component is calculated as the sliding average of the

variance of the interarrival gaps of a term.

$$score(REP) = [\,(t) + 1]^{-\epsilon} \tag{8.10}$$

ϵ is empirically being chosen as 0.1, because a smaller value by factor ten would not decrease the score even for very small values, while a factor larger than 0.1 would decrease the score too quickly. The moving average is used, because otherwise a few high gaps would spoil the value for the rest of the lifetime of the feedback term.

The *regular ReNove* algorithm only incorporates the *novelty* and the *repetitiveness* score.

Visual explanation of the rationale behind ReNove To better understand the influence of the three factors on the score of a feedback term, three hypothetical feedback term occurrence patterns will be presented as well as their score progress over time. The goal is to illustrate the idea of ReNove and how the combination of the two or three factors affects the *relevance feedback score* of a term at each point in time. The occurrence of a new document is marked by the dots on the lines.

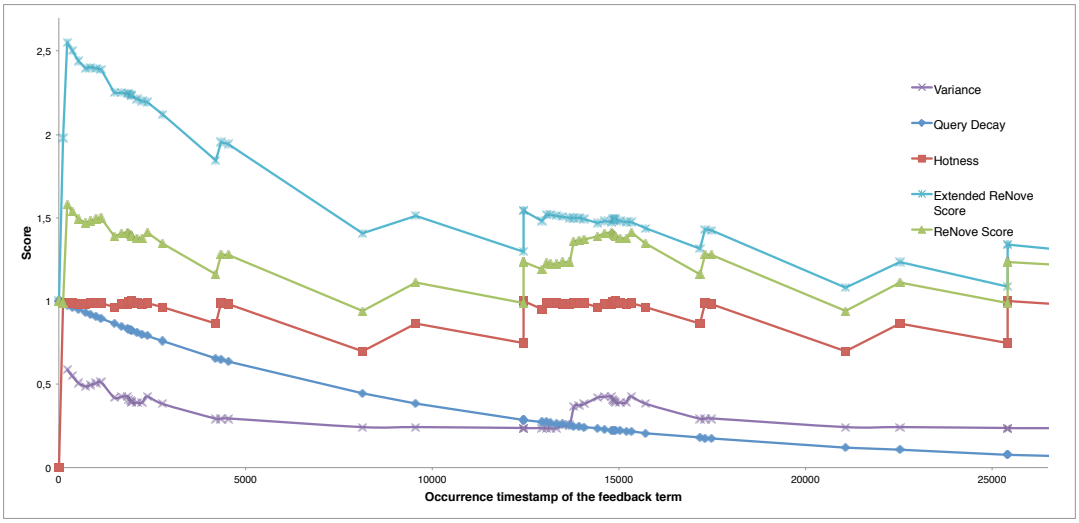

Figure 8.17: Lifetime of an unevenly spread feedback term

Figure 8.17 shows the score process of a term that appears unevenly spread over time. This means in this example that there is an initial peak with lots of term occurrences in the beginning. Afterwards the term only occurs a few times, again followed by a bursty period, finally there is a long period where the feedback term hardly occurs at all.

The graph contains the values for all three components as well as the scores of the extended and the regular version of ReNove. This graph shows that ReNove adapts well to all circumstances at the beginning, starting from the issuing of the query. The *query decay* score is high, of course, as well as the *hotness/novelty* factor and *repetitiveness* factor. While the query decay is logically steadily decreasing and thus losing influence on the relevance of the term, the other components succeed in modelling the temporal occurrence well. The hotness score with a maximum of one peaks when events occur in a bursty fashion and drops off slowly when less events arrive. The hyper-parameter λ can be used to adjust this drop off. The contribution of the variance component in the middle of the chart is interesting in this scenario. Here a bursty period yields an increase in this component, but not as dramatically as the hotness score. Nevertheless it contributes significantly to the score and thus can help to move a term into the top k feedback terms. It is more interesting to compare the score of the extended vs. the regular version. In the extended version the query decay is still present and thus the increase of the variance component is marginalized by the query decay decrease. The regular version in contrast shows a score increase due to the fact that the variance component is not marginalized.

Figure 8.18 shows the score evolution for a term that occurs very regularly in the feedback documents. The query decay component drops off very quickly and thus terms are quickly ignored for relevance feedback and query expansion. The ReNove algorithms also keep a high level for the term, but when the interarrival time starts to oscillate the algorithms recognize this and penalize the score values. In contrast, the hotness value remains constantly high and is insensitive to the slight changes in the interarrival gaps.

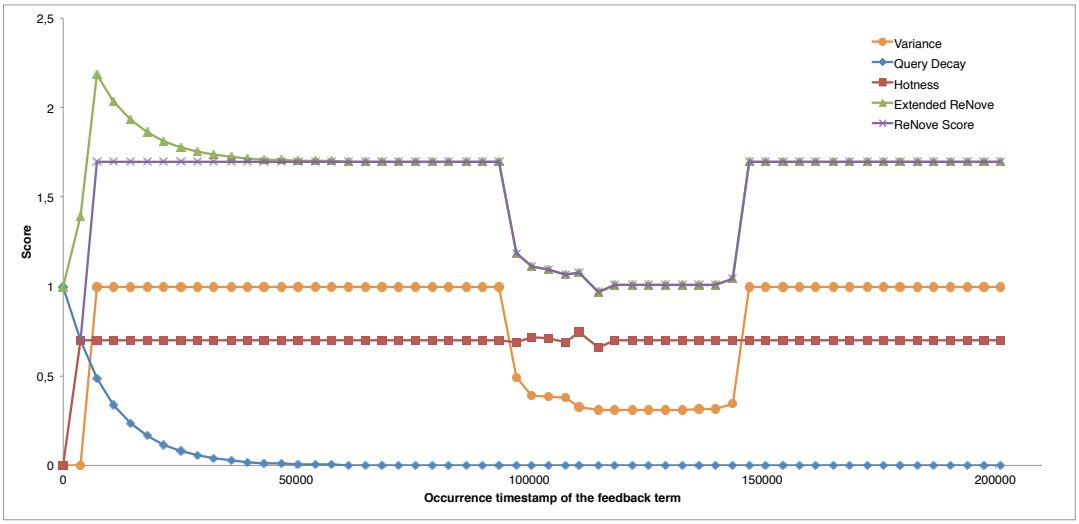

Figure 8.18: Lifetime of an evenly spread feedback term

The final graph shows the lifetime of a peaking feedback term. This means the term appears in a short bursty period and after that only very sporadically. The variance component in this case helps to keep a minimum score level, which increases the probability of the term making it into the *topk* feedback term and thus a candidate for a new filter event processing agent.

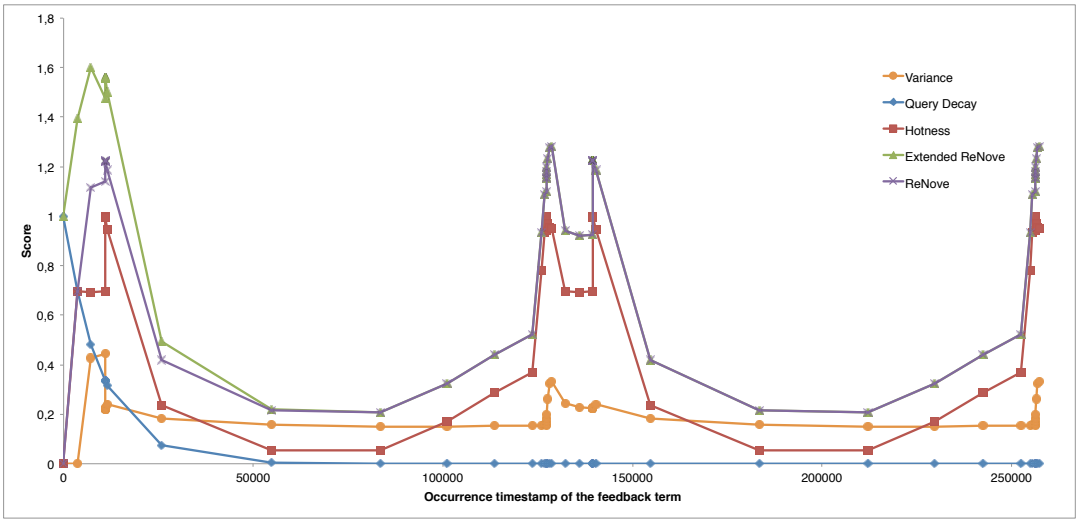

Figure 8.19: Lifetime of a peaking feedback term

8.5.3 Relevance feedback and query expansion based filtering process

The integration of relevance feedback and query expansion into *StreamFI* works as follows. The initial search profile, which consists of various filter terms, filters out candidate documents for scoring. After these documents have passed the query matching and filter policy component of *StreamFI*, the documents are scored using the baseline scoring algorithm *Okapi BM25*. The scored documents are then sent back into the event processing network.

A *Relevance Feedback EPA* listens for these scored documents. It only filters documents whose score is above the threshold, because only such documents were judged as relevant by the system and can thus be shown to the user.When it receives a new document that passed the threshold, it looks up the available relevance information. If such information is available, the terms from the scored document are extracted and saved in the feedback terms container, which associated with search profile, including the pertaining category (true positive or false positive)[43].

[43]To evaluate the recall effect of the ReNove algorithm the condition that the score must

In the next step, the feedback algorithm updates the scores of the feedback terms contained in the feedback terms container and the top k feedback terms are selected from the container[44].These feedback terms are then used to created a new dynamic[45] *Document Scoring EPA*. The filter string of these event processing agents consists of the initial filter string and the newly added feedback term.[46] Dynamic *Document Scoring EPAs* are limited in their lifetime. This means that if they do not match a document within a configurable period of time, they are deactivated. This approach helps to limit the required resources and minimizes the event processing effort.

8.5.4 Studying the effect of relevance feedback and query expansion on filtering performance

In order to assess the quality of the proposed ReNove algorithm an empirical evaluation is conducted. For this purpose a comparison against other feedback algorithms is required. Therefore, in this section ReNove will be compared against four algorithms, namely *Rocchio*, Lavrenko's *Relevance Model*, a *Kullback-Leibler* score and a temporal *decay-based* algorithm.[47].

Four aspects will be studied in detail:

1. The ability of each algorithms to identify potentially relevant documents from the stream.

2. The effect on the precision and particularly on the recall of each algorithm.

exceed the threshold was removed, as otherwise components like scoring and threshold determination would have had an effect at a stage were only the effectiveness of the relevance feedback component is of interest. This condition was of course reactivated for evaluation of the other components.

[44]This is important to remember when the temporal profiles of the various feedback algorithms is studied, because an algorithm that is not very time sensitive could block the top k feedback terms with the initial ones for the rest of the lifetime of the query.

[45]Dynamic for two reasons. First the event processing agent is created dynamically based on the feedback and second it gets removed from the event processing network after a certain period of time. This is to lower the resource consumption.

[46]In this stage of the evaluation every scored document is evaluated and the *Relevance Feedback EPA* is able to access the relevance information for every document. In a later stage and in a real world scenario this is not the case, because feedback is only available for documents that have crossed the decision boundary and been presented to the user (real feedback) or the system (pseudo feedback). Every document below the threshold is discarded and therefore not available to the system. But as stated above the goal is to investigate the effect of the feedback algorithms and therefore such restrictions are removed.

[47]Details on these algorithms can be found in chapter3.3.5. In summary, the algorithms were chosen by their scientific relevance and how well they fit into a streaming scenario.

3. The performance of each algorithm on each topic.
4. The quality of the generated feedback terms on the topic level.

Total number of relevant documents The total number of relevant documents is defined as the sum of all *true positive* and all *false negative* documents, i.e. all documents that were presented to the filtering algorithm and judged as relevant or non-relevant by the algorithm while being in fact relevant. Usually only recall is studied, but in order to evaluate the performance of a feedback algorithm the *false negative* number should also be taken into account, because this number represents documents that were selected from the stream and processed by the filtering algorithm, but *just* missed the decision boundary. Nevertheless, these documents could once pass the decision boundary, if an adjustment in term weighting, filter policies or threshold determination is made, and this adjustment happens to help converting the initially *false negative* document into a *true positive*. This chance does not exist for documents that are not even selected from the stream. Thus this number is relevant, as it shows how well an algorithm is suited to generated relevant feedback terms.

TREC 2011 corpus

The subset of the *TREC 2011 corpus* used for this evaluation contained a total of 1,812 relevant documents. Figure 8.20 shows the number of relevant documents that were filtered by the various algorithms using the top 50 feedback terms. It is obvious that all algorithms outperformed the baseline run without query expansion by at least 50%. ReNove even yielded more than 66% more relevant documents. The simple ReNove version called *RePet*[48] yielded 61.29%. The algorithm based on Kullback-Leibler divergence encountered the second highest number of documents. Relevance Model yielded 55.14% improvement. This is interesting as this approach does not take into account any available relevance information but just derives relevance assumptions from the available term probabilities. The classic Rocchio approach did also not score badly, even though not originally intended for use in high velocity text streams. Surprisingly the decay-based approach performed worst, even though incorporating some kind of exponential decay is a popular method in data streams (Cormode et al., 2009, p. 138). To summarise the probability based Kullback-Leibler divergence worked well. This supports the notion that the dynamic sub-corpora provide enough statistical information

[48]The name is derive from repetitiveness, as this feedback algorithm only uses the *score(REP)* component of ReNove

Algorithm	TP	FN	FP	TN	Total rel. found	Improvement
Renove	773	391	3076	4703	1164	166.29
KL	708	440	3399	5079	1148	164.00
Repet	729	400	2636	4635	1129	161.29
Relevance Model	663	423	2695	4895	1086	155.14
Rocchio	554	520	3251	4380	1074	153.43
Decay based	599	466	2871	4989	1065	152.14
without	331	369	684	968	700	100.00

Table 8.28: Total relevant found documents of TREC 2011 corpus using top 50 feedback terms. Statistically significant improvement of ReNove with p < 0.01 in bold.

that such an approach works efficiently. ReNove in contrast does not rely on any probabilities. Rather the smart combination of interarrival gaps (inspired by (Lee et al., 2011) and extended by a smoothed variance measure) and an exponential decay which is independent of the query time (like the decay-based algorithm) also yield substantially better feedback results.

In addition to overall figures – precision and recall will be studied in the next section – it is also interesting to evaluate the number of *true positives* yielded by each algorithm. And here the real advantage of ReNove based algorithms becomes visible. All feedback algorithms runs used the same setting in terms of term scoring, thresholding and filter policies, but the ReNove algorithms yielded the highest rate of *true positives* and the lowest of false negatives. Additionally, the *false positive* rate of the regular ReNove algorithm is the lowest among all algorithms. Thus, these figures underline the assumption that the ReNove algorithms are well suited for use in the further investigation of the *StreamFi* filtering system.

All improvements were also checked for statistical significance using a χ^2 test. This means the *true positive* and *false negative* values of ReNove were compared against the values from the other algorithms. In terms of improvements over the baseline run without any relevance feedback, all algorithms showed statistical significance with a p-value less than 0.01. The χ^2 tests comparing ReNove to every other algorithm also returned a p-value of less than 0.01.

Events 2012 corpus

The experiments also used the *Events 2012* corpus. Again the top 50 feedback terms were chosen, and table 8.29 summarises the results for these runs. The

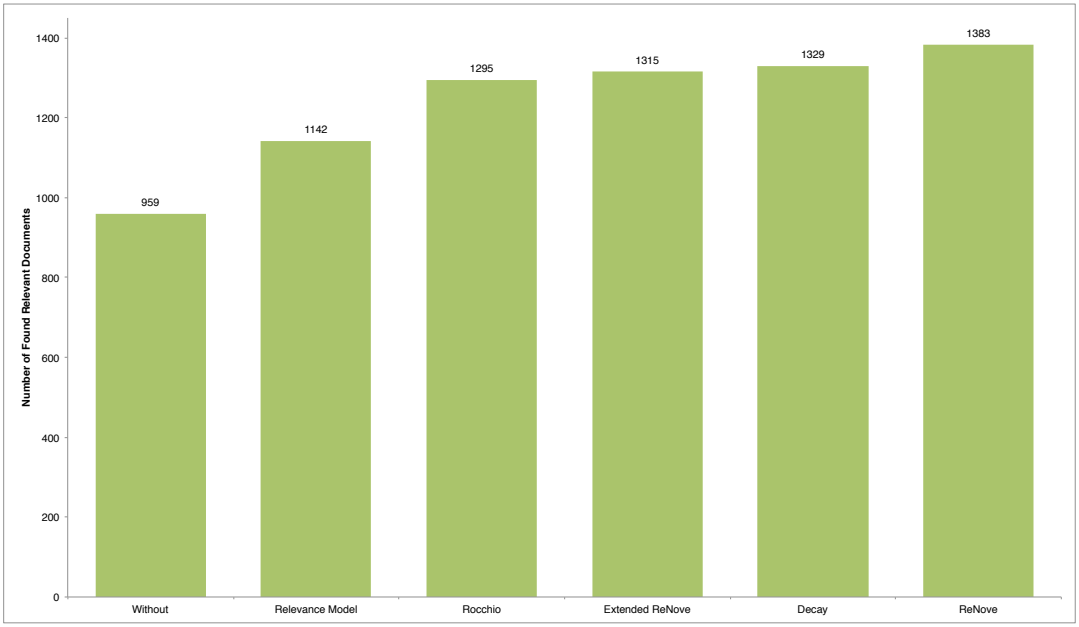

Figure 8.20: Total Number of Found Relevant Documents Per Algorithm for TREC 2011 Corpus (Top 50 Feedback Terms).

numbers for the conducted runs supported the assumption that ReNove is capable of finding more relevant documents within the corpus than the other algorithms. All improvements of ReNove in terms of increasing the number of *true positives* and false negatives were again statistically verified using a χ^2 test. This time all test of ReNove against non-ReNove algorithms returned a $p - values$ of less 0.01, thus the hypothesis that ReNove finds significantly more relevant documents was accepted. It is interesting to see that ReNove is better capable of finding more relevant documents in a corpus with a larger number of documents and more relevance judgements. This implies that ReNove requires a certain event density to perform well. It is also interesting that the decay-based approach returned the least number of relevant documents while being the second best in class for the *TREC 2011* corpus.

