Universität Regensburg

# Classifying user information needs in cooking dialogues – an empirical performance evaluation of transformer networks

Masterarbeit im Fach Digital Humanities am Institut für Information und Medien, Sprache und Kultur (I:IMSK)

Vorgelegt von: Patrick Schwabl
Adresse: Hemauerstraße 20, 93047 Regensburg
E-Mail (Universität): patrick.schwabl@stud.uni-regensburg.de
E-Mail (privat): patrick-schwabl@web.de
Matrikelnummer: 1740937
Erstgutachter: Prof. Dr. Bernd Ludwig
Zweitgutachter: PD Dr. David Elsweiler
Betreuer: Prof. Dr. Bernd Ludwig
Laufendes Semester: 3. Semester M.A. Digital Humanities
Abgegeben am: 04.04.2021

# Abstract

In this master's thesis, I carry out 3720 machine learning experiments. I want to test how transformer networks perform in a dialogue processing task. Transformer networks are deep neural networks that have first been proposed in 2017 and have since rapidly set new state of the art results on many tasks. To evaluate their performance in dialogue classification, I use two tasks from two datasets. One comes from a dialogue; the other does not. I compare various transformer network's F1 scores on these two classification tasks. I also look at many different baseline models, from random forest classifiers to long short-term memory networks. A theoretically derived taxonomy will be used to annotate dialogue data with information on dialogue flow. I will show that modelling human conversation is an intricate task and that more features do not necessarily make classification better. Five hypotheses are tested using statistical methods on the output data from the 3720 experiments. Those analyses show that results are very alike for the same machine learning algorithms on the two different tasks. Beyond performance evaluation, the aim is to use transformers to improve user information need classification. These needs I am examining in this study arise during assisted cooking dialogues with a conversational agent. Generally, I can show that transformer networks achieve better classification results than the established baseline models.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **ANOVA** | Analysis of variance |
| **BERT** | Bidirectional encoder representations from transformers |
| **CPU** | Central processing unit |
| **CV** | Cross-validation |
| **XLM** | Cross-lingual language models |
| **ConvBERT** | Convolutional BERT |
| **CNN** | Convolutional neural network |
| **DistillBERT** | Distilled BERT |
| **Electra** | Efficiently learning an encoder that classifies token replacements accurately |
| **FFN** | Feed-forward neural network |
| **GAN** | Generative adversarial network |
| **GPU** | Graphics processing unit |
| **HSD** | Honestly significant difference |
| **LSTM** | Long short-term memory |
| **ML** | Machine learning |
| **MLM** | Masked language modeling |
| **MLP** | Multi-layer perceptron |
| **NB** | Naive bayes |
| **NLP** | Natural language processing |
| **NSP** | Next sentence prediction |
| **NN** | Neural network |
| **OOV** | Out of vocabulary |
| **RNN** | Recurrent neural network |
| **Tf-idf** | Term frequency–inverse document frequency |
| **TN** | Transformer network |
| **YAML** | Yaml ain't markdown language |

# 1 Transformers - from Bert to BERT

In 2018, a dear childhood memory of many machine learners was being eradicated, or at least replaced. Bert became BERT. The former is a yellow, likeable puppet with a rather pessimistic attitude towards life. He and his friend Ernie live in Sesamstreet with their friends, where they play funny and educational sketches for children.

What on earth does this have to do with machine learning (ML)? Well, Bert, or rather BERT, is also the acronym for Bidirectional Encoder Representations from Transformers, which already sounds a lot more like ML. This is why my connotation with Bert has changed recently; however, not necessarily for the worse. Bert was a pleasant childhood memory. BERT is an exciting new technological advance in the field of natural language processing (NLP). The risk of confusing them led to quite a few memes on the internet. Much of this thesis will be about BERT and the family of ML models it belongs to, the so-called transformers.