The results throughout the experimentation showed that both ReNove versions succeeded in returning the largest number of potentially relevant documents of all algorithms for both corpora. Even though the results of ReNove did not significantly outperform the comparison algorithms, it nevertheless yielded the best performance. This is also notable due to the aforementioned quality of the relevance assessments of the Events 2012 corpus. Therefore, the results of this section can be regarded as the first indicator of the theoretical design of ReNove being valid. But an evaluation based only a single figure does reveal for which profiles the algorithm truly works. However, on this level of evaluation the finer details of the intrinsic functioning of ReN-

FB Algorithm	TP	FP	Rel Docs Found	p-Value
ReNove	3249	1472	4721	
Extended ReNove	3182	1519	4701	< 0,0896
Relevance Model	3358	968	4326	<0,01
Rocchio	3060	1189	4249	<0,01
Decay	3098	1093	4191	<0,01
Without	2767	946	3713	<0,01

Table 8.29: Total relevant found documents of Events 2012 corpus using top 50 feedback terms. Statistically significant improvement of ReNove with p < 0.01 in bold.

ove are still opaque. Therefore, the next sections will take a deeper look at the per-topic performance. Some topics will also be studied in more depth.

Recall improvements in detail This paragraph describes the effect of query expansion on recall, in terms of overall recall and also in detail on a topical level. While the *total number of relevant documents* is an indicator for the *potentially* relevant documents, recall is a well-defined *information retrieval* standard evaluation factor. This paragraph also shows that the superior performance of ReNove is not attributed to a single topic, but the effect is observable for more or less all topics.

The evaluation of the recall value is also important, because changes in the recall value among the algorithms show if the encountered feedback terms were of good quality. This is because the recall value can only increase if a document crosses the defined threshold. Only good quality feedback terms are supposed to provide a decisive contribution to the score and thus turn a document into a *true positive* instead of a *false negative*. Additionally, an increase of the recall value without further tuning of other parameters would also suggest that the algorithm is better capable of identifying score relevant terms.

TREC 2011

Figure 8.21 shows the average precision and recall values for the *TREC 2011* corpus.[49]. While the increase of relevant documents and thus the recall value

[49]The results for topics 12 and 49 were excluded as they had very little relevant documents (1 and 4) and yielded a recall value of 1. Furthermore topics 15, 18, 33 were excluded as for all algorithms the recall value was 0.

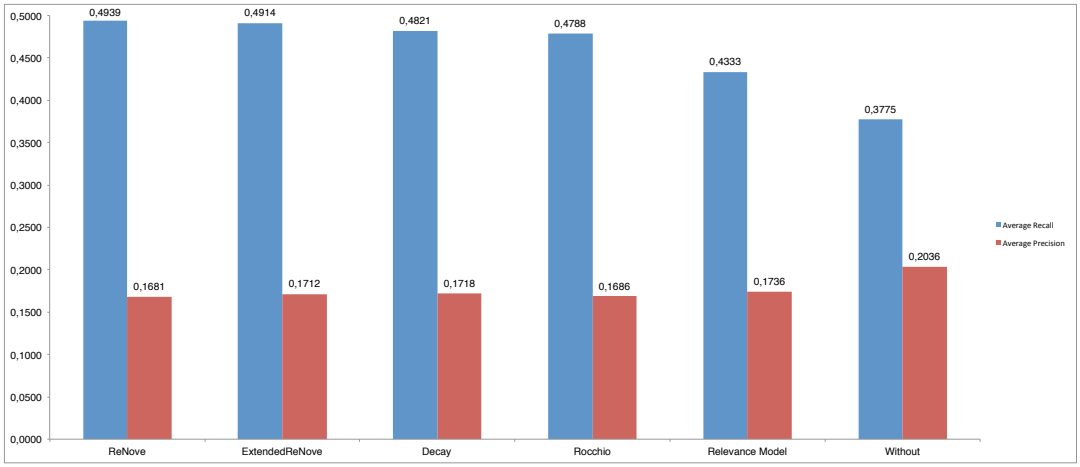

Figure 8.21: Recall and precision values - TREC 2011 corpus

increases from .3174 for the run without relevance feedback and query expansion to .4497 using ReNove, the precision value logically drops from .2501 to .1790. It is interesting to see that the difference between the best four algorithms is only marginally on average, but this could be due to the fact that a single topic raises the average. Therefore a detailed overview of the recall values per topic is shown in figure 8.22. The colouring helps to provide an immediate impression of the performance per algorithm. This analysis also shows that ReNove is capable of outperforming all other algorithms. Even though the difference is not high, it is nevertheless observable and as almost all topics receive more relevant documents from the ReNove algorithm it is valid to conclude that it is a reasonable feedback algorithm.

In-depth analysis of generated feedback terms This section studies two topics of each corpus in more detail. In each case, the best and worst performing ReNove topics were chosen and compared against the results from the algorithm that performed best at this topic. Only topics with a balanced result profile were considered, i.e. a reasonable number of true positive and of relevant documents must be available; profiles with extreme recall values like 0 and 1 were ignored. The goal is to illustrate with real world data the feedback terms that are generated by each topic and when they are generated.

TREC 2011

For the in-depth analysis topic 29 (*global warming weather climate change*, positive recall example) values and topic 37 (*Gifford's recovery*, negative recall examples) were chosen. The query for topic 29 was issued on 23 January

Topic	Decay	ExtendedReNove	Relevance Model	ReNove	Rocchio	Without
2	0,913	0,913	0,913	0,913	0,913	0,913
3	0,6667	0,7037	0,7037	0,7407	0,7037	0,6296
4	0,3111	0,3222	0,2889	0,3111	0,2889	0,2667
5	0,7778	0,8889	0,5556	0,7778	0,7778	0,5556
7	0,5135	0,5405	0,5225	0,5225	0,5315	0,4505
8	0,5778	0,5889	0,5556	0,6	0,5889	0,4778
9	0,5478	0,5391	0,3826	0,5565	0,5478	0,1565
10	0,5484	0,4839	0,3548	0,5484	0,4839	0,2903
13	0,5	0,5	0,4545	0,5	0,5	0,1364
14	0,1545	0,1364	0,0818	0,1273	0,1545	0,0818
17	0,2295	0,2787	0,1639	0,3443	0,2787	0,1475
19	0,6471	0,6667	0,5098	0,6667	0,6667	0,5294
20	0,9808	1	0,9038	1	1	0,7885
22	0,8047	0,7969	0,7656	0,8359	0,8203	0,6797
23	0,2235	0,2353	0,2	0,2235	0,2353	0,2118
24	0,6462	0,6462	0,6308	0,6462	0,6615	0,6308
25	0,1447	0,1579	0,1316	0,1711	0,1711	0,1316
27	0,42	0,44	0,42	0,42	0,42	0,38
28	0,5	0,5	0,5	0,5	0,5	0,5
29	0,4362	0,4574	0,383	0,4894	0,4255	0,3085
30	0,5	0,5125	0,1875	0,5	0,5125	0,175
32	0,5439	0,5614	0,5263	0,5088	0,5263	0,4561
34	0,3	0,35	0,275	0,35	0,325	0,15
35	0,5	0,5	0,5	0,6	0,6	0,4
37	0,3662	0,338	0,3239	0,2958	0,2817	0,1127
38	0,5769	0,5769	0,5769	0,5385	0,5769	0,5769
39	0,0909	0,0606	0,0303	0,0606	0,0606	0,0606
40	0,2353	0,2353	0,2353	0,2353	0,1765	0,1765
42	0,8214	0,8214	0,7857	0,8214	0,7857	0,7857
43	0,8148	0,8148	0,8148	0,7778	0,8148	0,7407
44	0,3043	0,3043	0,3478	0,3043	0,3043	0,3043
45	0,3875	0,4125	0,3875	0,5	0,4	0,3125
47	0,2308	0,2308	0,2308	0,2308	0,2308	0,2308
48	0,1754	0,193	0,0877	0,1754	0,1404	0,0877
Average	0,4821	0,4914	0,4333	0,4939	0,4825	0,3775

Figure 8.22: Recall values per topic - TREC 2011

2011 07:02 AM GMT and for topic 37 on 23 January 2011 01:59 AM GMT.

Figures 8.24, 8.25,8.26,8.27 show the temporal profile of generated feedback terms for the four use algorithms. The black dots mark the occurrence of feedback documents from the corpus. The blue dots are feedback terms generated by the relevance feedback and pseudo relevance feedback event processing agents. The size of the dot indicates the score. The position on the y-axis does not have any specific meaning. The dots are only spread over the y-axis so that they do not overlap.

Even though, as shown above, the overall difference in true positives between the algorithms is very small, the temporal profiles differ significantly. The first thing that becomes apparent is the temporal spread of the the feedback terms. While ReNove succeeds in producing feedback terms with a relevant score over the whole corpus lifetime, the decay-based approach only creates terms with a decent score close to the issuing of the query (23 January 2011 07:02 AM); the subsequent scores are low. This is logical, as the score for a decay-based approach with a fixed starting point (query issuing) rapidly drops due to the chosen exponential function. But despite the score feedback, terms are generated over time by the decay-based algorithm. Interestingly this happens more or less in two phases, the first burst is around 24 January 1 a.m. and the second around 26 January 1 a.m.. After this, feedback

Topic	Decay	ExtendedReNove	Relevance Model	ReNove	Rocchio	Without
2	0	0	0	0	0	0
3	2	1	1	0	1	3
4	0	-1	2	0	2	4
5	0	-1	2	0	0	2
7	1	-2	0	0	-1	8
8	2	1	4	0	1	11
9	1	2	20	0	1	46
10	0	2	6	0	2	8
13	0	0	1	0	0	8
14	-3	-1	5	0	-3	5
17	7	4	11	0	4	12
19	1	0	8	0	0	7
20	2	0	10	0	0	22
22	4	5	9	0	2	20
23	0	-1	2	0	-1	1
24	0	0	1	0	-1	1
25	2	1	3	0	0	3
27	0	-1	0	0	0	2
28	0	0	0	0	0	0
29	5	3	10	0	6	17
30	0	-1	25	0	-1	26
32	-2	-3	-1	0	-1	3
34	2	0	3	0	1	8
35	1	1	1	0	0	2
37	-5	-3	-2	0	1	13
38	-1	-1	-1	0	-1	-1
39	-1	0	1	0	0	0
40	0	0	0	0	1	1
42	0	0	1	0	1	1
43	-1	-1	-1	0	-1	1
44	0	0	-1	0	0	0
45	9	7	9	0	8	15
47	0	0	0	0	0	0
48	0	-1	5	0	2	5
Total	26	10	134	0	23	254

Figure 8.23: Difference of True Positive compared to ReNove - TREC 2011 corpus

events are only generated sporadically. The reason for this is not clear as the log file did not show any extraordinary number of score documents for this period. An in-depth analysis of the decay-based algorithm could be subject to further research.

Furthermore the score – indicated by the size of the dot – shows a high variability for the ReNove algorithm. This is a sign of the algorithm adopting well to the changes in the stream and in the topic. Another interesting aspect is the temporal profile of the Rocchio algorithm. This one fails completed to generate new feedback terms after 26 January and 6 February. It seems that the initial scores generated for the top k feedback terms is so large that other topics cannot make it into the top k afterwards. This is interesting as the scores for the Rocchio algorithm are based on the dynamic sub-corpus approach used throughout this thesis, of course. Hence a high degree of time-sensitiveness is included, but this is not enough to efficiently created good feedback terms.

The temporal profile for the Relevance Model algorithm shows a similar behaviour. There are relatively high scores of feedback terms at the start of the topic, but afterwards few new feedback terms are generated. Then, there is a long gap, after which few new terms are generated. It seems that the Relevance Model approach suffers from the same problems as the Rocchio

algorithm. Although based on a temporal corpus, this corpus is not enough to account for the specificities of a text stream and is thus not capable of efficiently creating new feedback terms.

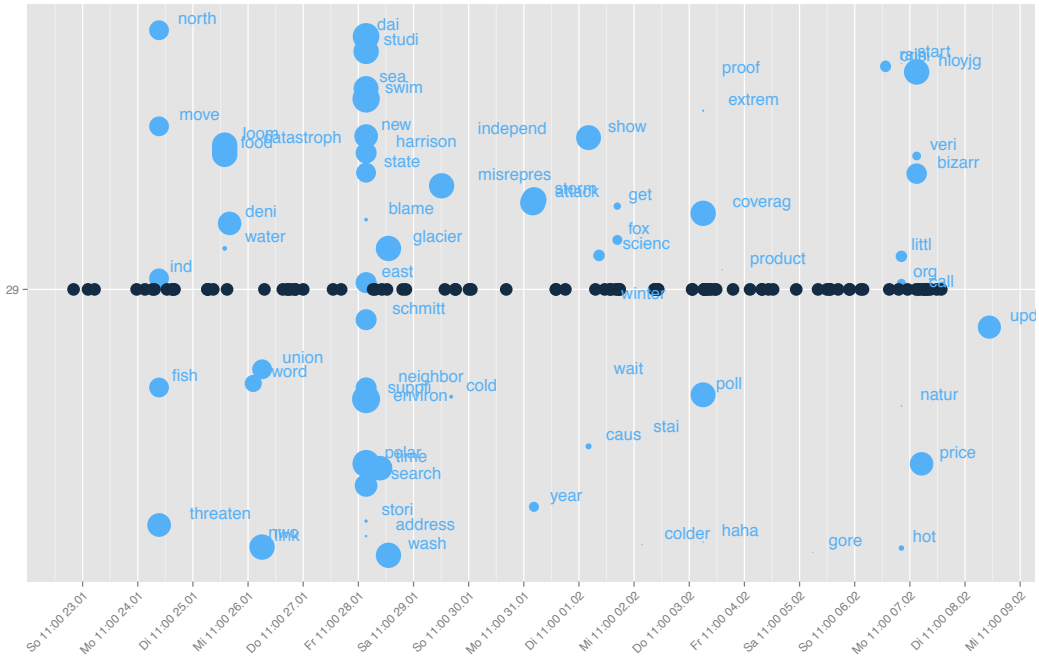

Figure 8.24: Temporal profile of feedback terms generated by the ReNove algorithm - Topic 29 TREC 2011

Summary This section explained the effects and principles of the ReNove feedback algorithm. It demonstrated that ReNove addresses the temporal and dynamic characteristics of text streams well and yields reasonable results in terms of recall and identification of relevant documents from a stream. In an extensive comparison with three state-of-the-art algorithms it was shown that ReNove outperforms each of them significantly.

8.6 Summary and Conclusion

This chapter examined essential components of an event-driven information filtering system, *viz.* filter policies, term weighting schemes, thresholds and query expansion mechanisms. Filter policies were introduced as the gatekeepers to the more computation intensive stages of the filtering process. Their defined goal was to remove as many obviously non-relevant documents as possible. In order to achieve this, three strategies were introduced:

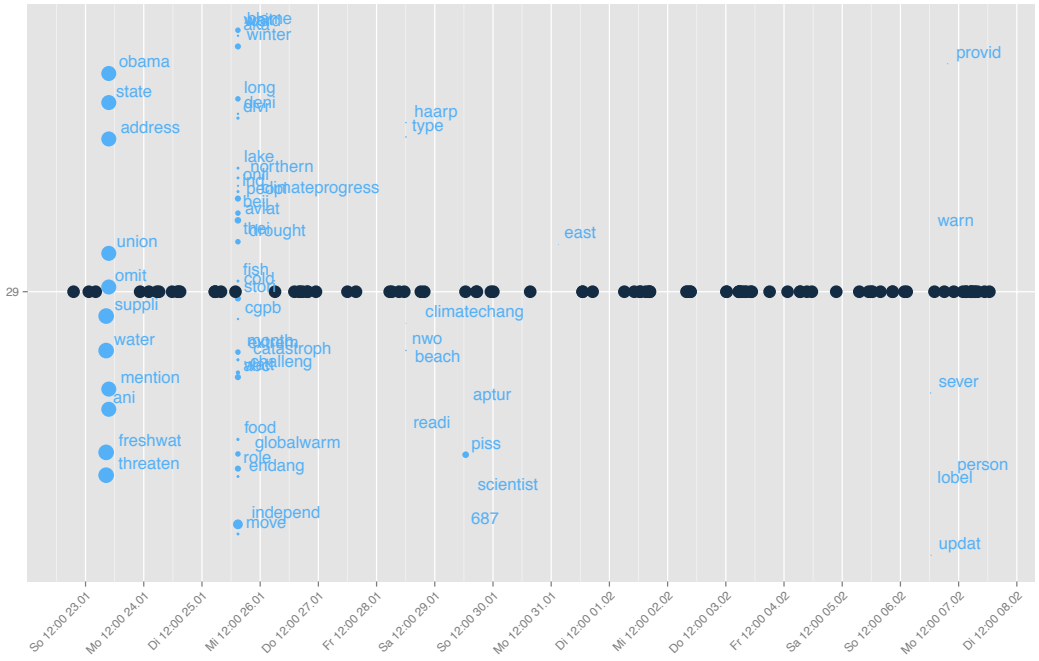

Figure 8.25: Temporal profile of feedback terms generated by the decay-based algorithm - Topic 29 TREC 2011

heuristic filter policies, machine learning based policies using weak classifiers and a frequent pattern mining approach. It showed that a decent heuristic filter is already a good candidate selector for later filtering stages. The weak classifier approach improved precision but also removed many documents and thus recall became weaker. For text streams with lower event density such as the TREC corpus, the weak classifier approach removes too many documents. For text streams with higher event density this approach can be a beneficial candidate selection method. The frequent pattern approach also yielded reasonable results, i.e. in higher density text streams this approach is beneficial. The appealing fact of this approach is that it generates filtering rules that are similar to heuristic filter policies, thus intelligible to humans. They can be studied, analysed and adapted by humans. This is not possible for the weak classifier approach, because the internal representation of the filter policy is encoded in the machine learning algorithm and the calculated parameters.

Then eighteen different term weighting schemes were studied, in order to verify if seminal approaches are also valid in high-volume text streams. It could be shown that the performance of a term weighting schemes depends significantly on the type of profile it is applied to. The best results were gath-

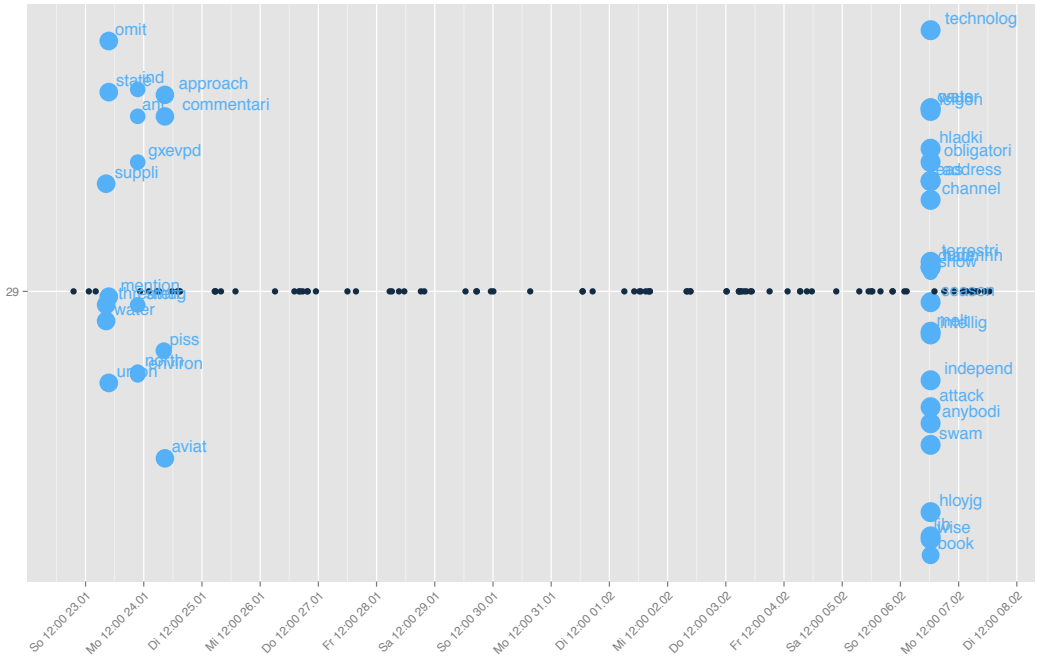

Figure 8.26: Temporal profile of feedback terms generated by the Rocchio algorithm - Topic 29 TREC 2011

ered for the temporally unambiguous search profiles, while for the atemporal profiles the evaluation values were not good. Also it could be shown that some of the newly proposed term weighting schemes, especially the Context Scorer, yielded good results. Interestingly, the dynamic sub-corpus approaches perform often better than the approach to use a static corpus that contained all available documents. This is a strong indicator that the dynamic sub-corpus approach is valid. The evaluation of the Events 2012 corpus showed biased results and underlined the assumption that the relevance assessments for this corpus must be used with caution. Nevertheless, the evaluation results from this corpus also provided valuable insight.

Subsequently, several threshold algorithms were examined. The goal was to find out if linear classifiers, based on thresholds that are calculated on the score of the documents, yield reasonable performance and furthermore, how they perform compared to machine learning approaches. A method to derive specialised thresholds based on different text stream criteria was introduced and as well as an meta threshold algorithm using harmonic means. Percentile and top-k approaches were studied as well. For the TREC corpus, a simple approach that averages the document score and takes this average as a threshold yielded the best results. The machine learning approaches did not

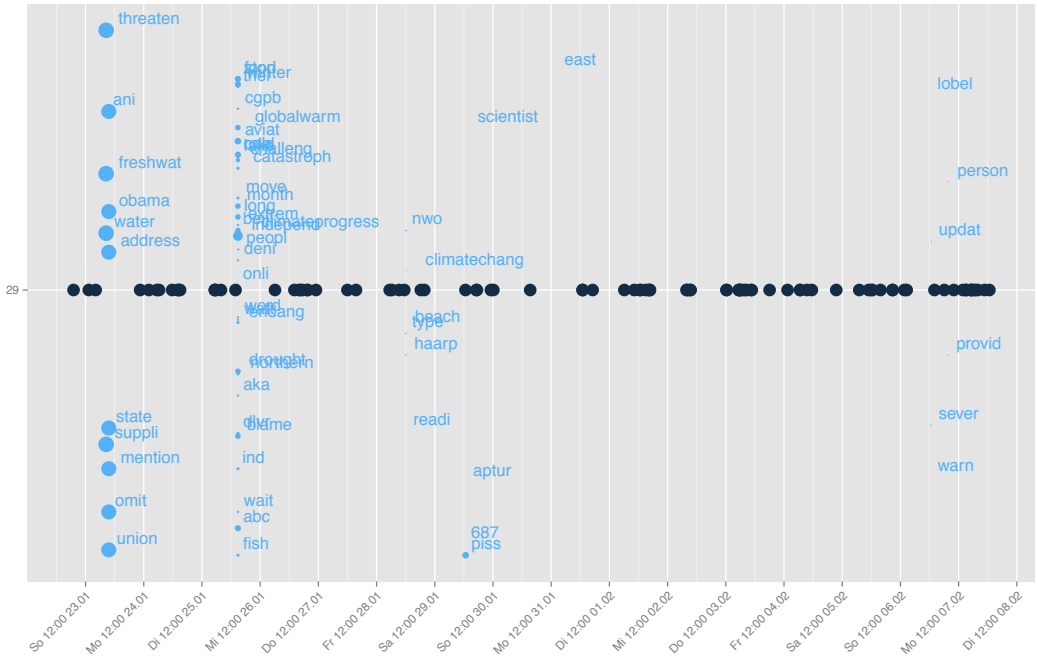

Figure 8.27: Temporal profile of feedback terms generated by the Relevance Model algorithm - Topic 29 TREC 2011

perform any better than the simple calculation based approaches. This does not mean machine learning based approaches can not be superior to simple calculation based ones, but in a scenario were the training data is limited[50] or the relevance assessments are not necessarily of the best quality, it is difficult for a machine learning algorithm to learn to perform well. But in *Big Data times* this problem is about to disappear. Nevertheless, the threshold-based approaches are useful for fast decision making without many computational effort.

Finally, query expansions mechanisms were studied. A new approach called ReNove was introduced that was explicitly designed to take the special properties of text streams into account. This approach was compared against several seminal relevance feedback and query expansion algorithms such Rocchio or Lavrenko's Relevance Model. It could be shown that ReNove performed significantly better than the other approaches and yielded more and better feedback terms that could be used for query expansion.

In summary, this chapter showed that seminal term weighting schemes still

[50]The TREC corpus only provided only approx. 50,000 relevance judgements where 9,000 could be used a training items. The Events corpus had in total 150,000 potential training items, but as discussed the quality is doubtful and the amount is still considerably small.

work well in high-volume text streams, that filter policies can be an efficient means to remove non-relevant documents, that simple threshold mechanisms can outperform machine learning approaches in scenarios where training data are scarce, and that a relevance feedback algorithm that accounts for the particularities of text streams can produce good quality query expansion terms.

Chapter 9

Summary and outlook

This dissertation has studied the case of information filtering and text stream analysis in high velocity text streams using event processing means. The focus for the dissertation has been on short documents coming from social media platforms such as Twitter or Facebook, but is supposed to be applicable to other short document sources such as source code commits, e-mail subject or chat logs. As the volume of user generated textual context is still to increase, it has been argued that efficient methods to process the ever-growing text volume are required. Event processing is an established methodology and is frequently applied in the context of temporal and flowing data. However, it has predominantly been applied to domains such as retail, logistics or finance for instance. Text stream data have not been at the focus of event processing, although the concepts that are commonly used in event processing like pattern matching, continuous computation or sliding windows can be beneficial in the context of filtering high-volume and highly dynamic text stream data. This dissertation showed that both domain areas can be combined in an efficient manner and evaluated how information filtering methods perform in a high-volume text stream scenario. The results were gathered by using a research approach based on Design Science that focuses on the practical and real-word results. This chapter provides a summary of the main contributions and sums up the findings made in this dissertation.

9.1 Summary of contributions

9.1.1 Introduction of event-driven information filtering and text stream processing

Summary The central impulse for this dissertation was the fact increasing volumes of text stream data are generated and that event-driven means, which are well suited for processing continuous data, have not been studied systematically in the context of text streams. Therefore, section 1.1 made the case for event-driven information filtering and text stream processing. Chapter 2 provided a concise overview of essential event processing concepts, and Chapter 3 provided a review of relevant concepts in information filtering and information retrieval. In both chapters the discussion was conducted with high-volume text streams in mind.

Contributions

- Discussions of the rationale for the combination of event processing and text stream data.

9.1.2 An event reference model for event-driven information filtering

Summary In order to be able to process text streams with event-driven means, an event reference model was required. This serves as a common vocabulary for relevant domain concepts and illustrates their relationships. Thereby, text stream data are unlocked for the processing with event-driven means, facilitating the discussion about and the implementation of event-driven text stream analysis systems. The event reference model was built using standard design methods in the area of reference modelling. In order to derive a solid reference model, a structured approach of seven design steps was followed. Subsequently, the proposed reference architecture was evaluated by several IT experts using a questionnaire based on the latest in reference model evaluation research. The evaluation showed that the proposed event reference model was considered as a valuable contribution and should be kept alive for further research. Additionally, a basic set of reference tasks in the area of information filtering and text stream processing was introduced, which incorporate event-driven aspects into the execution of the proposed tasks. This conveyed an impression of how event processing requirements affect the processing of such tasks.

Contributions

1. Design of an event reference model that allows turning un- and semi-structure text streams into discrete event types, to be processed using event-driven means.
2. Design and proposal of information filtering and text stream processing reference tasks in an event-driven manner.

9.1.3 A reference architecture for event-driven information filtering

Summary In addition to the conceptual foundations in form of a reference model, more advice for the concrete implementation of event-driven text stream analysis and information filtering systems was required. Therefore, a reference architecture was proposed that describes the essential components of text stream processing along the *domain neutral* event processing reference architecture proposed by the EPTS.

Contributions

1. Comprehensive overview of designing reference architecture in event processing.
2. Design of a reference architecture for event-driven information filtering and text stream processing including a detailed description of a functional view and a process view.
3. Basic set of event-driven architectural patterns to facilitate the implementation of event-driven text stream processing systems.

9.1.4 Analysis of dynamic sub-corpora

Summary Next to the combination of event processing and text stream analysis, there was a focus on limited memory. This aspect was represented by using sliding windows that define a highly volatile and predominantly dynamic sub-corpus. In order to allow for efficient text stream processing, it was necessary to study the influence of varying sizes of sliding windows on the structural properties of dynamic sub-corpora. It was shown that small sliding windows contain sufficient textual information for providing reasonable filtering and text stream analysis capabilities.

Contributions

1. Extensive statistical and exploratory analysis of sliding windows of various sizes and their effects on the structural properties of dynamic sub-corpora.
2. Study of adaptive and automated ways of determining sliding window sizes without upfront analysis effort.
3. Exploratory data analysis of various linguistics aspects of two social media text stream corpora.

9.1.5 Evaluation of event-driven information filtering components

Summary Following the verification of the dynamic sub-corpus approach and demonstration of the application of the event-driven approach on text stream analysis in Chapter 7, Chapter 8 discussed four essential components of information filtering, *viz.* filter policies, term weighting schemes and scoring, thresholds and query expansions. The concept of filter policies was introduced in the context of information filtering. Filter policies were defined as the gate keepers to the more processing-intensive filtering stages of the event processing network. Their goal is to remove as many as non-relevant documents as possible while keeping the false negative rate low. Then three approaches to define filter policies were discussed. First, heuristic filter policies were introduced. These are based on the domain expertise of a user and are generated manually. The second was a new approach based on frequent pattern mining. This approach was based on the idea to classify stream features like document length, spelling errors, amount of parts of speech into discrete categories which were then mapped onto distinct numerical ids. These ids were then fed into a frequent pattern mining algorithm that generated rules that could be used to filter incoming documents. Finally, weak classifiers were studied in terms of their capability to act as a preliminary filter stage.

Contributions

1. Introduction and discussion of different approaches for filter policy generation including performance evaluation.
2. Discussion, evaluation and comparison of another eighteen term weighting schemes, including the proposal of several new such schemes.
3. Discussion, evaluation and comparison of score-based threshold algorithms and machine learning classifications approaches.

4. Discussion, evaluation and comparison of query expansion mechanisms including the proposal of a new query expansions algorithm.

The novelty of this contributions is due to the special setting, i.e. the application of dynamic sub-corpora and the evaluation along three types of temporal search profiles.

9.2 Outlook and future work

This dissertation has presented the foundations for event-driven information filtering. This included the proposal of an event reference model, a reference architecture and an extensive study of information filtering components. Nevertheless, these were only the foundations, and there is space left for future research. Therefore, topics that can be of interest in this context will be briefly discussed to outline future research agendas.

9.2.1 Investigation of further text sources

The corpora used in this dissertation were based on data from Twitter and Facebook. But as stated at the beginning, there are further sources of short document streams that are worth analysing. It would be interesting to investigate whether the event-driven information filtering approach could be used on

- code commits from Github, Bitbucket or an internal source code management platform from a large software company. These sources could be used to study what is currently developed, how commits look in general from a linguistic point of view and whether there are any complex event patterns.
- e-mail titles without the content, in order to get a high-level, real-time insight into a stream of mail. As the means proposed here are rather lightweight, it could be interesting to use this approach to study e.g. spam mails in real-time from a linguistic aspect for instance.
- text stream data from further social media platforms like Instagram, Pinterest or Quora, in order to analyse the generated text streams in a way as proposed in this dissertation.

The proposed event-driven system could be extended to form a real-time text analysis platform. At its best it could provide a plug-in mechanism to

dynamically add new text source and provide a rich user interface that allows to monitor the text streams in real-time along a drill-down functionality to perform linguistic analysis[1].

9.2.2 Evaluation and revision of reference model and reference architecture

The event reference architecture was evaluated by several IT experts. This provided good insight into the strength and weaknesses of the reference model. But it would be beneficial to continue to work on this reference model, in order to improve it and to propagate its use in event-driven information filtering and text stream analysis.

9.2.3 Further reference implementations

As discussed in Chapter 6, there are many new and exciting stream processing frameworks like Apache Spark Streaming, Apache Storm or Apache Flink. It would be interesting to re-implement *StreamFI* with several new stream processing engines, in order to assess the applicability of the reference model and the reference architecture and mature both artefacts using the findings from these implementations. An assessment in terms of performance and other non-functional requirements could also be beneficial for an commercial adaptation of the reference model and reference architecture.

9.2.4 Extension of architecture pattern library

The patterns that were introduced in section 5.2.2.3 only represent a basic set of patterns that were derived in the course of this dissertation. These patterns also represent only a few event-driven architectural patterns. Therefore it would be beneficial to start a systematic pattern library for event-driven systems.

9.2.5 Automated components configuration

A filtering system consists of many moving parts, primarily related to the interplay between threshold, scoring and relevance feedback. Each of the components has one or more parameters that can be tuned. Thus, it would

[1] A few proposal are made in the last section of this chapter.

be interesting to establish a system that automatically finds the optimal parameters and adjusts the system on-the-fly.

9.2.6 Automated search profile categorization

The evaluation chapter showed that the filtering performance of the studied components differed significantly between the three used search profile categories. Therefore, it would be beneficial to discern the category of a search profile automatically, in order to provide the approach best suited for each search profile category.

9.2.7 Visualization

Information is nothing if it cannot be digested by users. Therefore, new and state-of-the-art visualization techniques are required to allow users to interact with the data that are generated by text streams in general and the system outlined in this dissertation in particular. A few aspects that could be addressed are outlined here.

9.2.7.1 Design of interactive filtering GUI

Like in information retrieval the quality of the results in information filtering depends strongly on the user profile and the used search terms. As users are usually not able to formulate their information need in such a way that the first set of search terms satisfactorily matches their information need, amendments to the initial query formulation are required. Here, an interactive form of relevance feedback and query expansion could be beneficial to improve the filtering results. For instance, the proposed ReNove algorithms produces candidate terms for query expansion and the user can interactively approve or decline the proposed terms. Of course, this has been studied in information retrieval research before, but the interesting aspects come from the dynamic nature of text streams. Here questions regarding usability, interface design and also cognitive aspects come here into play and offer space for research.

9.2.7.2 Design of interactive corpus analysis GUI

Chapter 7 discussed a static analysis of various text stream features. Several text stream properties like document length, parts-of-speech, etc. were addressed in terms of their time series progress and their distributional param-

eters. This analysis was post-hoc and static, i.e. the raw data were produced by *StreamFI*, collated, and subsequently analysed using *R*. This whole process could also be made interactive and in real-time. This user interface could contain functionalities to dynamically integrate new text streams without any manual configuration. Then, it should be possible to interactively define text sub-streams of interest and visualize the text stream; e.g. by using automatically and in real-time updateable visualization components that display time series progress or distributions. A drill-down functionality should also be contained, which allows selecting interesting time ranges directly from the time-series graphs. There are already a multitude of real-time dashboards and stream visualization applications, like *s-Visualizer*[2] or *Tableau*[3], but it would be interesting to investigate a specialized user interface exclusively focussed on the information filtering and information retrieval. It could offer functionalities for interactive stemming, word sense disambiguation or parts of speech annotations, in general tasks from static corpus analysis could be moved to an interactive level. A good starting point for patterns in this context is provided in (Burghardt, 2014).

9.2.7.3 Interactive design of digital humanities related event patterns

As outlined in the paragraph about digital humanities in section 4.2.4.4, a further topic for research could be the definition of patterns concerned with questions arising from digital humanities. A first step could be to verify whether the proposed patterns in table 4.1 work or not, and if similar event patterns could be generated. In a second step, an interactive user interface could be designed that allows defining simple sub-patterns or points of interest within a text stream that could be highlighted by the user interface. The user could gather building blocks for complex event patterns in the process. The integration and visualization of partial event pattern matches would also be beneficial in this context.

All the results and findings presented in this dissertation could be the foundation and starting point to address these outlined future research topics.

[2] `http://www.sqlstream.com/blaze/s-visualizer/`
[3] `ttps://www.tableau.com`

Bibliography

Abbate, J. (2000). *Inventing the internet*. Cambridge, MA: MIT Press.

Abel, F., Hauff, C., Houben, G.-J., Stronkman, R., & Tao, K. (2012). Semantics + filtering + search = twitcident. exploring information in social web streams. In *Proceedings of the 23rd acm conference on hypertext and social media* (pp. 285–294). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2309996.2310043

Adi, A., Botzer, D., Nechushtai, G., & Sharon, G. (2006). Complex event processing for financial services. In *Services computing workshops, 2006. scw'06. ieee* (pp. 7–12). Chicago, IL.

Adi, A., & Etzion, O. (2004). Amit - the situation manager. In (Vol. 13, pp. 177–203). Secaucus, NJ: Springer-Verlag. Retrieved from http://dx.doi.org/10.1007/s00778-003-0108-y doi: 10.1007/s00778-003-0108-y

Aggarwal, C. C., & Han, J. (2014). *Frequent pattern mining*. Springer Publishing Company, Incorporated.

Aggarwal, C. C., & Zhai, C. X. (2012). *Mining text data*. New York, NY: Springer.

Ahlemann, F., & Gastl, H. (2007). Process model for an empirically grounded reference model construction. In *Reference modeling for business systems analysis.* Hershey, PA: Idea Group Inc.