BERT is the most widely known of these new deep neural networks. How these transformer networks (TN) work and compare with other ML algorithms when applied to NLP will be the topic of this thesis. This study aims to determine what influence contextual features have on text classification in a dialogue setting and a non-dialogue setting. To this end, different ML models will be compared, with a particular focus on transformer networks. I will use two different NLP tasks. The first concerns the cooking domain; the other is a political science task and lies within voting. Both are classification tasks with short textual input as well as some additional features. However, one data set comes from human dialogues. The other is a collection of independent documents.

Using TN for dialogue processing often uses only textual input as features. If there are others, they are such that they can be computed on the fly during training and classification. That is perfectly sensible since real-world systems have to be operatable and it is very expansive to enrich user input with information annotated by humans. However, if you could model human dialogue in a theoretically well-founded way, and connect this with ML, it will likely increase performance. If the increase is significant enough, realisations that are also feasible in practice will follow.

Consider this the broader aim of the study. I try to evaluate whether the theoretical model of a dialogue improves a TNs understanding of what a human user wants. In this wake, I will also investigate how different TNs compare with each other.

There is comparably little literature on transformers yet, some two or three textbooks on their application in NLP have been published at the time of this writing. Papers are plentiful and the reader will see which matter for this thesis soon. Most notably are Vaswani et al. (2017) and Devlin et al. (2019), were the general idea of a TN and BERT were introduced. For theoretical considerations, I will primarily refer to Wandke (2005). His approach will be used to bring the notion of context into the data.

There are many cognitive theories of human dialogue and researchers already tried to integrate them with machine learning. (Malchanau 2019; Amanova et al. 2016) However, rather then only trying to get information about the human dialogue, the annotations proposed in this thesis aim to enrich utterances with information on the state of the conversational agent as well.

After this introduction, I will start with preliminary remarks on my data, where it comes from, and how it looks like. I explain the labels of both classification tasks as well as textual and other input to the classifiers. In addition to that, there are some theoretical explanations of the contextual features. In part three, I lay out my research design by describing what rationale I will follow in the remainder of this work. I also formulate my research question and hypotheses and give a first overview of the ML experiments I made. Section four provides a methodological overview of transformers and BERT. I will talk about transformers and their extension called multimodal transformers because they are not yet widely known. There are concepts like attention and transfer learning which deserve a more detailed discussion and will also be covered in part four. For all the other classifiers, I refer the reader to the relevant literature. After that, I will explain my concrete architectures and classifier implementation in part five. Here, the hypotheses are also reformulated and specified as to be strictly testable within this thesis's framework. In part six, I perform statistical analyses on all the experimental output data. I use inferential statistical methods to evaluate model performance over 3720 experiments. The last part concludes and gives an outlook.

# 2 Preliminaries

## 2.1 Remarks on data - textual input feature and labels

My data comes from two sources. The first one I will call cooking data from here on, the second I will call voting data. They are from different domains and were gathered in different contexts. Also, both pose different classification problems. Still, they also share some characteristics.

**Table 2.1:** Comparison of data sources

| Cooking data | Voting data |
|---|---|
| Short textual input | Short textual input |
| Whole sentences, incomplete sentences | Whole sentences, word lists |
| 2829 samples | 2211 samples |
| Multi-label classification | Binary classification |
| Gathered during an experiment | Gathered during an opinion poll |
| Dialogue data, samples depend on each other | Each sample is an independent document |

The most obvious difference, of course, are the domains from which each data set comes. One is voting, and one is cooking. Why bring these two together?

The reason is that I have been trying to improve classification results on the above-mentioned cooking corpus for a while when I attended a workshop where one topic was NLP and its use for survey research. The presenters later also published a paper on how pre-trained models and transfer learning (we will see what that is later in section 4.1.1) can help categorize survey respondents based on open-ended questions. (Meidinger and Aßenmacher 2021) I found that I had a comparable task with the cooking data. There are respondents who write whole sentences in open-ended survey questions, and there are such people who give very short replies - sometimes just a few words. This structure is comparable with the cooking dialogue data. Sometimes there are very short utterances, grammatically not even sentences, and sometimes there are whole, correct sentences. The difference is that all the responses to be classified in the voting data are independent of each other because they come from different people. In the cooking data, they are dialogues and consequently not independent of each other.

Meidinger and Aßenmacher (2021) had good results with off the shelf transformer models on their tasks. Therefore I want to bring these models into the domain of dialogue data. To keep the comparison with a non-dialogue task, I decided to include data from a study that I was part of and that was carried out during the Bavarian state election of 2018. In the following section, I will briefly describe both datasets and how they came to be.

Also, the text in both data sets is human-produced, so to say. The dialogue data contains utterances as humans would speak them. The survey data - although written - also contains such informal speech-like sentences while not being dialogical. If you want to compare dialogue data with non-dialogue data, this is crucial. It is hard to think about data of human utterances given as they would be spoken and at the same time do not come from a dialogue.

### 2.1.1 Cooking data

The cooking data were gathered during experiments by Frummet et al. (2019). They conducted in-situ experiments where a human assumed the role of a conversational agent, like an Amazon Alexa. The idea was to simulate how a speech assistant with human-like conversation abilities would interact with users. Subjects were given cooking ingredients and then asked to cook a meal that they have not cooked before. Experiments were done in users own kitchen at mealtime to make the situation as realistic as possible. Therefore, users were also allowed to use ingredients they had in store at home. However, the task still was to use as many of the provided ingredients as possible. To do so, they could ask their conversational agent (i.e. the experimenter) anything. The agent then was allowed to look anything up on the internet and respond to the best of his knowledge. From the whole textual material, 2829 user utterances were selected that were believed to hold a so-called information need. “The term information need is often understood as an individual or group’s desire to locate and obtain information to satisfy a conscious or unconscious need.” (Wikipedia 2020; see also Rather 2018) In the cooking case, this could be the need to get information about how much milk is needed at this step of the recipe. Three rows of the data the experiments resulted in can be seen in table 2.2.

**Table 2.2:** Three rows from the cooking data

| Label | User utterance | Reference | Motiv and goal formation |
|---|---|---|---|
| Recipe | Ähm kannst du mir Gerichte mit Spargel suchen mit möglichst vielen Milchprodukten. | plan | in |
| Recipe | Also ich hab überlegt Spargel mit Tomatenrühreiern dann wären drei Gerichte weg ähm aber das passt glaub ich nicht zusammen und keine Ahnung. | plan | in |
| Recipe | Ja oder äh hast du schon was gefunden? | plan | in |

These are not all the features present in the cooking data, and the table is just meant to give a first idea. Column three and four, for instance, are annotated features that were coded according to the taxonomy, which I will describe later. All 2829 user utterances are annotated with one of 13 information need labels. These labels are *Amount, Cooking technique, Equipment, Ingredient, Knowledge, Meal, Miscellaneous, Preparation, Recipe, Temperature, Time, User Command,* and *User Search Actions.* Some of the names are telling, others not so much. In the following table 2.3, you can see the explanations taken from the annotation guidelines by Frummet (2019).

Just as Frummet et al. (2019) already found when looking into this corpus, I find the labels’ distribution to be extremely skewed (see figure 2.1).

**Table 2.3:** Table of label explanations

| Label | Explanation |
|---|---|
| Amount | Amount related information needs occuring during cooking. |
| Ingredient | Ingredient related information needs occuring during cooking. |
| Preparation | Information needs that concretely relate to the preparation process of a recipe that has been selected. |
| Cooking Technique | Information needs that relate to cooking techniques and preparation steps that aren't specified in the selected recipe. |
| Recipe | Information needs relating to the recipe selection. |
| Time | Information needs relating to temporal aspects of the cooking/preparation process. |
| Equipment | Information needs relating to the needed cooking utensils. |
| Knowledge | Information needs that relate to definitions of certain aspects/entities occurring in the context of cooking. |
| Meal | Information needs that relate to the currently prepared meal. However, these are more general and do not relate to certain preparation steps. |
| Temperature | Information needs relating to temperature. |
| User Command | Utterances of this class cannot be viewed as formal information needs. They include commands users can pose to the system. |
| User Search Actions | Utterances of this class cannot be viewed as formal information needs. They include search actions users can pose to the system. |
| Miscellaneous | Information needs that cannot be assigned to one of the aforementioned classes. |