Albakour, M.-D., Macdonald, C., & Ounis, I. (2013). On sparsity and drift for effective real-time filtering in microblogs. In *Proceedings of the 22nd acm international conference on conference on information and knowledge management* (pp. 419–428). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2505515.2505709 doi: 10.1145/2505515.2505709

Alegria, A. (2011). *Complex Event Processing with Esper.* Retrieved from http://de.slideshare.net/antonio_alegria/complex-event-processing-with-esper-10122384 (Last accessed on 30.10.2015)

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language - Towns, Buildings, Construction*. New York, NY: Oxford University Press.

Allan, J. (1996). Incremental relevance feedback for information filtering. In *Proceedings of the 19th annual international acm sigir conference on research and development in information retrieval* (pp. 270–278). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/243199.243274 doi: 10.1145/243199.243274

Allan, J. (Ed.). (2002). *Topic detection and tracking: Event-based information organization*. Norwell, MA: Kluwer Academic Publishers.

Allan, J., Carbonell, J., Doddington, G., Yamron, J., & Yang, Y. (1998). Topic detection and tracking pilot study: Final report. *Proceedings of the DARPA broadcast news transcription and understanding workshop, 1998*, 194-218.

Allan, J., Papka, R., & Lavrenko, V. (1998). On-line new event detection and tracking. In *Proceedings of the 21st annual international acm sigir conference on research and development in information retrieval* (pp. 37–45). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/290941.290954 doi: 10.1145/290941.290954

Allen, J. F. (1983, November). Maintaining knowledge about temporal intervals. *Communication of the ACM*, *26*(11), 832–843. Retrieved from http://doi.acm.org/10.1145/182.358434 doi: 10.1145/182.358434

Allwood, J. (1981). On the Distinctions between Semantics and Pragmatics. In *Crossing the boundaries in linguistics* (pp. 177–189). Dordrecht, NL: Springer Netherlands.

Amati, G., & Van Rijsbergen, C. J. (2002, October). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, *20*(4), 357–389. Retrieved from http://doi.acm.org/10.1145/582415.582416 doi: 10.1145/582415.582416

Angelov, S., Grefen, P., & Greefhorst, D. (2009, Sept). A classification of software reference architectures: Analyzing their success and effectiveness. In *Joint working ieee/ifip conference on software architecture, 2009 european conference on software architecture. wicsa/ecsa 2009* (p. 141-150). Cambridge, UK. doi: 10.1109/WICSA.2009.5290800

Angelov, S., Trienekens, J., & Kusters, R. (2013). Software Reference Architectures - Exploring Their Usage and Design in Practice. In *Software architecture* (pp. 17–24). Berlin, Germany: Springer.

Angelov, S., Trienekens, J. J., & Grefen, P. (2008). Towards a method for the evaluation of reference architectures: Experiences from a case. In *Software architecture* (pp. 225–240). Heidelberg, Germany: Springer.

Apache Spark. (2015a, 04). *Apache Spark - Spark Programming Guide.* Retrieved from `https://spark.apache.org/docs/latest/programming-guide.html#shared-variables` (Last accessed on 16.05.2015)

Apache Spark. (2015b, 04). *Apache Spark Streaming - A Quick Example.* Retrieved from `https://spark.apache.org/docs/latest/streaming-programming-guide.html#a-quick-example` (Last accessed on 16.05.2015)

Arampatzis, A. (2001). *Adaptive and Temporally-dependent Document Filtering* (Unpublished doctoral dissertation). University of Nijmegen.

Arampatzis, A., Kamps, J., & Robertson, S. (2009). Where to stop reading a ranked list?: Threshold optimization using truncated score distributions. In *Proceedings of the 32nd international acm sigir conference on research and development in information retrieval* (pp. 524–531). New York, NY: ACM. Retrieved from `http://doi.acm.org/10.1145/1571941.1572031`

Arampatzis, A., & Robertson, S. (2011). Modeling score distributions in information retrieval. *Information Retrieval*, *14*(1), 26-46. Retrieved from `http://dx.doi.org/10.1007/s10791-010-9145-5` doi: 10.1007/s10791-010-9145-5

Arampatzis, A., & van Hameran, A. (2001). The score-distributional threshold optimization for adaptive binary classification tasks. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 285–293). New York, NY: ACM. Retrieved from `http://doi.acm.org/10.1145/383952.384009` doi: 10.1145/383952.384009

Barbieri, D. F., Braga, D., Ceri, S., Valle, E. D., & Grossniklaus, M. (2010). Continuous queries and real-time analysis of social semantic data with c-sparql. In *In proceedings of social data on the web workshop at the 8th international semantic web conference.* New York, NY: Springer.

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (2nd ed.). Boston, MA: Addison-Wesley.

Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice, 3rd edition*. Upper Saddle River, NJ: Addison-Wesley.

Bauer, A., & Wolff, C. (2012). Using Stream Features for Instant Document Filtering. In *Text retrieval conference (trec) 2012 proceedings.* Gaithersburg, MD. Retrieved from http://trec.nist.gov/pubs/trec21/papers/unir_de.microblog.final2.pdf

Bauer, A., & Wolff, C. (2014). An event processing approach to text stream analysis: Basic principles of event based information filtering. In *Proceedings of the 8th acm international conference on distributed event-based systems* (pp. 35–46). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2611286.2611288 doi: 10.1145/2611286.2611288

Baumgarten, C. (1999). A probabilistic solution to the selection and fusion problem in distributed information retrieval. In *Proceedings of the 22nd annual international acm sigir conference on research and development in information retrieval* (pp. 246–253). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/312624.312685 doi: 10.1145/312624.312685

Becker, J. (1998). *Die Grundsätze ordnungsmäßiger Modellierung und ihre Einbettung in ein Vorgehensmodell zur Erstellung betrieblicher Informationsmodelle.* Retrieved from http://www.wi-inf.uni-duisburg-essen.de/MobisPortal/pages/rundbrief/pdf/Beck98.pdf (Last accessed on 30.06.2015)

Becker, J., Matzner, M., Müller, O., & Walter, M. (2011, August). A Review of Event Formats as Enablers of Event-Driven BPM. In *Business process management workshops* (pp. 433–445). Berlin, Germany: Springer.

Beevolve. (2012, 10). *An Exhaustive Study of Twitter Users Across the World - Social Media Analytics | Beevolve.* Retrieved from http://www.beevolve.com/twitter-statistics/#a3 (Last accessed on 19.05.2015)

Belkin, N. J., & Croft, W. B. (1992). Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, *35*(12), 29–38. Retrieved from http://doi.acm.org/10.1145/138859.138861 doi: 10.1145/138859.138861

Bennett, S. (2014, September). *The History of Hashtags in Social Media Marketing [INFOGRAPHIC].* Retrieved from http://www.adweek.com/socialtimes/history-hashtag-social-marketing/501237 (Last accessed on 27. 09. 2015)

Berndt, D. J., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Workshop on knowledge discovery in databases* (pp. 359–370). Palo Alto, CA: AAAI Press.

Berry, R., Niblett, P., Jacobsen, A., Vincent, P., Seeger, B., & Eugster, P. (2011, March). The event processing manifesto. In M. Chandy, O. Etzion, & R. von Ammon (Eds.), *Dagstuhl seminar on event processing* (pp. 1–60). Dagstuhl, Germany. Retrieved from http://drops.dagstuhl.de/opus/volltexte/2011/2985/pdf/10201.SWM.2985.pdf

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the seventh SIAM international conference on data mining* (pp. 443–448). Retrieved from http://dx.doi.org/10.1137/1.9781611972771.42 doi: 10.1137/1.9781611972771.42

Bifet, A., Holmes, G., Pfahringer, B., & Gavaldà, R. (2011). Detecting Sentiment Change in Twitter Streaming Data. *Journal of Machine Learning Research: Workshop and Conference Proceedings Series*, *17*, 5–11.

Bitkom. (2012). *Big Data im Praxiseinsatz – Szenarien, Beispiele, Effekte* (M. Weber, Ed.). Berlin, Germany: BITKOM - Bundesverband Informationswirtschaft.

Blaszka, M., Burch, L. M., Frederick, E. L., Clavio, G., Walsh, P., Sanderson, J., et al. (2012). # worldseries: An empirical examination of a twitter hashtag during a major sporting event. *International Journal of Sport Communication*, *5*(4), 435–453. Retrieved from https://www.academia.edu/3256651/_WorldSeries_An_Empirical_Examination_of_a_Twitter_Hashtag_During_a_Major_Sporting_Event

Bocchi, L., & Ciancarini, P. (2003). A perspective on multiagent coordination models. In M.-P. Huget (Ed.), *Communication in multiagent systems* (Vol. 2650, p. 146-163). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-44972-0_7 doi: 10.1007/978-3-540-44972-0_7

Borko, H. (1968). Information science: What is it? *American Documentation*, *19*(1), 3–5. Retrieved from http://dx.doi.org/10.1002/asi.5090190103 doi: 10.1002/asi.5090190103

Boyd, D., & Crawford, K. (2012, June). Critical questions for Big Data. *Information, Communication & Society*, *15*(5), 662–679.

Brahaj, A. (2009). *List of English Stop Words.* Retrieved from http://norm.al/2009/04/14/list-of-english-stop-words/

Bruns, R., & Dunkel, J. (2010). *Event-Driven Architecture – Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse*. Berlin, Germany: Springer-Verlag.

Bryer, J., & Speerschneider, K. (2015). likert: Functions to analyze and visualize likert type items [Computer software manual]. Retrieved from http://CRAN.R-project.org/package=likert (R package version 1.3.1)

Buchgeher, G., & Weinreich, R. (2013). Towards Continuous Reference Architecture Conformance Analysis. In *Software architecture* (pp. 332–335). Berlin, Germany: Springer.

Buchmann, A., Pfohl, H.-C., Appel, S., Freudenreich, T., Frischbier, S., Petrov, I., & Zuber, C. (2011). Event-driven services: Integrating production, logistics and transportation. In *Service-oriented computing* (pp. 237–241). Berlin, Germany: Springer.

Burghardt, M. (2014). *Engineering Annotation Usability* (Unpublished doctoral dissertation). University of Regensburg.

Burton, K., Niels, K., & Soboroff, I. (2011, July). The icwsm 2011 spinn3r dataset. Palo Alto, CA: AAAI Press.

Busch, M., Gade, K., Larson, B., Lok, P., Luckenbill, S., & Lin, J. (2012). Earlybird: Real-Time Search at Twitter. In *2012 ieee 28th international conference on data engineering* (pp. 1360–1369). Arlington, VA: IEEE.

BusinessDictionary.com. (2012, 6). *Event.* Retrieved from http://www.businessdictionary.com/definition/event.html (Last accessed on 24.06.2012)

Büttcher, S., Clarke, C. L., & Cormack, G. V. (2010). *Information retrieval: Implementing and evaluating search engines*. The MIT Press.

Byron, L., & Wattenberg, M. (2008, November). Stacked graphs – geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, *14*(6), 1245–1252. Retrieved from http://dx.doi.org/10.1109/TVCG.2008.166 doi: 10.1109/TVCG.2008.166

Callan, J. (1998). Learning while filtering documents. In *Proceedings of the 21st annual international acm sigir conference on research and development in information retrieval - sigir '98* (pp. 224–231). New York, NY: ACM Press.

Carpineto, C., & Romano, G. (2012, January). A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, *44*(1), 1:1–1:50. Retrieved from http://doi.acm.org/10.1145/2071389.2071390 doi: 10.1145/2071389.2071390

Cataldi, M., Di Caro, L., & Schifanella, C. (2010). Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the tenth international workshop on multimedia data mining* (pp. 4:1–4:10). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1814245.1814249 doi: 10.1145/1814245.1814249

Cattell, R. (2011, May). Scalable sql and nosql data stores. *SIGMOD Rec.*, *39*(4), 12–27. Retrieved from http://doi.acm.org/10.1145/1978915.1978919 doi: 10.1145/1978915.1978919

Chandy, K. M., & Schulte, W. R. (2009). *Event Processing: Designing IT Systems for Agile Companies – Minibook*. New York, NY: McGraw-Hill Osborne Media.

Chen, C., Chen, Y., & Chen, M. (2007). An aging theory for event life-cycle modeling. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *37*(2), 237–248.

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, *1*(1), 9–36. Retrieved from http://doi.acm.org/10.1145/320434.320440 doi: 10.1145/320434.320440

Chomsky, N. (2002). *Syntactic structures* (2nd ed.). Berlin, Germany: Mouton de Gruyter.

Cleverdon, C., Mills, J., & Keen, M. (1966). *Aslib Cranfield research project - Factors determining the performance of indexing systems; Volume 1, Design; Part 1, Text* (Tech. Rep.). Cranfield University.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Hillsdayle, NJ: Lawrence Erlbaum Associates.

Commission, I. E. (2013). *Bus – iec 60050 – details for iev number 351-32-10.* Retrieved from http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=351-56-10 (Last accessed on 12.06.2015)

Cooper, W. S. (1976). *The Suboptimality of Retrieval Rankings Based on Probability of Usefulness.* Berkeley, CA: School of Library and Information Studies.

Cormode, G., & Muthukrishnan, S. (2005, April). An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, *55*(1), 58–75. Retrieved from http://dx.doi.org/10.1016/j.jalgor.2003.12.001 doi: 10.1016/j.jalgor.2003.12.001

Cormode, G., Shkapenyuk, V., Srivastava, D., & Xu, B. (2009). Forward decay: A practical time decay model for streaming systems. In *Proceedings of the 2009 ieee international conference on data engineering* (pp. 138–149). Washington, DC: IEEE Computer Society. Retrieved from http://dx.doi.org/10.1109/ICDE.2009.65 doi: 10.1109/ICDE.2009.65

Cristea, V., Pop, F., Dobre, C., & Costan, A. (2011). Distributed Architectures for Event-Based Systems. In *Reasoning in event-based distributed systems* (pp. 11–45). Berlin, Germany: Springer Berlin Heidelberg.

Crochemore, M., & Lecroq, T. (1997). Pattern matching and text compression algorithms. In *The computer science and engineering handbook* (2nd ed.). Boca Raton, FL: Chapman and Hall/CRC.

Croft, W. B., Metzler, D., & Strohman, T. (2010). *Search Engines - Information Retrieval in Practice*. Boston: Pearson.

Crystal, D. (2008). *A Dictionary of Linguistics and Phonetics* (6th ed.). Malden, MA: Blackwell Publishing.

Dayal, U., Blaustein, B., Buchmann, A., Chakravarthy, U., Hsu, M., Ledin, R., ... Jauhari, R. (1988, March). The hipac project: Combining active databases and timing constraints. *SIGMOD Rec.*, *17*(1), 51–70. Retrieved from http://doi.acm.org/10.1145/44203.44208 doi: 10.1145/44203.44208

Dean, B. (2015). *Google's 200 Ranking Factors: The Complete List.* Retrieved from http://backlinko.com/google-ranking-factors (Last accessed on 12.12.2015)

Dean, J., & Ghemawat, S. (2008, January). Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, *51*(1), 107–113. Retrieved from http://doi.acm.org/10.1145/1327452.1327492 doi: 10.1145/1327452.1327492

Dechter, R., Geffner, H., & Halpern, J. Y. (2010). Heuristics, Probability and Causality. A Tribute to Judea Pearl. *Heuristics, Probability and Causality. A Tribute to Judea Pearl*.

DeMarco, T. (1979). *Structured analysis and system specification*. Upper Saddle River, NJ: Yourdon Press.

Denning, P. J. (1982, March). Acm president's letter: Electronic junk. *Communications of the ACM*, *25*(3), 163–165. Retrieved from http://doi.acm.org/10.1145/358453.358454 doi: 10.1145/358453.358454

Dittrich, K., Gatziu, S., & Geppert, A. (1995). The active database management system manifesto: A rulebase of ADBMS features. *Rules in Database Systems*, 1–17.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87.

Domo. (2014). *Data Never Sleeps 2.0.* Retrieved from `https://www.domo.com/learn/data-never-sleeps-3-0` (Last accessed on 27.11.2015)

Dong, A., Zhang, R., Kolari, P., Bai, J., Diaz, F., Chang, Y., ... Zha, H. (2010). Time is of the essence: Improving recency ranking using twitter data. , 331–340.

Dong, L., Wang, D., & Sheng, H. (2006). Design of rfid middleware based on complex event processing. In *2006 ieee conference on cybernetics and intelligent systems* (pp. 1–6). Bangkok, Thailand.

Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. New York, N.Y.: Wiley-Interscience.

Dunkel, J., Ossowski, S., Ortiz, R., & Fernandez, A. (2009, 05/2009). Injecting semantics into event-driven architectures. In *Iceis 2009* (Vol. DISI, p. 70-75). Milan, Italy.

Efron, M., & Golovchinsky, G. (2011). Estimation methods for ranking recent information. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, 495. Retrieved from `http://portal.acm.org/citation.cfm?doid=2009916.2009984` doi: 10.1145/2009916.2009984

Efron, M., Lin, J., & Soboroff, I. (2013, 5). *Trec 2013 track guidelines.* `https://github.com/lintool/twitter-tools/wiki/TREC-2013-Track-Guidelines`. (Last accessed on 06.06.2015)

Ehresmann, A., von Ammon, R., & Iakovidis, D. K. (2012, June). *Ubiquitous Complex Event Processing in Exocortex Applications and Mathematical Approaches.* `http://www.complexevents.com/wp-content/uploads/2012/06/uCepCortex-Camad2012.pdf`. Retrieved from `http://www.complexevents.com/wp-content/uploads/2012/06/uCepCortex-Camad2012.pdf`

Elsawy, E., Mokhtar, M., & Magdy, W. (2014, November). TweetMogaz v2: Identifying News Stories in Social Media. In *Cikm '14: Proceedings of the 23rd acm international conference on conference on information and knowledge management*. ACM.

EsperTech Inc. (2015). *Performance-Related Information.* Retrieved from http://www.espertech.com/esper/performance.php (Last accessed on 12.06.2015)

Esswein, W., Zumpe, D.-K. S., & Sunke, N. (2004). Identifying the quality of e-commerce reference models. , 288–295. Retrieved from http://doi.acm.org/10.1145/1052220.1052257 doi: 10.1145/1052220.1052257

Etzion, O., & Niblett, P. (2011). *Event processing in action*. Stamford, CT: Manning Publications.

Evans, E. (2004). *Domain-driven Design*. Upper Saddle River, NJ: Addison-Wesley Professional.

Fagin, R. (2002, June). Combining fuzzy information: An overview. *SIGMOD Rec.*, *31*(2), 109–118. Retrieved from http://doi.acm.org/10.1145/565117.565143 doi: 10.1145/565117.565143

Fettke, P., & Loos, P. (2002). Methoden zur Wiederverwendung von Referenzmodellen–Übersicht und Taxonomie. Westfälische Wilhelms-Universität Münster. Institut f. Wirtschaftsinformatik.