This makes our cooking classification task even more challenging. Some labels only have a dozen or even fewer samples, while Preparation has almost 800. The three top classes easily account for half of all cases (i.e. samples). In section 3.3, I will discuss what problems this causes and how they can be mitigated. The utterances column of table 2.2 shows what participants said to the conversational agent. This varies in length; the shortest utterances are one word long while the longest one contains 95 words.

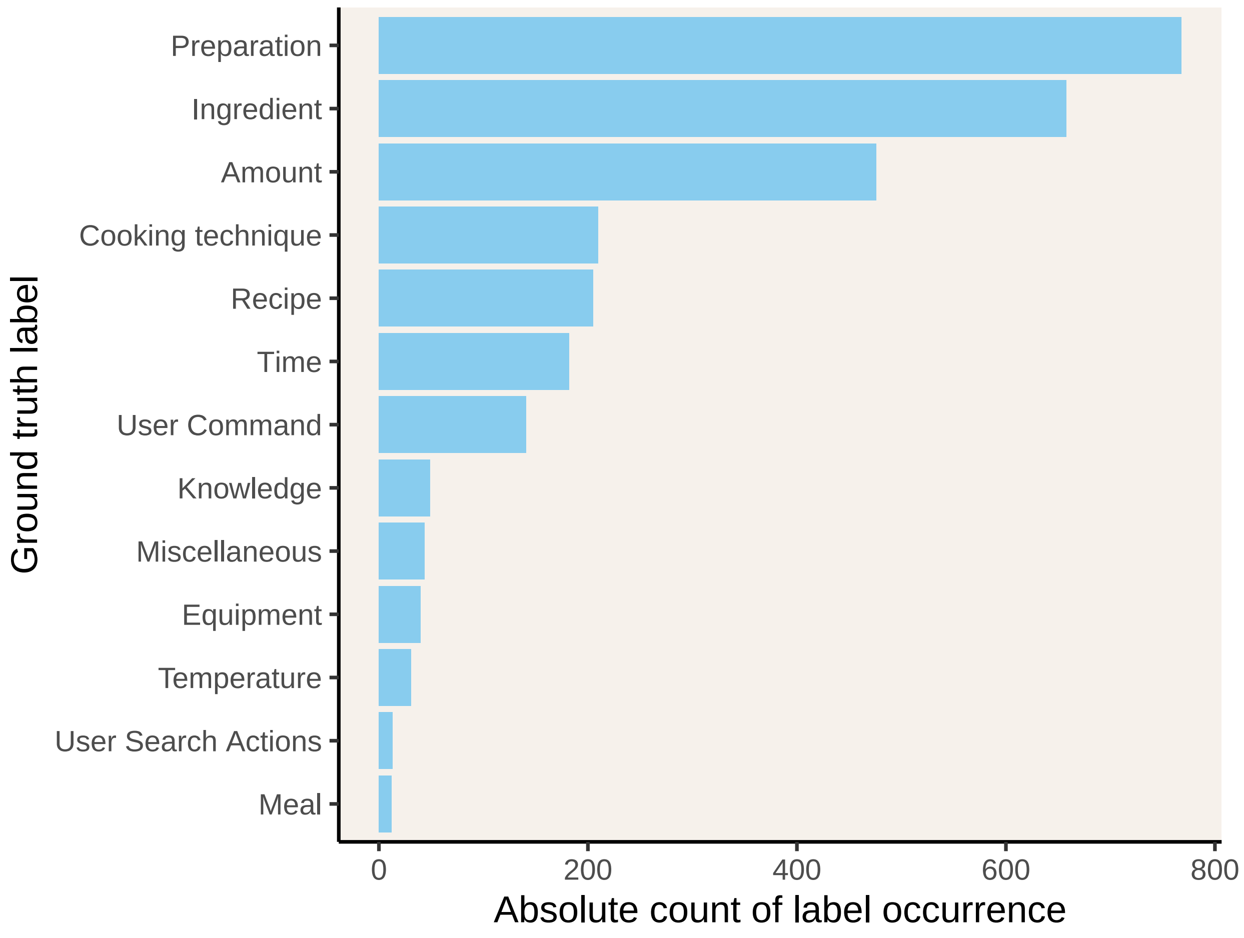


**Figure 2.1:** Distribution of ground truth labels - cooking data

### 2.1.2 Voting data

The origin of this data set is a study carried out during the Bavarian state election of 2018. It was done by the chair of Political Science Research Methods in Regensburg. Our team gathered data from over 16,000 voters and asked them open-ended as well as closed questions. We surveyed people right after they came out of the voting booth - a method called exit polls. A book with the results and exact methodology of the study will be published in 2021, see Heinrich and Walter-Rogg (in appearance).

2211 of these roughly 16,000 repsondents replied to the open-ended questions and will therefore be used in the remainder of this study. The wording of the two open ended questions was the following:

*Welches politische Thema hat Ihre Wahlentscheidung bei der heutigen Landtagswahl am stärksten beeinflusst? (engl. Which political issue most*

> *influenced your voting decision in today's state election? Translation P.S.)*
>
> *Falls Sie heute mit Ihrer Zweitstimme anders als bei der Landtagswahl 2013 gewählt haben: Können Sie uns bitte den wichtigsten Grund dafür nennen? (engl. If you voted differently today with your second vote than in the 2013 state election: Can you please tell us the most important reason for this? Translation P.S.)*

Both questions reflect issues that voters deem essential for making their decision. In its extensive definition, a political issue can be anything that concerns voters. It can be as broad as environmental politics and very specific, like when you can mow your lawn at the weekends in certain constituencies. If it is high enough on the political agenda and important to voters, the question of how society should handle something becomes a political issue. (Schoen and Weins 2014, p. 285)