Fettke, P., & Loos, P. (2003). Multiperspective Evaluation of Reference Models – Towards a Framework. In *Conceptual modeling for novel application domains* (pp. 80–91). Berlin, Germany: Springer.

Fettke, P., & Loos, P. (2004). Referenzmodellierungsforschung. *Wirtschaftsinformatik*, *46*(5), 331–340.

Fiscus, J. G., & Doddington, G. R. (2002). Topic detection and tracking evaluation overview.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, *76*(5), 378–382.

Frank, U. (2007). *Evaluation of Reference Models.* Hershey, PA: Idea Group Inc.

Galster, M., & Avgeriou, P. (2011). Empirically-grounded reference architectures: A proposal. , 153–158. Retrieved from http://doi.acm.org/10.1145/2000259.2000285 doi: 10.1145/2000259.2000285

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley.

Garlan, D., Bachmann, F., Ivers, J., Stafford, J., Bass, L., Clements, P., & Merson, P. (2010). *Documenting Software Architectures: Views and Beyond* (2nd ed.). Boston, MA: Pearson Education, Inc.

Gartner, Inc. (2014, August). *Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business.* Retrieved from https://www.gartner.com/newsroom/id/2819918 (Last accessed on 12.06.2015)

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., ... Smith, N. A. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. , 42–47. Retrieved from http://dl.acm.org/citation.cfm?id=2002736.2002747

Giorgino, T. (2009). Computing and visualizing dynamic time warping alignments in R: the dtw package. *Journal of Statistical Software*.

Godwin, M. (n.d.). *Meme, Counter-meme.* Retrieved from http://archive.wired.com/wired/archive/2.10/godwin.if.html (Last accessed on 29.11.2015)

Governor, J., Hinchcliffe, D., & Nickull, D. (2009). *Web 2.0 Architectures*. Beijing, China: O'Reilly Media.

Greefhorst, D., Koning, H., & Vliet, H. V. (2006, February). The many faces of architectural descriptions. *Information Systems Frontiers*, *8*(2), 103–113. Retrieved from http://dx.doi.org/10.1007/s10796-006-7975-x doi: 10.1007/s10796-006-7975-x

Grigorik, I. (2011, May). *Stream(SQL) Event Processing with Esper.* Retrieved from https://www.igvita.com/2011/05/27/streamsql-event-processing-with-esper/ (Last accessed on 27.11.2015)

Gross, B. M. (1964). *The managing of organizations: the administrative struggle* (Vol. 1). New York, NY: Free Press of Glencoe.

Gupta, M., Gao, J., Zhai, C., & Han, J. (2012). Predicting future popularity trend of events in microblogging platforms. In (Vol. 49, pp. 1–10). Hoboken, NJ: Wiley Subscription Services, Inc. Retrieved from http://dx.doi.org/10.1002/meet.14504901207 doi: 10.1002/meet.14504901207

Haghani, P., Michel, S., & Aberer, K. (2009). Evaluating top-k queries over incomplete data streams. In *Proceedings of the 18th acm conference on information and knowledge management* (pp. 877–886). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1645953.1646064 doi: 10.1145/1645953.1646064

Haghani, P., Michel, S., & Aberer, K. (2010). The gist of everything new: Personalized top-k processing over web 2.0 streams. In *Proceedings of the 19th acm international conference on information and knowledge management* (pp. 489–498). New York, NY: ACM.

Haghani, P., Michel, S., & Aberer, K. (2012, February). Efficient monitoring of personalized hot news over web 2.0 streams. *Comput. Sci.*, *27*(1), 81–92.

Hall, D., Chong, C.-Y., Llinas, J., & Liggins II, M. (2012). *Distributed Data Fusion for Network-Centric Operations*. Baco Raton, FL: CRC Press.

Hall, D. L., & Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, *85*(1), 6–23.

Hall, D. L., & Llinas, J. (2001). *Handbook on multisensor data fusion (Electrical engineering & applied signal processing)*. Boca Raton, FL: CRC Press.

Hanani, U., Shapira, B., & Shoval, P. (2001, August). Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, *11*(3), 203–259.

Harman, D. (1994). Overview of the Third Text REtrieval Conference (TREC-3). , 1–20. Retrieved from `http://trec.nist.gov/pubs/trec3/overview.ps` (Last accessed on 29.11.2015)

Hausenblas, M., & Bijnens, N. (2015). *Lambda Architecture - What is the Lambda Architecture.* Retrieved from `http://lambda-architecture.net/` (Last accessed on 22.06.2105)

He, D., & Parker, D. S. (2010). Topic dynamics: An alternative model of bursts in streams of topics. In *Proceedings of the 16th acm sigkdd international conference on knowledge discovery and data mining* (pp. 443–452). New York, NY: ACM.

Hevner, A., & Chatterjee, S. (2010). *Design Research in Information Systems*. New York, NY: Springer.

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105.

Hewitt, C. (2010). Actor Model of Computation: Scalable Robust Information Systems. *arXiv preprint arXiv:1008.1459*. Retrieved from `http://arxiv.org/pdf/1008.1459.pdf` (Last accessed on 29.11.2015)

Hewitt, C., Bishop, P., & Steiger, R. (1973). A Universal Modular ACTOR Formalism for Artificial Intelligence. In *Proceedings of the 3rd international joint conference on artificial intelligence* (pp. 235–245). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Hinze, A. (2003). *A-MEDIAS: Concept and design of an adaptive integrating event notification service* (Doctoral dissertation, Freie Universität Berlin). Retrieved from http://www.diss.fu-berlin.de/2003/248 (http://www.diss.fu-berlin.de/2003/248)

Hjørland, B. (2005). *Relevance - in: The epistemological lifeboat.* Retrieved from http://www.iva.dk/jni/lifeboat/info.asp?subjectid=114 (Last accessed on 29.11.2015)

Hofmeister, C., Kruchten, P., Nord, R. L., Obbink, H., Ran, A., & America, P. (2005). Generalizing a model of software architecture design from five industrial approaches. In *Proceedings of the 5th working ieee/ifip conference on software architecture* (pp. 77–88). Washington, DC: IEEE Computer Society.

Hofmeister, C., Kruchten, P., Nord, R. L., Obbink, H., Ran, A., & America, P. (2007). A general model of software architecture design derived from five industrial approaches. *J. Syst. Softw.*, *80*(1), 106–126. Retrieved from http://dx.doi.org/10.1016/j.jss.2006.05.024 doi: 10.1016/j.jss.2006.05.024

Hofmeister, C., Nord, R., & Soni, D. (2000). *Applied software architecture*. Reading, MA: Addison-Wesley Professional.

Hong, Y., Fei, Y., & Yang, J. (2013). Exploiting topic tracking in real-time tweet streams. In *Proceedings of the 2013 international workshop on mining unstructured big data using natural language processing* (pp. 31–38). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2513549.2513555 doi: 10.1145/2513549.2513555

Horn, C. (2010). Analysis and Classification of Twitter messages.

Huang, J., Thornton, K. M., & Efthimiadis, E. N. (2010). Conversational tagging in twitter. *HT*, 173–178.

Hull, D. A. (1998). The TREC-7 Filtering Track: Description and Analysis. *TREC*, 9–32. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=2AA6A3905D2AA740F925FC16D4350792?doi=10.1.1.51.7780&rep=rep1&type=pdf (Last accessed on 29.11.2015)

Hummer, W., Inzinger, C., Leitner, P., Satzger, B., & Dustdar, S. (2012). Deriving a unified fault taxonomy for event-based systems. In *Proceedings of the 6th acm international conference on distributed event-based systems* (pp. 167–178). New York, NY: ACM.

International Organization for Standardization. (1994). *Information Processing Systems, Open Systems Interconnection, Basic Reference Model.* Geneve, Switzerland.

Irani, D., Webb, S., & Pu, C. (2010). Study of trend-stuffing on twitter through text classification. In *Ceas 2010 - seventh annual collaboration, electronic messaging, antiabuse and spam conference.*

ISO. (2011). *IEEE: ISO/IEC/IEEE 42010: 2011: Systems and Software Engineering, Architecture Description.* Geneva, Switzerland: ISO - International Organization of Standardization.

Israel, G. D. (1992). *Determining sample size.* Miami, FL.

Jain, R. (2008). Eventweb: Developing a human-centered computing system. *Computer*, *41*(2), 42-50. doi: http://doi.ieeecomputersociety.org/10.1109/MC.2008.49

Janert, P. K. (2010). *Data Analysis with Open Source Tools*. Sebastopol, CA: O'Reilly Media Inc.

JBoss. (2015, 5). *Drools Fusion - Temporal Reasoning.* Retrieved 05.03.2015, from `http://docs.jboss.org/drools/release/6.2.0.CR1/drools-docs/html/DroolsComplexEventProcessingChapter.html#d0e10852` (Last accessed on 30.5.2015)

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11-21.

Jones, R., & Diaz, F. (2007, July). Temporal profiles of queries. *ACM Trans. Inf. Syst.*, *25*(3).

Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (1st ed.). Englewood Cliffs, CA: Prentice Hall.

Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for Architecture Evaluation* (Tech. Rep. No. CMU/SEI-2000-TR-004). Pittsburgh, PA: Software Engineering Institute.

Kelter, U. (n.d.). *Hinweise zum Ablauf von Promotionsvorhaben, der Struktur einer Dissertation und ihrer Vorabversionen.* Retrieved from `http://pi.informatik.uni-siegen.de/kelter/div/diss_entwuerfe` (Last accessed on 19.05.2015)

Kenney, J. F., & Keeping, E. S. (1954). *Mathematics of Statistics* (3rd ed., Vol. 1). Princeton, NJ: Van Nostrand Company.

Keogh, E. J., & Kasetty, S. (2003). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *Data Min. Knowl. Discov. ()*, *7*(4), 349–371.

Kleinberg, J. (2002). Bursty and hierarchical structure in streams. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining.* New York, NY: ACM.

Kleisarchaki, S. (2012). *A Methodological Framework for Statistical Analysis of Text from Social Media* (Unpublished master's thesis). University of Crete.

Klinkenberg, R., & Joachims, T. (2000). Detecting Concept Drift with Support Vector Machines. *ICML*, 487–494.

Klinkenberg, R., & Renz, I. (1998). Adaptive information filtering: Learning drifting concepts. *Proceedings of AAAI-98/ICML-98 – Workshop Learning for Text Categorization*. Retrieved from `http://www-ai.cs.uni-dortmund.de/PublicPublicationFiles/klinkenberg_renz_98b.pdf`

Kotov, A., Zhai, C., & Sproat, R. (2011). Mining named entities with temporally correlated bursts from multilingual web news streams. In *Proceedings of the fourth acm international conference on web search and data mining* (pp. 237–246). New York, NY: ACM.

Kotsiantis, S. B., & Kanellopoulos, D. (2006). Data preprocessing for supervised leaning. *... Journal of Computer ...*.

Kroeger, P. R. (2005). *Analyzing Grammar*. Cambridge, UK: Cambridge University Press.

Kruchten, P. (2003). *The Rational Unified Process* (3rd ed.). Reading, MA: Addison-Wesley.

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *Software, IEEE*, *12*(6), 42–50.

Kumar, H. (2015, April). *Apache Storm Performance Tuners.* Retrieved from `http://www.ericsson.com/research-blog/data-knowledge/apache-storm-performance-tuners/` (Last accessed on 12.06.2015)

Kumar, S., Morstatter, F., & Liu, H. (2013). Twitter Data Analytics.

Kunz, W., & Rittel, H. (2000). *Die informationswissenschaften : Ihre anstze, probleme, methoden und ihr ausbau in der bundesrepublik deutschland.* Munich, Germany: R. Oldenbourg. Retrieved from http://scidok.sulb.uni-saarland.de/volltexte/2000/31 (Last accessed on 29.11.2015)

Kuropka, D. (2004). *Modelle zur Repräsentation natürlichsprachlicher Dokumente: Ontologie-basiertes Information-Filtering-und-Retrieval mit relationalen Datenbanken.* Berlin, Germany: Logos Verlag.

Kürsten, J. (2012). *A Generic Approach to Component-Level Evaluation in Information Retrieval* (Unpublished doctoral dissertation). Chemnitz.

Lafferty, J., & Zhai, C. (2002). Probabilistic relevance models based on document and query generation. In *Language modeling and information retrieval* (pp. 1–10). Dordrecht, Netherlands: Kluwer Academic Publishers.

Lamport, L. (1978, July). Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, *21*(7), 558–565. Retrieved from http://doi.acm.org/10.1145/359545.359563 doi: 10.1145/359545.359563

Landis, J., & Koch, G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, *33*(1), 159.

Lanquillon, C. (2001). *Enhancing text classification to improve information filtering* (Doctoral dissertation, University of Marburg, Germany). Retrieved from http://edoc2.bibliothek.uni-halle.de/hs/download/pdf/285?originalFilename=true

Lanquillon, C., & Renz, I. (1999). Adaptive information filtering: Detecting changes in text streams. In *Proceedings of the eighth international conference on information and knowledge management* (pp. 538–544). New York, NY: ACM.

Lavrenko, V., & Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 120–127). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/383952.383972 doi: 10.1145/383952.383972

Le, A., Arkhipov, A., Hanley, M., & Nyholm, S. (2006, February). *Microsoft .NET Pet Shop 4: Migrating an ASP.NET 1.1 Application to 2.0.* Retrieved from https://msdn.microsoft.com/en-us/library/aa479070.aspx (Last accessed on 05.07.2015)

Lee, C.-H., Wu, C.-H., & Chien, T.-F. (2011). Burst: A dynamic term weighting scheme for mining microblogging messages. , 548–557. Retrieved from http://dl.acm.org/citation.cfm?id=2009463.2009531

Lenhart, A. (n.d.). *Teens, social media & technology overview 2015.* Retrieved 09.04.2015, from http://www.pewinternet.org/2015/04/09/teens-social-media-technology-2015/ (Last accessed on 12.12.2015)

Lewis, D. D. (1995a). Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international acm sigir conference on research and development in information retrieval* (pp. 246–254). New York, NY: ACM.

Lewis, D. D. (1995b). The TREC-4 filtering track. *Proceedings 4th Text REtrieval Conference*.

Lewis, D. D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). Training algorithms for linear text classifiers. In *Proceedings of the 19th annual international acm sigir conference on research and development in information retrieval* (pp. 298–306). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/243199.243277 doi: 10.1145/243199.243277

Li, X., & Croft, W. B. (2003). Time-based language models. *Proceedings of the twelfth international conference on Information and knowledge management - CIKM '03*, 469. Retrieved from http://portal.acm.org/citation.cfm?doid=956863.956951 doi: 10.1145/956950.956951

Liang, F., Qiang, R., & Yang, J. (2012). Exploiting real-time information retrieval in the microblogosphere. In *Proceedings of the 12th acm/ieee-cs joint conference on digital libraries* (pp. 267–276). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2232817.2232867 doi: 10.1145/2232817.2232867

Licklider, J. C. R., & Taylor, R. W. (1990). *In memoriam, J.C.R. Licklider, 1915-1990.* Palo Alto, CA: digital – System Research Center. Retrieved from http://memex.org/licklider.pdf (Last accessed on 27.11.2015)

Linguistic Data Consortium. (2004). TDT 2004: Annotation Manual. , 1–15. Retrieved from https://catalog.ldc.upenn.edu/docs/LDC2006T19/TDT2004V1.2.pdf (Last accessed on 13.11.2015)

Luckham, D. (n.d.). *Complex event processing.* Retrieved from http://complexevents.com/stanford/cep/ (Last accessed on 16.06.2012])

Luckham, D. (2002). *The power of events: An introduction to complex event processing in distributed enterprise systems* (1st ed.). Boston, MA: Addison-Wesley Professional.

Luckham, D. (2006, June). *What's the Difference Between ESP and CEP?* Retrieved from http://www.complexevents.com/2006/08/01/what%e2%80%99s-the-difference-between-esp-and-cep/ (Last accessed on 31.05.2015)

Luckham, D. (2008). *A Short History of Complex Event Processing.* Retrieved from http://complexevents.com/wp-content/uploads/2008/02/1-a-short-history-of-cep-part-1.pdf (Last accessed on 13.11.2015)

Luckham, D., & Schulte, R. W. (2013, 1). *Why Companies Should Develop Event Models.* Retrieved from http://www.complexevents.com/wp-content/uploads/2013/01/Event-Models-Luckham-schulte-Final-Jan-2013.pdf (Last accessed on 19.05.2015)

Luckham, D., & Schulte, W. R. (2011, 07). *Epts event processing glossary v2* (Tech. Rep.). Event Processing Technical Society. Retrieved from http://www.complexevents.com/wp-content/uploads/2011/08/EPTS_Event_Processing_Glossary_v2.pdf

Luckham, D. C., & Vera, J. (1995, September). An event-based architecture definition language. *IEEE Trans. Softw. Eng.*, *21*(9), 717–734. Retrieved from http://dx.doi.org/10.1109/32.464548 doi: 10.1109/32.464548

Luhn, H. P. (1958, October). A business intelligence system. *IBM J. Res. Dev.*, *2*(4), 314–319. Retrieved from http://dx.doi.org/10.1147/rd.24.0314 doi: 10.1147/rd.24.0314

Lynch, N. A. (1996). *Distributed algorithms*. Morgan Kaufmann Publishers Inc.

Ma, S., & Hellerstein, J. L. (2001). Mining partially periodic event patterns with unknown periods. In *Proceedings of the 17th international conference on data engineering* (pp. 205–214). Washington, DC: IEEE Computer Society. Retrieved from http://dl.acm.org/citation.cfm?id=645484.656369

Macdonald, C., Efron, M., Lin, J., & Soboroff, I. (2012, 5). *Trec 2012 track guidelines.* Retrieved from https://sites.google.com/site/microblogtrack/2012-guidelines (Last accessed on 06.06.2015)

Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., & Cohen, M. D. (1987, May). Intelligent information-sharing systems. *Commun. ACM*, *30*(5), 390–402. Retrieved from http://doi.acm.org/10.1145/22899.22903 doi: 10.1145/22899.22903

March, S. T., & Smith, G. F. (1995, December). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266. Retrieved from http://dx.doi.org/10.1016/0167-9236(94)00041-2 doi: 10.1016/0167-9236(94)00041-2

Maron, E. (1964). *Mechanized Documentation: The Logic Behind a Probabilistic Interpretation*. Santa Monica, CA: Rand Corporation.

Maron, M. E., & Kuhns, J. L. (1960, July). On relevance, probabilistic indexing and information retrieval. *J. ACM*, *7*(3), 216–244. Retrieved from http://doi.acm.org/10.1145/321033.321035 doi: 10.1145/321033.321035

Martínez-Fernández, S., Ayala, C. P., Franch, X., & Marques, H. M. (2013). Benefits and Drawbacks of Reference Architectures. In *Software architecture* (pp. 307–310). Berlin, Germany: Springer.