The labels in this task are binary. Either a person is a swing voter or not.[1] I coded this variable depending on whether a voter cast his second vote in the Bavarian election of 2013 and 2018 for the same party. Just as in Germany's general elections, the Bavarian voting system lets its voters cast two votes. However, contrary to the federal level, both - second and first vote - are equally important for the final distribution of seats in parliament. (Bayerischer Landtag 2021) Three reasons made me focus on second votes anyway. First, we did not ask for respondents first vote in our study. Second, half of the voting population does not know the difference between the second and first vote; this percentage has been stable for many years (Westle and Tausendpfund 2019, p. 22) . Therefore, it is very likely that most voters are note aware of the intricacies of the Bavarian state voting system either. They will map the knowledge they have of the German system to the state level. Third, the second vote is by definition the party vote, while the first vote is cast

[1]Why did I opt for a binary problem here since labels in the cooking data have 13 different classes? The reason is that generating all possible combinations of switch voting would lead to a large number of labels. For instance, a person who voted CSU in 2013 has Greens, Social Democrats, Free Voters, Alternative for Deutschland, and Free Democrats to choose from if they intend to switch away from CSU. All in all, this leaves us with 30 labels (six parties make 6*5 combinations). Of course, I could concentrate on some exciting switch voting patterns, from CSU to Greens, for examples; and choose 13 patterns that will be my labels. Then both tasks - voting and cooking - would have the same number of labels. That, however, would eradicate a large number of the switch voters from the 2211 samples, something that I can not reasonably do as my data set is small enough already. Later, we will see that comparing binary and multi-label classification tasks is an interesting setting and can deliver illuminating insights.