Martínez-Fernández, S., Ayala, C. P., Franch, X., Marques, H. M., & Ameller, D. (2013). A Framework for Software Reference Architecture Analysis and Review. In *Cibse 2013 - x workshop latinoamericano ingeniería de software experimental* (pp. 89–102). Montevideo, Uruguay.

Matloff, N. (2008). *Introduction to discrete-event simulation and the simpy language.* Retrieved from http://heather.cs.ucdavis.edu/~matloff/156/PLN/DESimIntro.pdf (Last accessed on 18.05.2012)

Matook, S., & Indulska, M. (2009, April). Improving the quality of process reference models: A quality function deployment-based approach. *Decision Support Systems*, *47*(1), 60–71. Retrieved from http://dx.doi.org/10.1016/j.dss.2008.12.006 doi: 10.1016/j.dss.2008.12.006

Matthews, G. (2004). Fairness Opinions: Common Errors and Omissions. In *The handbook of business valuation and intellectual property analysis.* New York, NY: McGraw-Hill.

Matuszak, A. (n.d.). *Differences between Arithmetic, Geometric, and Harmonic Means.* Retrieved from http://economistatlarge.com/finance/applied-finance/differences-arithmetic-geometric-harmonic-means

McCarthy, D., & Dayal, U. (1989a). The architecture of an active database management system. *ACM SIGMOD Record*, *18*(2), 215–224.

McCarthy, D., & Dayal, U. (1989b, June). The architecture of an active database management system. *SIGMOD Rec.*, *18*(2), 215–224. Retrieved from http://doi.acm.org/10.1145/66926.66946 doi: 10.1145/66926.66946

McCreadie, R., Soboroff, I., Lin, J., Macdonald, C., Ounis, I., & McCullough, D. (2012). On building a reusable twitter corpus. , 1113–1114. Retrieved from http://doi.acm.org/10.1145/2348283.2348495 doi: 10.1145/2348283.2348495

McGee, M. (2011). *Organic Click-Thru Rates Tumbling; Only 52% Click On Page One, Study Suggests.* Retrieved from http://searchengineland.com/organic-click-thru-rates-tumbling-study-97338 (Last accessed on 19.05.2015)

McMinn, A. J., Moshfeghi, Y., & Jose, J. M. (2013). Building a Large-scale Corpus for Evaluating Event Detection on Twitter. *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, 409-418.

Medvidovic, N., Oreizy, P., Robbins, J. E., & Taylor, R. N. (1996, October). Using object-oriented typing to support architectural design in the c2 style. *SIGSOFT Softw. Eng. Notes*, *21*(6), 24–32. Retrieved from http://doi.acm.org/10.1145/250707.239106 doi: 10.1145/250707.239106

Merriam Webster Online. (2012, 6). *Event.* Retrieved from http://www.merriam-webster.com/dictionary/event (Last accessed on 29.11.2015)

Metzler, D., & Bruce Croft, W. (2007, June). Linear feature-based models for information retrieval. *Inf. Retr.*, *10*(3), 257–274. Retrieved from http://dx.doi.org/10.1007/s10791-006-9019-z doi: 10.1007/s10791-006-9019-z

Metzler, D. A. (2007). *Beyond bags of words: effectively modeling dependence and features in information retrieval* (Unpublished doctoral dissertation). University of Massachusetts Amherst.

Michelson, B. M. (2006). *Event-driven architecture overview* (Tech. Rep.). Boston, MA: Patricia Seybold Group. Retrieved from http://www.omg.org/soa/Uploaded%20Docs/EDA/bda2-2-06cc.pdf (Last accessed on 13.11.2015)

Microsoft. (2012). *What is an Enterprise Service Bus.* Retrieved from https://msdn.microsoft.com/en-us/biztalk/dd876606 (Last accessed on 12.07.2015)

Miner, D., & Shook, A. (2012). *MapReduce Design Patterns – Building Effective Algorithms and Analytics for Hadoop and Other Systems*. Beijing, China: O'Reilly Media, Inc.

Miyanishi, T., Seki, K., & Uehara, K. (2013, October). Improving pseudo-relevance feedback via tweet selection. In *Cikm '13: Proceedings of the 22nd acm international conference on conference on information & knowledge management*. ACM Request Permissions.

Morris, C. (1946). *Signs, language and behavior*. New York, NY: Prentice Hall.

Moukas, A., & Maes, P. (1998, January). Amalthaea: An evolving multi-agent information filtering and discovery system for the www. *Autonomous Agents and Multi-Agent Systems*, *1*(1), 59–88. Retrieved from http://dx.doi.org/10.1023/A:1010094506174 doi: 10.1023/A:1010094506174

Mouratidis, K., Bakiras, S., & Papadias, D. (2006). Continuous monitoring of top-k queries over sliding windows. In *Proceedings of the 2006 acm sigmod international conference on management of data* (pp. 635–646). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1142473.1142544 doi: 10.1145/1142473.1142544

Mouratidis, K., & Pang, H. (2009). An incremental threshold method for continuous text search queries. *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, 1187–1190.

Mouratidis, K., & Pang, H. (2011). Efficient evaluation of continuous text search queries. *IEEE Transactions on Knowledge and Data Engineering*, *23*(10), 1469–1482.

Muller, G. J. (2004). *CAFCR: A Multi-view Method for Embedded Systems Architecting. Delft* (Unpublished doctoral dissertation). Delft University of Technology.

Müller, M. (2007). *Information Retrieval for Music and Motion*. Secaucus, NJ: Springer-Verlag New York, Inc.

Nance, R. E. (1993). A history of discrete event simulation programming languages. In *The second acm sigplan conference on history of programming languages* (pp. 149–175). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/154766.155368 doi: 10.1145/154766.155368

National Institute of Standards and Technology (NIST). (2013). *Engineering Statistics Handbook - 2.2.2.1. Shewhart control chart.* Retrieved from http://www.itl.nist.gov/div898/handbook/mpc/section2/mpc221.htm (Last accessed on 30.09.2015)

Naveed, N., Gottron, T., Kunegis, J., & Alhadi, A. C. (2011). Bad news travel fast: A content-based analysis of interestingness on twitter. In *Proceedings*

of the 3rd international web science conference (pp. 8:1–8:7). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2527031.2527052 doi: 10.1145/2527031.2527052

Nikolov, S. (2012). *Trend or No Trend: A Novel Nonparametric Method for Classifying Time Series*. Massachusetts Institute of Technology. Retrieved from http://dspace.mit.edu/handle/1721.1/85399

Noll, M. G. (2013). *Implementing Real-Time Trending Topics With a Distributed Rolling Count Algorithm in Storm.* Retrieved from http://www.michael-noll.com/blog/2013/01/18/implementing-real-time-trending-topics-in-storm/ (Last accessed on 29.11.2015)

Nwana, H. S. (1996). Software agents: An overview. *Knowledge Engineering Review, Vol. 11, No 3*, pp.1-40.

Oard, D. W. (1997, March). The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, *7*(3), 141–178. Retrieved from http://dx.doi.org/10.1023/A:1008287121180 doi: 10.1023/A:1008287121180

OASIS - Organization for the Advancement of Structured Information Standards. (2006). *Reference Model for Service Oriented Architecture.* Retrieved from https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf (Last accessed on 27.11.2015)

OASIS - Organization for the Advancement of Structured Information Standards. (2013). *Reference architecture foundation for service oriented architecture version 1.0* (1.0 ed.). OASIS Service Oriented Architecture Reference Model TC. Retrieved from http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html (Last accessed on 27.11.2015)

Office of the DoD CIO. (2010). *DoD Reference Architecture Description* (Tech. Rep.). Retrieved from http://dodcio.defense.gov/Portals/0/Documents/DIEA/Ref_Archi_Description_Final_v1_18Jun10.pdf

Ounis, I. (2005). *BM25.* Retrieved from http://ir.dcs.gla.ac.uk/wiki/BM25 (Last accessed on 15. 11 2015)

Ounis, I., Macdonald, C., & Soboroff, I. (2008). On the TREC Blog Track. In *Proceedings of the 2n international conference weblogs and social media, icwsm 2008.* NIST. Retrieved from http://www.dcs.gla.ac.uk/~craigm/publications/ounis08trecblog.pdf

Owen, S., Anil, R., Dunning, T., & Friedman, E. (2011). *Mahout in Action*. Shelter Island, NY: Manning Publications Company.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of emnlp* (pp. 79–86).

Paschke, A., Vincent, P., Alves, A., & Moxey, C. (2012). Tutorial on advanced design patterns in event processing. In *Proceedings of the 6th acm international conference on distributed event-based systems* (pp. 324–334). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2335484.2335519 doi: 10.1145/2335484.2335519

Paschke, A., Vincent, P., Moxey, C., Hirzel, M., & Alves, A. (2012, 07). *Event Processing Reference Architecture - Design Patterns.* Retrieved from http://de.slideshare.net/isvana/epts-debs2011-event-processing-reference-architecture-and-patterns-tutorial-v1-2

Paschke, A., Vincent, P., & Springer, F. (2011, 11). Standards for Complex Event Processing and Reaction Rules. In *Rule - based modeling and computing on the semantic web* (pp. 128–139). Berlin, Germany: Springer.

Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D. J., Johnson, N., Cohen, K. B., & Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In *Proceedings of the workshop on bionlp 2007 biological, translational, and clinical language processing - bionlp '07* (pp. 97–104). Morristown, NJ: Association for Computational Linguistics.

Preiss, B. R. (2000). *Data structures and algorithms with object-oriented design patterns in Java*. New York, NY: John Wiley & Sons, Inc. hhttp://www.brpreiss.com/books/opus5/. (Last accessed 18.05.2012)

Qin, L., Yu, J. X., & Chang, L. (2012, 07). Diversifying top-k results. *Proceedings of the VLDB Endowment*, *5*(11), 1124–1135. Retrieved from http://dx.doi.org/10.14778/2350229.2350233 doi: 10.14778/2350229.2350233

Rakthanmanon, T., Campana, B. J. L., Mueen, A., Batista, G. E. A. P. A., Westover, M. B., 0002, Q. Z., ... Keogh, E. J. (2012). Searching and mining trillions of time series subsequences under dynamic time warping. *KDD*, 262–270.

Ratanamahatana, C. A., & Keogh, E. (2004). Everything you know about dynamic time warping is wrong. *3rd Workshop on Mining Temporal and Sequential Data, in conjunction with 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD-2004)*.

Reed, J. W., Jiao, Y., Potok, T. E., Klump, B. A., Elmore, M. T., & Hurson, A. R. (2006). TF-ICF: A New Term Weighting Scheme for Clustering Dynamic Data Streams. In *Machine learning and applications, 2006. icmla '06. 5th international conference on* (pp. 258–263).

Renners, L. (2013). *Complex Event Processing auf der Basis unsicherer Daten* (Unpublished master's thesis). FH Hannover, Hannover.

Renners, L., Bruns, R., & Dunkel, J. (2012). Situation-aware energy control by combining simple sensors and complex event processing. *Workshop on AI Problems and Approaches for Intelligent Environments*, *12*, 33. Retrieved from http://ceur-ws.org/Vol-907/AI4IE_proceedings.pdf#page=37

Riemer, D., Stojanovic, L., & Stojanovic, N. (2012). Using complex event processing for modeling semantic requests in real-time social media monitoring. In *Sixth international aaai conference on weblogs and social media.* Dublin, Ireland. Retrieved from http://www.aaai.org/ocs/index.php/ICWSM/ICWSM12/paper/viewFile/4754/5083 (Last accessed on 13.11.2015)

Rijsbergen, C. J. V. (1979). *Information retrieval* (2nd ed.). Newton, MA: Butterworth-Heinemann.

Robertson, S. (2002, 04). Threshold setting and performance optimization in adaptive filtering. *Information Retrieval*, *5*(2-3), 239–256. Retrieved from http://dx.doi.org/10.1023/A:1015702129514 doi: 10.1023/A:1015702129514

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, *60*(5), 503–520.

Robertson, S., Kanoulas, E., & Yilmaz, E. (2013). Modelling score distributions without actual scores. In *Proceedings of the 2013 conference on the theory of information retrieval* (pp. 20:85–20:92). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2499178.2499181 doi: 10.1145/2499178.2499181

Robertson, S. E. (1997). Readings in information retrieval. In K. Sparck Jones & P. Willett (Eds.), (pp. 281–286). San Francisco, CA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=275537.275701

Robertson, S. E., & Hull, D. A. (2000). The TREC-9 Filtering Track Final Report. *TREC*. Retrieved from http://trec.nist.gov/pubs/trec9/papers/filtering_track.ps

Robertson, S. E., & Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, *27*(3), 129–146. Retrieved from http://dx.doi.org/10.1002/asi.4630270302 doi: 10.1002/asi.4630270302

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international acm sigir conference on research and development in information retrieval* (pp. 232–241). New York, NY: Springer-Verlag New York, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=188490.188561

Rolland, C. (1998). A comprehensive view of process engineering. In *Advanced information systems engineering* (pp. 1–24). Berlin, Germany: Springer.

Roth, M., & Donath, S. (2012). Applying complex event processing towards monitoring of multi-party contracts and services for logistics–a discussion. In *Business process management workshops 2011* (pp. 458–463). Berlin, Germany: Springer.

Roush, W. (2010, 10 28). *The Two-Second Advantage: Talking with TIBCO's Vivek Ranadivé.* Retrieved from http://www.xconomy.com/san-francisco/2010/10/28/the-two-second-advantage-talking-with-tibcos-vivek-ranadive/

Rozsnyai, S., Schiefer, J., & Schatten, A. (2007). Concepts and models for typing events for event-based systems. In *Proceedings of the 2007 inaugural international conference on distributed event-based systems* (pp. 62–70). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1266894.1266904 doi: 10.1145/1266894.1266904

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

Sahami, M., & Heilman, T. D. (2006). A web-based kernel function for measuring the similarity of short text snippets. *Proceedings of the 15th international conference on World Wide Web - WWW '06*.

Salton, G. (1968). *Automatic information organization and retrieval*. New York, NY: McGraw Hill.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*(5), 513–523. Re-

trieved from http://dx.doi.org/10.1016/0306-4573(88)90021-0 doi: 10.1016/0306-4573(88)90021-0

Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., & Sperling, J. (2009). Twitterstand: news in tweets. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems* (pp. 42–51).

Saracevic, T. (1975, 11). RELEVANCE: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information science*, *26*(6), 321–343.

Schaich, E., Köhle, D., Schweitzer, W., & Wegner, F. (1993). *Statistik für volkswirte, betriebswirte und soziologen*. München: Verlag Vahlen.

Schilling, B., Koldehofe, B., Rotherme, K., & Ramachandran, U. (2013). *Access policy consolidation for event processing systems*. Washington, DC: IEEE Computer Society. Retrieved from http://dx.doi.org/10.1109/NetSys.2013.18 doi: 10.1109/NetSys.2013.18

Schneider, U., & Werner, D. (2001). *Taschenbuch der Informatik*. Munich, Germany: Fachbuchverl. Leipzig im Carl-Hanser-Verlag.

Schütte, R. (1998). *Grundsätze ordnungsgemäßer Referenzmodellierung – Konstruktion konfigurations– und anpassungsorientierter Modelle*. Wiesbaden, Germany: Springer Fachmedien.

Schütze, H., Raghavan, P., & Manning, C. (2009). *An introduction to information retrieval*. Cambridge, UK: Cambridge University Press.

Sebastiani, F. (2002, March). Machine learning in automated text categorization. *ACM Comput. Surv.*, *34*(1), 1–47. Retrieved from http://doi.acm.org/10.1145/505282.505283 doi: 10.1145/505282.505283

Sen, S., Stojanovic, N., & Shemrani, B. F. (2010). Echopat: A system for real-time complex event pattern monitoring. In *Proceedings of the fourth acm international conference on distributed event-based systems* (pp. 107–108). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1827418.1827442 doi: 10.1145/1827418.1827442

Shannon, C. E. (2001, January). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, *5*(1), 3–55. Retrieved from http://doi.acm.org/10.1145/584091.584093 doi: 10.1145/584091.584093

Shen, X., Tan, B., & Zhai, C. (2005). Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international acm sigir conference on research and development in information retrieval* (pp. 43–50). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1076034.1076045 doi: 10.1145/1076034.1076045

Shou, L., Wang, Z., Chen, K., & Chen, G. (2013). Sumblr: Continuous summarization of evolving tweet streams. In *Proceedings of the 36th international acm sigir conference on research and development in information retrieval* (pp. 533–542). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2484028.2484045 doi: 10.1145/2484028.2484045

Shraer, A., Gurevich, M., Fontoura, M., & Josifovski, V. (2013). Top-k publish-subscribe for social annotation of news. *Proceedings of the VLDB Endowment*, *6*(6), 385–396.

Soboroff, I., Ounis, I., & Lin, J. (2012). Overview of the TREC-2012 Microblog Track. In *The twenty-first text retrieval conference (trec 2012) proceedings.* Gaithersburg, MD. Retrieved from http://trec.nist.gov/pubs/trec21/papers/MICROBLOG12OVERVIEW.pdf

Spark Jones, K. (1971). *Automatic keyword classification for information retrieval*. London, UK: Butterworths.

Springer, F., Etzion, O., Paulus, T., Von Ammon, R., Emmersberger, C., & Ertlmaier, T. (2009). Existing and future standards for event-driven business process management. In *Proceedings of the third acm international conference on distributed event-based systems.* New York, NY: ACM Press.

Starke, G. (2014). *Effektive Softwarearchitekturen: Ein praktischer Leitfaden*. Munich, Germany: Carl Hanser Verlag.

Stifani, R., Pappe, S., Breiter, G., & Behrendt, M. (2012). IBM Cloud Computing Reference Architecture. *IBM Academy of Technology*, *3*(1). Retrieved from http://www-05.ibm.com/it/cloud/downloads/Cloud_Computing.pdf (Last accessed on 01.12.2015)

Structured Analysis Wiki. (n.d.). *Chapter 9 Dataflow Diagrams - Structured Analysis Wiki.* Retrieved 12.06.2011, from http://yourdon.com/strucanalysis/wiki/index.php/Chapter_9 (Last accessed on 30.06.2015)

Stuehmer, R. (2014, 12). *An RDF Model for Events.* Retrieved from http://www.roland-stuehmer.de/content/rdf-model-events (Last accessed on 30.06.2015)

Sun, A., & Lim, E.-P. (2001). Hierarchical text classification and evaluation. In *Proceedings of the 2001 ieee international conference on data mining* (pp. 521–528). Washington, DC: IEEE Computer Society. Retrieved from http://dl.acm.org/citation.cfm?id=645496.657884

Sun Microsystems. (1988). *RPC: Remote Procedure Call Protocol Specification Version 2.* Retrieved from https://www.ietf.org/rfc/rfc1057.txt

Swets, J. A. (1969). Effectiveness of information retrieval methods. *American Documentation*, *20*(1), 72–89.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011, June). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, *37*(2).