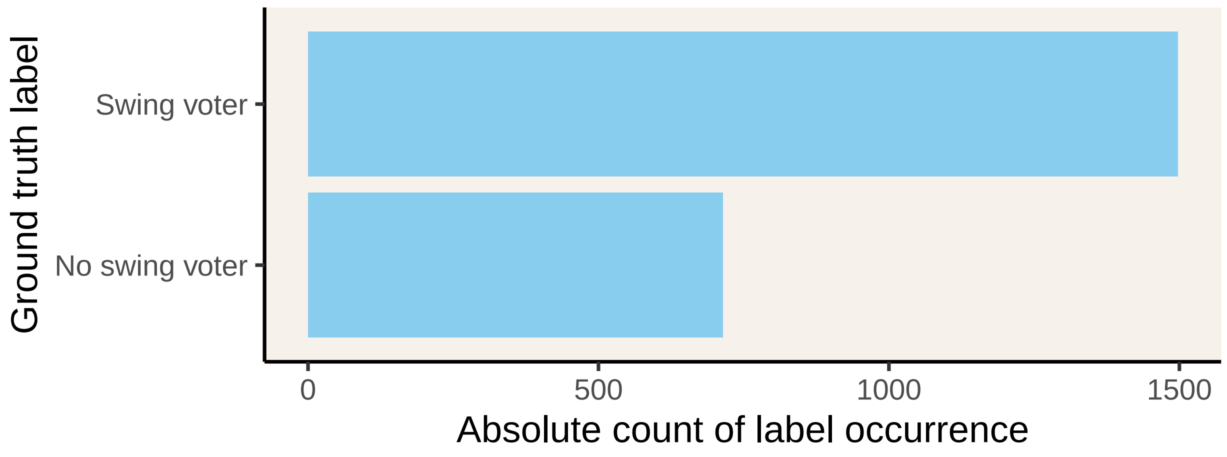


**Figure 2.2:** Distribution of ground truth labels - voting data

for a specific candidate. Therefore, we can justifiably assume that the difference in the second vote is a good proxy for whether a voter switched parties or not. As shown in figure 2.2, almost 1500 survey respondents are swing voters while only about 700 are not swing voters. Therefore, the dataset also has unbalanced labels. Still, this is nowhere near as dramatic as in the cooking data. It should also not be a problem for the classifiers since both groups have sufficient samples.

## 2.2 Theoretical considerations and explanation of additional features

### 2.2.1 Cooking data[2]

Human dialogue is dynamic and non-sequential. Understanding it as a succession of exchanged utterances does resemble its complexity. (Malchanau 2019, p. ii) One way to make this complexity accessible for ML algorithms would be to pass the whole dialogue to a model and tell it which part of the dialogue you want to have classified *while still considering the whole dialogue.* However, many ML models work on a sample basis and can not digest a whole dialogue to do some classification. Therefore, single utterances have to be put into the context of the dialogue rather than the other way around. We annotated single samples with information on dialogue context. How we did this will be explained now.

[2]I took this subsection from a term paper that I wrote during a machine learning course at the University of Regensburg. Course participants did the data coding as part of the final project.

This section provides an overview of the schema based on which we annotated additional features to the cooking data. The taxonomy we used was first proposed by Wandke (2005), but the specific form applied here was derived from Henkel (2007); there, it was employed in the context of human-computer interaction. The author states that "this taxonomy, consisting of five dimensions, can be used to describe a wide variety of assistance systems." (Henkel 2007, p. 10, translation P.S.) This is quite precisely what our coding aims to do. We want to annotate user requests to the system, that is, queries that participants directed at the experimenter as described in section 2.1. We identify at which stage of the taxonomy the system currently is when it receives a particular request. This, in turn, helps to classify user information needs correctly. A speech recognition system in the context of cooking might then be better able to distil from the utterance the information it needs to respond appropriately. This distillation, meaning categorization into one of the 13 categories mentioned above, is what we aim at. Wandkes' taxonomy has four dimensions of which we only use one, the first. This first dimension has six phases on which every utterance of experiment subjects was coded. Those phases are:

- Motive and goal formation
- Information intake
- Information analysis and integration
- Selection of an action
- Execution of an action
- Effects control.