Terrier Team. (2011). *Divergence From Randomness (DFR) Framework.* Retrieved from http://terrier.org/docs/v3.5/dfr_description.html

Teymourian, K., & Paschke, A. (2009). Towards semantic event processing. In *Proceedings of the third acm international conference on distributed event-based systems* (pp. 29:1–29:2). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1619258.1619296 doi: 10.1145/1619258.1619296

Teymourian, K., & Paschke, A. (2010). Enabling knowledge-based complex event processing. In *Proceedings of the 2010 edbt/icdt workshops* (pp. 37:1–37:7). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1754239.1754281 doi: 10.1145/1754239.1754281

The Open Group. (2011). *TOGAF Version 9.1.* Retrieved from http://pubs.opengroup.org/architecture/togaf9-doc/arch/ (Last accessed on 27.11.2015)

Toffler, A. (1970). *Future shock*. New York, NY: Bantam Book.

Tryfonopoulos, C., Koubarakis, M., & Drougas, Y. (2009). Information filtering and query indexing for an information retrieval model. *ACM Transactions on Information Systems*, *27*(2), 10:1–10:47. Retrieved from http://doi.acm.org/10.1145/1462198.1462202 doi: 10.1145/1462198.1462202

Tucker, A. B. (2004). *Computer Science Handbook, Second Edition*. Baco Raton, FL: Chapman and Hall/CRC.

Vaishnavi, V., & Kuechler, B. (2004, January). *Design science research in information systems.* Retrieved from http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf (Last accessed on 15.11.2015)

Vincent, P. (2008, July). *Complex event processing for real-time commerce.* Washington, DC. Retrieved from http://www.omg.org/news/meetings/workshops/Real-time_WS_Final_Presentations_2008/Session%206/06-01_Vincent.pdf (Workshop on Distributed Object Computing for Real-time and Embedded Systems. Last accessed on 10. 11. 2015)

Vincent, P. (2009). *CEP versus ESP - an essay (or maybe a rant).* Retrieved 21.08.2009, from http://www.tibco.com/blog/2009/08/21/cep-versus-esp-an-essay-or-maybe-a-rant/ (Last accessed on 17.06.2015)

Vincent, P. (2010). *The CEP Market 2010: Extending Capability.* Retrieved 12.01.2010, from http://www.tibco.com/blog/2010/01/12/the-cep-market-2010-extending-capability/ (Last accessed on 17.06.2012)

Voorhees, E. M. (2003). Overview of TREC 2003. *TREC*, 1–13. Retrieved from http://trec.nist.gov/pubs/trec12/papers/OVERVIEW.12.pdf

Vouzoukidou, N., Amann, B., & Christophides, V. (2012). Processing continuous text queries featuring non-homogeneous scoring functions. In *Proceedings of the 21st acm international conference on information and knowledge management* (pp. 1065–1074). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/2396761.2398404 doi: 10.1145/2396761.2398404

Wang, F., Liu, S., Liu, P., & Bai, Y. (2006). Bridging physical and virtual worlds: complex event processing for rfid data streams. In *Advances in database technology-edbt 2006* (pp. 588–607). Heidelberg, Germany: Springer.

Wang, X., Fang, H., & Zhai, C. (2007). Improve retrieval accuracy for difficult queries using negative feedback. In *Proceedings of the sixteenth acm conference on conference on information and knowledge management* (pp. 991–994). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/1321440.1321593 doi: 10.1145/1321440.1321593

Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., & Keogh, E. (2012, February). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, *26*(2), 275–309.

Wang, X., Wei, F., Liu, X., Zhou, M., & Zhang, M. (2011). Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. *CIKM*, 1031–1040.

Wasserkrug, S., Gal, A., Etzion, O., & Turchin, Y. (2008). Complex event processing over uncertain data. *Proceedings of the second international conference on Distributed event-based systems*, 253–264.

Webconfs. (2013). *Stop Words.* Retrieved from http://www.webconfs.com/stop-words.php (Last accessed on 17.05.2015)

Weng, J., & Lee, B.-S. (2011). *Event detection in twitter.* AAAI Press. Retrieved from https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2767

Westermann, U., & Jain, R. (2007, January). Toward a common event model for multimedia applications. *IEEE MultiMedia*, *14*(1), 19–29. Retrieved from http://dx.doi.org/10.1109/MMUL.2007.23 doi: 10.1109/MMUL.2007.23

White, F. E. (1991). *Data fusion lexicon* (Tech. Rep. No. 144275). San Diego, CA: Joint Directors of Laboratories.

White, J. E. (1976). *RFC707: A High-Level Framework for Network-Based Resource Sharing* (Tech. Rep.). Retrieved from https://tools.ietf.org/html/rfc707 (Last accessed on 13.11.2015)

Wickham, H. (n.d.). *ggplot2 0.9.3.1 - Box and whiskers plot.* Retrieved from http://docs.ggplot2.org/0.9.3.1/geom_boxplot.html (Last accessed on 11.12.2015)

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, *23*(1), 69–101.

Wikipedia. (2015a). *F1 score - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=F1_score&oldid=652994012 (Last accessed on 31.05.2015)

Wikipedia. (2015b). *Google Real-Time Search - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Google_Real-Time_Search&oldid=636347555 (Last accessed on 30.06.2015)

Wikipedia. (2015c, October). *Heuristic - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Heuristic&oldid=684185658 (Last accessed on 10.10.2015)

Wikipedia. (2015d, June). *Hype cycle - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Hype_cycle&oldid=661202056 (Last accessed on 12.06.2015)

Wikipedia. (2015e, October). *Information retrieval - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=685819601 (Last accessed on 19.10.2015)

Wikipedia. (2015f). *Lunch - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Lunch&oldid=682770218 (Last accessed on 28.09.2015)

Wikipedia. (2015g, June). *Pearson product-moment correlation coefficient - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Pearson_product-moment_correlation_coefficient&oldid=680206318 (Last accessed on 13.09.2015)

Wikipedia. (2015h). *Real-time computing - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Real-time_computing&oldid=655615435 (Last accessed on 16.05.2015)

Wikipedia. (2015i, June). *RM-ODP - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=RM-ODP&oldid=670345015 (Last accessed on 30.06.2015)

Wikipedia. (2015j, June). *Spearman's rank correlation coefficient - Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Spearman%27s_rank_correlation_coefficient&oldid=680219041 (Last accessed on 13.09.2015)

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Amsterdam: Elsevier Science.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., ... Steinberg, D. (2007, December). Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, *14*(1), 1–37. Retrieved from http://dx.doi.org/10.1007/s10115-007-0114-2 doi: 10.1007/s10115-007-0114-2

Xu, J., & Croft, W. B. (2000, January). Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, *18*(1), 79–112. Retrieved from http://doi.acm.org/10.1145/333135.333138 doi: 10.1145/333135.333138

Yang, Y. (2001). A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 137–145). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/383952.383975 doi: 10.1145/383952.383975

Yang, Y., Ault, T., Pierce, T., & Lattimer, C. W. (2000). Improving text categorization methods for event tracking. In *Proceedings of the 23rd annual international acm sigir conference on research and development in information retrieval* (pp. 65–72). New York, NY: ACM. Retrieved from http://doi.acm.org/10.1145/345508.345550 doi: 10.1145/345508.345550

Yang, Y., & Mao, G. (2013). A Self-Adaptive Sliding Window Technique for Mining Data Streams. In *Intelligence computation and evolutionary computation* (pp. 689–697). Berlin, Germany: Springer Berlin Heidelberg.

Yogatama, D., Wang, C., Routledge, B. R., Smith, N. A., & Xing, E. P. (2014). Dynamic Language Models for Streaming Text. *TACL*, 181–192.

Zang, C., & Fan, Y. (2007, February). Complex event processing in enterprise information systems based on rfid. *Enterp. Inf. Syst.*, *1*(1), 3–23. Retrieved from http://dx.doi.org/10.1080/17517570601092127 doi: 10.1080/17517570601092127

Zappavigna, M. (2012). *Discourse of Twitter and Social Media: How We Use Language to Create Affiliation on the Web*. London: Bloomsbury Academic.

Zhang, J., Chen, S., Liu, Y., Yin, J., Wang, Q., Xu, W., & Guo, J. (2012). PRIS at 2012 Microblog Track. *TREC 2012*.

Zhang, Y., & Callan, J. (2001a). A generative model for filtering thresholds. In *Workshop on language modeling and information retrieval.*

Zhang, Y., & Callan, J. (2001b). Maximum likelihood estimation for filtering thresholds. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 294–302).

Zhou, Q., Simmhan, Y., & Prasanna, V. (2012). *Scepter: Semantic complex event processing over end-to-end data flows* (Tech. Rep. No. 12-926). Retrieved from http://www.cs.usc.edu/assets/001/82856.pdf (Last accessed on 13.11.2015)

Zhu, H. (2005). *Software Design Methodology*. Oxford: Elsevier.

Zikopoulos, P. C., deRoos, D., Parasuraman, K., Deutsch, T., Corrigan, D., & Giles, J. (2012). *Harness the Power of Big Data: The IBM Big Data Platform*. New York, NY: McGraw-Hill Osborne Media.

Zubiaga, A., Spina, D., Martínez, R., & Fresno, V. (2015, 03). Real-time classification of Twitter trends. *Journal of the Association for Information Science and Technology*, *66*(3), 462–473.

Zuva, K., & Zuva, T. (2012). Evaluation of Information Retrieval Systems. *International Journal of Computer Science & Information Technology*, *4*(3). Retrieved from http://airccse.org/journal/jcsit/0612csit04.pdf

Appendix A

Structural analysis of Events and ICWSM corpus

In this appendix the details of the structural analysis for the Events and the ICWSM corpus are presented. The analysis followed the same structure as the analysis for the Trec corpus in section 7.5.

Events 2012 corpus

ICWSM 2012 corpus

Finally the *ICWSM 2011* corpus features are studied as a third example. Figure A.1 shows the progress for the *average lemma count* property, which is exactly the same as for the aforementioned two corpora. Also the statistical values as well as the time warping analysis are similar.

	80 sec	40 sec	20 sec	10 sec	5 sec
80 sec	1	0.980	0.905	0.139	0.104
40 sec	0.980	1	0.922	0.133	0.101
20 sec	0.905	0.922	1	0.121	0.090
10 sec	0.139	0.133	0.121	1	0.314
5 sec	0.104	0.101	0.090	0.314	1

Table A.1: Correlation calculated using Pearson's product moment for average document length Trec 2011

	2 sec	5 sec	10 sec	20 sec	40 sec	80 sec	160 sec	random
2 sec	0	3,301.77	3,681.51	3,545.88	3,619.39	3,662.31	3,605.52	4,191.59
5 sec	3,301.77	0	2,173.59	2,684.13	2,900.73	3,050.48	2,624.06	4,730.90
10 sec	3,681.51	2,173.59	0	2,679.64	3,106.27	3,337.40	1,930.24	4,739.46
20 sec	3,545.88	2,684.13	2,679.64	0	890.31	956.02	1,882.87	4,630.51
40 sec	3,619.39	2,900.73	3,106.27	890.31	0	362.85	2,146.23	4,721.93
80 sec	3,662.31	3,050.48	3,337.40	956.02	362.85	0	2,246.82	4,743.28
160 sec	3,605.52	2,624.06	1,930.24	1,882.87	2,146.23	2,246.82	0	4,874.46
random	4,191.59	4,730.90	4,739.46	4,630.51	4,721.93	4,743.28	4,874.46	0

Table A.2: Temporal similarity of average lemma count time series for Events 2012 corpus calculated using dynamic time warping

	80 sec	40 sec	20 sec	10 sec	5 sec
80 sec	1	0.980	0.905	0.139	0.104
40 sec	0.980	1	0.922	0.133	0.101
20 sec	0.905	0.922	1	0.121	0.090
10 sec	0.139	0.133	0.121	1	0.314
5 sec	0.104	0.101	0.090	0.314	1

Table A.3: Correlation calculated using Pearson's product moment for average document length ICWSM 2011

	80 sec	40 sec	20 sec	10 sec	5 sec
80 sec	1	0.973	0.896	0.059	0.037
40 sec	0.973	1	0.905	0.060	0.039
20 sec	0.896	0.905	1	0.062	0.044
10 sec	0.059	0.060	0.062	1	0.350
5 sec	0.037	0.039	0.044	0.350	1

Table A.4: Correlation calculated using Spearman's rank corrleation for average document length ICWSM 2011

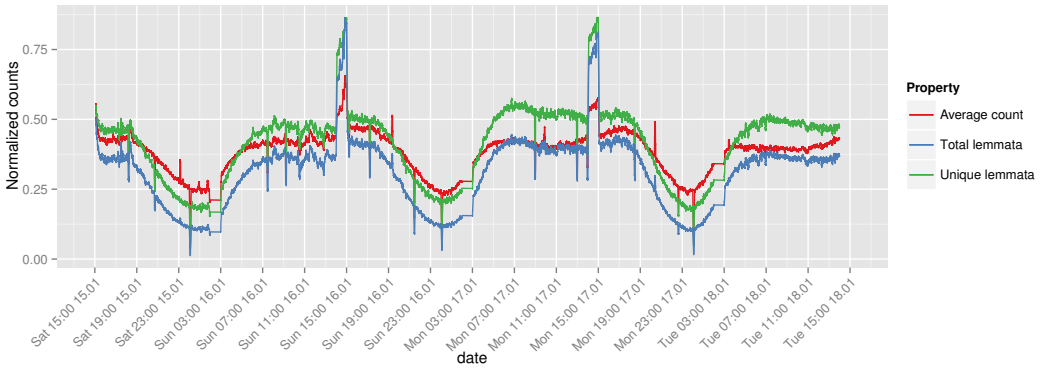

Figure A.1: Time series progress of *average count per lemma* for the *ICWSM 2011* corpus

	5 sec	10 sec	20 sec	40 sec	80 sec	random
5 sec	0	2,173.59	2,684.13	2,900.73	3,050.48	4,782.16
10 sec	2,173.59	0	2,679.64	3,106.27	3,337.40	4,769.40
20 sec	2,684.13	2,679.64	0	890.31	956.02	4,630.92
40 sec	2,900.73	3,106.27	890.31	0	362.85	4,723.56
80 sec	3,050.48	3,337.40	956.02	362.85	0	4,746.50
random	4,782.16	4,769.40	4,630.92	4,723.56	4,746.50	0

Table A.5: Temporal similarity of average lemma count time series for ICWSM 2011 corpus calculated using dynamic time warping

Appendix B

Search queries TREC 2011 and Events 2012 corpus

This appendix contains the topics (or query strings) that were used to gather the results for chapter 8. Column *topic* contains the query terms that were used and column *issued at* shows when the query was issued, i.e. from which point in time the system could filter documents for a given topic.

ID	Topic	Issued at
2	2022 FIFA soccer	Tue Feb 08 18:51:44 +0000 2011
3	Haiti Aristide return	Tue Feb 08 21:32:13 +0000 2011
4	Mexico drug war	Wed Jan 23 04:26:57 +0000 2011
5	NIST computer security	Fri Feb 04 17:44:09 +0000 2011
7	Pakistan diplomat arrest murder	Thu Jan 27 14:36:55 +0000 2011
8	Phone hacking British politicians	Sun Jan 23 04:20:34 +0000 2011
9	Toyota Recall	Tue Jan 25 01:25:03 +0000 2011
10	Egyptian protesters attack museum	Sat Jan 29 20:06:35 +0000 2011
12	Assange Nobel peace nomination	Mon Jan 31 21:02:33 +0000 2011
13	Oprah Winfrey half-sister	Mon Jan 24 15:43:41 +0000 2011
14	Release of The Rite	Wed Feb 02 12:31:02 +0000 2011
17	White Stripes breakup	Mon Jan 24 01:57:18 +0000 2011
19	Cuomo budget cuts	Mon Jan 24 16:17:23 +0000 2011
20	Taco Bell filling lawsuit	Sun Jan 23 16:16:28 +0000 2011
22	Health care law unconstitutional	Mon Jan 31 07:05:11 +0000 2011
23	Amtrak train service	Mon Jan 24 01:49:59 +0000 2011
24	Super Bowl seats	Thu Feb 03 15:57:33 +0000 2011
25	TSA airport screening	Sun Jan 23 03:41:51 +0000 2011
27	Reduce energy consumption	Sun Jan 23 03:55:25 +0000 2011
28	Detroit auto show	Wed Jan 26 22:46:12 +0000 2011
29	Global warming weather	Sun Jan 23 07:02:55 +0000 2011
30	Keith Olberman new job	Tue Jan 25 03:23:08 +0000 2011
32	State of the union and jobs	Sun Jan 23 01:53:45 +0000 2011
33	Dog Whisperer Cesar Millans techniques	Thu Jan 27 19:27:54 +0000 2011
34	MSNBC Rachel Maddow	Fri Feb 04 22:42:20 +0000 2011
35	Sargent Shriver tributes	Mon Jan 24 07:18:17 +0000 2011
37	Giffords recovery	Sun Jan 23 01:59:03 +0000 2011
38	Protests in Jordan	Tue Feb 01 12:46:40 +0000 2011
39	Egyptian curfew	Fri Jan 28 18:14:09 +0000 2011

Table B.1: Topics TREC 2011 corpus.

ID	Topic	Issued at
40	Beck attacks Piven	Mon Jan 31 20:33:37 +0000 2011
42	Holland Iran envoy recall	Mon Feb 07 20:47:13 +0000 2011
43	Kucinich olive pit lawsuit	Sat Jan 29 08:06:05 +0000 2011
44	White House spokesman replaced	Fri Jan 28 13:35:45 +0000 2011
45	Political campaigns and social media	Sun Jan 23 21:41:06 +0000 2011
47	Organic farming requirements	Tue Feb 08 00:12:47 +0000 2011
48	Egyptian evacuation	Mon Jan 31 09:36:57 +0000 2011

Table B.2: Topics TREC 2011 corpus continued.

ID	Topic	Issued at
14	Giants Tigers win world series	Sun Oct 28 01:16:00 +0000 2012
43	Sandy landfall Atlantic City	Sun Oct 28 17:30:00 +0000 2012
46	Purchase Lucasfilm Walt Disney	Tue Oct 30 20:30:00 +0000 2012
49	Cowboys Giants game	Sun Oct 28 13:00:00 +0000 2012
50	LA Lakers Mavericks	Mo Oct 29 12:00:00 +0000 2012
51	Suicide Silence Lucker	Thu Nov 01 11:09:00 +000 2012
53	Arsenal Reading league cup	Tue Oct 30 22:12:00 +0000 2012
55	Parker Spurs buzzer beater	Thu Nov 01 21:00:00 +0000 2012
56	Cancelled New York Marathon	Wed Oct 31 23:23:00 +0000 2012
74	Syrian Air commander assassinated Damascus	Mo Oct 29 16:59:00 +0000 2012
146	Suarez celebrate Liverpool	Sun Oct 28 04:41:00 +0000 2012
154	Pittsburgh steelers uniform	Sun Oct 28 10:00:00 +0000 2012
155	Packers Bears	Sun Oct 28 17:30:00 +0000 2012
162	Country Music Award	Thu Nov 01 17:00:00 +0000 2012
163	Sporting event	Mon Oct 29 19:23:00 +0000 2012
164	Lebron NBA championship rings	Tue Oct 30 11:36:00 +0000 2012
167	Taylor Swift CMA	Thu Nov 01 18:02:00 +0000 2012
169	Carrie Marie Underwood CMA	Thu Nov 01 19:00:00 +0000 2012
170	Blake Shelton Mirinda Lambert	Thu Nov 01 19:07:00 +0000 2012
187	Vaxevanis arrested Swiss bank accounts	Sun Oct 28 00:23:00 +0000 2012
249	Fuel tank explode Riyadh	Wed Oct 31 02:32:00 +0000 2012
254	Child perjury Spanier	Thu Nov 01 15:39:00 +0000 2012
256	Bloomberg endorses Obama	Thu Nov 01 22:19:00 +0000 2012
265	Bo Xiali immunity	Fri Oct 26 05:00:00 +0000 2012
434	People evacuated coast India	Thu Nov 01 15:32:00 +0000 2012
436	Sandusky child abuse	Mon Oct 29 18:24:00 +0000 2012
454	Bahrain protests banned	Tue Oct 30 00:06:00 +0000 2012
456	Penguin Random House merger	Mon Oct 29 00:27:00 +0000 2012
484	Attacks Catholic church Nigeria	Sun Oct 28 01:05:00 +0000 2012
486	SpaceX ISS refuel	Sun Oct 28 00:42:00 +0000 2012
488	Glitter arrested sex offences	Sun Oct 28 01:29:00 +0000 2012

Table B.3: Topics Events 2012 corpus.

Appendix C

Further results filter policy evaluation

This appendix contains additional evaluation data of the various filter policies as discussed in section 8.2.4.2 for the Events 2012 corpus.

	Atemporal	Temporally unambiguous	Temporally ambiguous
Precision			
Baseline + Classifier	0.1444	**0.1514**	0.1235
Frequent Pattern	0.0919	**0.1262**	0.0962
Baseline	0.1215	**0.1352**	0.1076
Recall			
Baseline + Classifier	0.1922	**0.2024**	0.0749
Frequent Pattern	0.2357	**0.3247**	0.1480
Baseline	0.4194	**0.4360**	0.2065
F1-Measure			
Baseline + Classifier	**0.1721**	0.1034	0.0934
Frequent Pattern	0.0980	**0.1195**	0.0782
Baseline	**0.1352**	0.1311	0.0927
Sensitivity			
Baseline + Classifier	0.5801	**0.8185**	0.5500
Frequent Pattern	0.5482	**0.6860**	0.5851
Baseline	0.6161	**0.7122**	0.5996
T11SU			
Baseline + Classifier	0.1473	0.1246	**0.1876**
Frequent Pattern	0.0496	0.0688	**0.1045**
Baseline	0.0714	0.0720	**0.1055**

Table C.1: Evaluation of filter policies applied on the Events 2012 corpus

Appendix D

Further results term weighting evaluation

This appendix contains the evaluation results for the term weighting schemes presented in section 8.3 applied to the Events 2012 corpus. As a reminder the precision, recall, f1 and T11SU values are macro-averaged values just as in the other evaluations throughout the thesis.

Term weighting	Category	TP	FP	FN	Precision	Recall	F1	T11SU
Burst	Atemporal	608	1335	232	0.3129	0.6933	0.3515	0.7238
DfIdf	Atemporal	639	4663	640	0.1977	0.4074	0.2279	0.6542
Boolean	Atemporal	827	1791	452	0.1795	0.3876	0.2280	0.5027
ArithmeticMeanScorer	Atemporal	1013	5555	266	0.1674	0.4826	0.1813	0.7353
TerrerInL2	Atemporal	914	4555	365	0.1635	0.4351	0.1786	0.6910
LanguageModel	Atemporal	1069	4943	210	0.1395	0.4984	0.1592	0.7799
GeometricMeanScorer	Atemporal	1039	5161	240	0.1315	0.4705	0.1786	0.5978
HarmonicMeanScorer	Atemporal	914	3605	365	0.1282	0.4340	0.1760	0.5582
HarmonicMeanScorer2	Atemporal	980	4935	299	0.1236	0.4470	0.1700	0.5631
TFIDF	Atemporal	1038	5148	241	0.1202	0.4868	0.1387	0.6983
TerrierDFRBose	Atemporal	901	4257	378	0.1140	0.4234	0.1572	0.5455
IDFOnly	Atemporal	902	4252	377	0.1137	0.4236	0.1569	0.5457
RegularOkapi	Atemporal	1047	5288	232	0.1124	0.4909	0.1565	0.6244
TerrierPl2	Atemporal	1055	5344	224	0.1124	0.4933	0.1566	0.6270
Poisson	Atemporal	949	4446	127	0.1015	0.3418	0.1717	0.6486
ContextScorer	Atemporal	910	6427	369	0.0993	0.4345	0.1371	0.5470
ProbScorer	Atemporal	553	4074	523	0.0970	0.2349	0.1976	0.4513

Continued on next page

Term weighting	Category	TP	FP	FN	Precision	Recall	F1	T11SU
LuceneOkapi	Atemporal	336	2905	943	0.0921	0.2478	0.1164	0.3414
Boolean	Unambiguous	1177	21103	820	0.2061	0.3254	0.1365	0.4401
TerrerInL2	Unambiguous	1134	48641	863	0.1963	0.3095	0.1191	0.4259
ArithmeticMeanScorer	Unambiguous	938	106263	1059	0.1704	0.2486	0.0876	0.3596
GeometricMeanScorer	Unambiguous	1371	41475	626	0.1683	0.3609	0.1181	0.5166
TFIDF	Unambiguous	1273	62139	724	0.1473	0.3923	0.1267	0.5817
HarmonicMeanScorer	Unambiguous	1397	43840	600	0.1471	0.4185	0.1354	0.6253
LuceneOkapi	Unambiguous	1338	82365	659	0.1393	0.3716	0.1395	0.6307
RegularOkapi	Unambiguous	1502	61972	495	0.1376	0.4655	0.1330	0.7287
TerrierPl2	Unambiguous	1499	63781	498	0.1376	0.4653	0.1334	0.7305
IDFOnly	Unambiguous	1616	53751	381	0.1348	0.5066	0.1380	0.8201
TerrierDFRBose	Unambiguous	1624	53929	372	0.1345	0.5153	0.1375	0.8351
HarmonicMeanScorer2	Unambiguous	1010	68307	987	0.1261	0.3271	0.1173	0.4817
ContextScorer	Unambiguous	1303	102212	694	0.1214	0.4131	0.1202	0.7129
LanguageModel	Unambiguous	1053	64486	944	0.1198	0.3150	0.1278	0.4352
DfIdf	Unambiguous	761	111960	1236	0.0957	0.2095	0.0771	0.2894
Poisson	Unambiguous	633	26560	610	0.0893	0.1552	0.0582	0.2953

Continued on next page

Term weighting	Category	TP	FP	FN	Precision	Recall	F1	T11SU
ProbScorer	Unambiguous	647	23722	596	0.0748	0.1552	0.0612	0.3120
Burst	Unambiguous	80	2190	311	0.0391	0.1526	0.0504	0.5413
Boolean	Ambiguous	963	7838	680	0.1513	0.1489	0.1775	0.2542
GeometricMeanScorer	Ambiguous	1061	16597	582	0.1486	0.1781	0.1472	0.5330
LanguageModel	Ambiguous	1025	14832	618	0.1462	0.2240	0.1142	0.5303
ArithmeticMeanScorer	Ambiguous	1301	16447	342	0.1434	0.2645	0.1724	0.6141
TerrerInL2	Ambiguous	1215	11575	428	0.1422	0.2318	0.1738	0.4514
DfIdf	Ambiguous	1273	14433	370	0.1301	0.2617	0.1588	0.5875
TFIDF	Ambiguous	859	15820	784	0.1227	0.1958	0.0975	0.4954
HarmonicMeanScorer	Ambiguous	969	13757	674	0.1188	0.1749	0.0966	0.4972
TerrierPl2	Ambiguous	969	17901	674	0.1134	0.2169	0.0958	0.5996
RegularOkapi	Ambiguous	958	17730	685	0.1125	0.2084	0.0948	0.5846
HarmonicMeanScorer2	Ambiguous	659	15362	984	0.1078	0.1732	0.0838	0.4445
LuceneOkapi	Ambiguous	911	10517	732	0.1046	0.1725	0.1083	0.3872
ContextScorer	Ambiguous	1077	21449	566	0.0971	0.2742	0.1050	0.6351
TerrierDFRBose	Ambiguous	916	16047	725	0.0949	0.1695	0.0871	0.4922
IDFOnly	Ambiguous	846	15990	797	0.0918	0.1625	0.0868	0.4608

Continued on next page

Term weighting	Category	TP	FP	FN	Precision	Recall	F1	T11SU
ProbScorer	Ambiguous	906	12732	186	0.0815	0.1336	0.0867	0.8067
Poisson	Ambiguous	855	13235	237	0.0727	0.1254	0.0829	0.8101
Burst	Ambiguous	488	7038	351	0.0512	0.2157	0.0603	0.5289

Table D.2: Evaluation of term weights applied on Events 2012 corpus per temporal category.

Appendix E

Further results threshold evaluation

This appendix contains additional evaluation results for the Events corpus using the threshold methods presented in section (cf. 8.4.4.1). As a reminder the precision, recall, f1 and T11SU values are macro-averaged values just as in the other evaluations throughout the thesis. Table E.2 summarises the results. It must be noted that the results are biased for the Events corpus. The precision values are in general poor and do not differ between categories. This supports the idea that the relevance feedback is poor, as no clear distinction in the results is observable. But this distinction could be expected if the search profiles had reasonable relevance assessments. The TREC corpus shows clearly how the distinction should look like. In terms of recall and F1 measure the AverageScore performed best. This supports the finding from the other score based threshold evaluations that a straight forward approach works – in doubt – best.

Term weighting	Category	TP	FP	FN	Precision	Recall	F1
TopKThreshold	Atemporal	58	225	1221	0,1733	0,0654	0,1630
Percentile	Atemporal	453	1863	826	0,1409	0,2678	0,1800
StddevOverallScore	Atemporal	664	2469	615	0,1365	0,3398	0,1823
AveragePositiveScoreStrategy	Atemporal	714	2915	565	0,1359	0,3871	0,1826
MovingAveragePerQuery	Atemporal	727	4228	552	0,1287	0,3922	0,1417
PercentilePerQuery	Atemporal	195	1228	1084	0,1215	0,1461	0,1303
HarmonicMeanThreshold	Atemporal	450	2182	829	0,1204	0,2564	0,1560
AverageScorePerQuery	Atemporal	738	4543	541	0,1203	0,4074	0,1336
StddevPerQuery	Atemporal	357	2190	922	0,1141	0,2484	0,1419
AverageScore	Atemporal	1047	5276	232	0,1125	0,4909	0,1568
GaussianExponentialThreshold	Atemporal	644	4787	635	0,1103	0,4104	0,1226
AveragePositiveScorePerQuery	Atemporal	290	1845	989	0,1100	0,1884	0,1115
ExponentialMovingAveragePerTotalMatches	Atemporal	836	5535	443	0,1017	0,4024	0,1168
StatsPerSourceMatches	Atemporal	419	2640	860	0,0961	0,1968	0,1249
StatsPerSourceMatchesPerQuery	Atemporal	306	2796	973	0,0868	0,1909	0,1080
TopKThreshold	Unambiguous	148	869	1849	0,1764	0,0721	0,1168
AveragePositiveScoreStrategy	Unambiguous	955	24646	1042	0,1586	0,2831	0,1167

Continued on next page

Term weighting	Category	TP	FP	FN	Precision	Recall	F1
StddevOverallScore	Unambiguous	930	23936	1067	0,1572	0,2722	0,1158
Percentile	Unambiguous	744	15059	1253	0,1493	0,2095	0,1099
MovingAveragePerQuery	Unambiguous	1279	68019	718	0,1454	0,4052	0,1366
AverageScore	Unambiguous	1502	61907	495	0,1377	0,4655	0,1331
PercentilePerQuery	Unambiguous	586	18263	1411	0,1365	0,1997	0,1154
HarmonicMeanThreshold	Unambiguous	846	21626	1151	0,1351	0,2735	0,1239
AverageScorePerQuery	Unambiguous	1372	69357	625	0,1317	0,4385	0,1291
ExponentialMovingAveragePerTotalMatches	Unambiguous	1291	73434	706	0,1312	0,4229	0,1285
StddevPerQuery	Unambiguous	929	30827	1068	0,1270	0,3233	0,1211
StatsPerSourceMatchesPerQuery	Unambiguous	953	34076	1044	0,1258	0,3350	0,1208
GaussianExponentialThreshold	Unambiguous	1354	122559	643	0,1238	0,4495	0,1219
AveragePositiveScorePerQuery	Unambiguous	805	17985	1192	0,1235	0,2963	0,1177
StatsPerSourceMatches	Unambiguous	648	25045	1349	0,1224	0,2175	0,1057
TopKThreshold	Ambiguous	86	496	1557	0,1455	0,0298	0,0984
AveragePositiveScoreStrategy	Ambiguous	636	8169	1007	0,1331	0,1321	0,0939
MovingAveragePerQuery	Ambiguous	939	15128	704	0,1247	0,1993	0,1005
Percentile	Ambiguous	391	4542	1252	0,1233	0,0722	0,1009

Continued on next page

Term weighting	Category	TP	FP	FN	Precision	Recall	F1
StddevOverallScore	Ambiguous	482	6973	1161	0,1218	0,0987	0,0934
AverageScore	Ambiguous	959	17713	684	0,1126	0,2085	0,0949
AverageScorePerQuery	Ambiguous	951	15832	692	0,1072	0,2057	0,0924
HarmonicMeanThreshold	Ambiguous	363	6073	1280	0,0999	0,0846	0,0836
StddevPerQuery	Ambiguous	507	7636	1136	0,0996	0,1101	0,0871
GaussianExponentialThreshold	Ambiguous	1177	21294	466	0,0979	0,2600	0,0915
PercentilePerQuery	Ambiguous	362	4159	1281	0,0931	0,0752	0,0743
ExponentialMovingAveragePerTotalMatches	Ambiguous	712	18848	931	0,0869	0,1745	0,0735
StatsPerSourceMatchesPerQuery	Ambiguous	377	9848	1266	0,0840	0,1068	0,0678
AveragePositiveScorePerQuery	Ambiguous	343	6461	1300	0,0748	0,0822	0,0680
StatsPerSourceMatches	Ambiguous	230	7603	1413	0,0641	0,0568	0,0504

Table E.2: Evaluation of score based threshold strategies applied on Events 2012 corpus per temporal category.

Table E.3 shows the results for the various machine learning classifiers. The precision values are poor, what is due to quality of the relevance assessments. The recall values are better, but if a look at the total figures is taken, that the models yielded many false positive, i.e. they judged nearly every document as relevant. Thus, in a scenario where relevance feedback is scarce or poor, machine learning does not work very well.

The performance per profile category of the classifiers in shown in table E.5.

Classifier	Category	TP	FP	FN	Precision	Recall	F1
Passive Aggressive	Atemporal	349	5329	930	0.0923	0.2173	0.1103
Random Forest	Atemporal	653	8221	626	0.0819	0.3555	0.0908
Plat SMO	Atemporal	846	8168	433	0.0969	0.4036	0.1090
StochasticMultinomialLogReg	Atemporal	34	555	1245	0.1028	0.0126	0.0535
LinearSGD	Atemporal	1217	9011	62	0.1127	0.6238	0.1317
Passive Aggressive	Unambig.	648	104923	1349	0.1157	0.2187	0.1070
Random Forest	Unambig.	1131	159378	866	0.1101	0.3698	0.1089
Plat SMO	Unambig.	1290	160613	707	0.1147	0.4134	0.1141
StochasticMultinomialLogReg	Unambig.	38	3705	1959	0.0433	0.0184	0.0491
LinearSGD	Unambig.	1910	160896	87	0.1266	0.5833	0.1291
Passive Aggressive	Ambig.	723	18271	920	0.0773	0.2065	0.0833
Random Forest	Ambig.	1153	30488	490	0.0788	0.2887	0.0760
Plat SMO	Ambig.	1256	31143	387	0.0873	0.3036	0.0826
StochasticMultinomialLogReg	Ambig.	45	888	1598	0.0836	0.0304	0.0466
LinearSGD	Ambig.	1508	31962	135	0.1035	0.3512	0.0957

Table E.3: Performance evaluation on profile category level of various machine learning classifiers applied on the Events 2012 corpus.

Classifier	Category	TP	FP	FN	TN
kNN	1	263	13718	59	319
LinearSGD	1	231	10238	91	3799
NaiveBayes	1	169	8751	153	5286
Passive Aggressive	1	150	7371	172	6666
Plat SMO	1	261	13689	61	348
Random Forest	1	291	13429	31	608
StochasticMultinomialLogReg	1	251	9753	71	4284
kNN	2	306	910	76	69
LinearSGD	2	281	730	101	249
NaiveBayes	2	184	616	198	363
Passive Aggressive	2	157	518	225	461
Plat SMO	2	309	911	73	68
Random Forest	2	318	893	64	86
StochasticMultinomialLogReg	2	290	735	92	244
kNN	3	277	1714	63	107
LinearSGD	3	233	1346	107	475
NaiveBayes	3	150	987	190	834

Continued on next page

Classifier	Category	TP	FP	FN	TN
Passive Aggressive	3	120	780	220	1041
Plat SMO	3	272	1705	68	116
Random Forest	3	277	1688	63	133
StochasticMultinomialLogReg	3	270	1372	70	449

Table E.5: Evaluation of classifiers applied on the Trec 2011 corpus per temporal category.