The first phase, *motive and goal formation*, is present if a subject's utterance aims at receiving a justification or reason for an action. Looking for a specific recipe also falls into this category. Two examples here are "But then why do I only have to peel at the bottom?" or "Ok uh, I would like to cook something and do not know what. What can I cook?"[3]

Phase two - *information intake* - applies when an utterance states a question that aims at clarifying the state of the system or the state of the environment. "Can

[3]All utterances were in the German language. I translated them here to make the text more readable.

you repeat that? I didn't get that." or "Because I still have some tomato sauce, I just don't know if there are pieces in it." are two examples here.

The third phase of our taxonomy is called *information integration.* We code an utterance as such whenever a subject has understood a response given by the system but can not integrate it into the larger context of the situation (being the task to cook a specific recipe). As two examples from our data, we can give "How much Curcuma?" or "What is Cilantro?".

We call phase four *selection of an action.* We annotate an utterance this way if it concerns recipe selection and whenever a subject asks which option to choose from two or more possibilities. In our data, "Um, I think I'll have chickpea-lentil stew?" and "Do I do this in a pot or with a kettle?" are both instances of this phase.

Our taxonomy's penultimate category, called *execution of an action*, is pretty much the logical consequence from phase four. When an action is selected, but a subject still does not quite understand how to carry it out, we code it as *execution of an action.* Two examples are "What does cooking mean?" and "How does that work?".

The last phase is called *effects control* and is used by subjects to assure themselves, whether the execution of an action was correct. Often effects control occurs together with the question about the next step. Here we can bring forward "I cut the onions, what do I do now?" as an experiment participant's exemplary statement.

One last codable dimension that is not part of Wandke's taxonomy is *reference.* It can be coded as either *plan* or *activity.* The former refers to the procedure of the recipe/recipe selection. The latter is applied if an utterance concerns concrete recipe steps and the execution of the recipe. A *plan* example would be "Um What is the first step of the recipe?" An example of *activity* is "Does it say how many uh teaspoons of vegetable broth you [need]?"

We hoped that the effort of coding all our data in this way would enrich it enough to improve categorization substantially. Especially information on human dialogue flow helps classification. Table 2.4 below shows some example utterances. It also shows some columns we coded. Coding language was German. Some words in the table are self-explaining. *Fertig* means *done* and *offen* means *open* in englisch. When a phase has passed and the subject has proceeded, it is coded

**Table 2.4:** Three rows of Wandke annotations

| utterance | reference | motive | i_intake | i_integr | select_act | exec_act | eff_cont |
|---|---|---|---|---|---|---|---|
| Welche Zutaten drin sind. | Plan | fertig | in | offen | offen | offen | offen |
| Mach weiter. | Plan | fertig | in | offen | offen | offen | offen |
| Bei welcher Plattform? | Plan | in | offen | offen | offen | offen | offen |

as *done*. The system's current phase is coded as *in*, and everything that lies ahead always receives the tag *open*.

Going into the details of the coding process is neither needed here nor within the scope of this thesis. The interested reader can refer to me if coding schemes should be made available.

In addition to those seen in table 2.4 there are seven more annotated features. I will also include them in some experiments. They are

- previous information need,
- penultimate information need,
- need type,
- position of utterance in test person's dialogue,
- step number,
- step component number,
- recipe number.[4]

The first two features are the two preceding labels of the current label to be classified. That might raise some eyebrows since this essentially gives the classifiers much information on the current ground truth label.[5] However, since we are faced with a dialogue task here, I wanted to model a 'perfect world', where I can capture the cognitive task of engaging in a human conversation as best as possible. This is known in the literature on dialogue state tracking, where lagged labels are often

[4]Annotations for these features were done by Alexander Frummet.

[5]To some extent, every annotation of data in a classification task gives information on the ground truth. That is what annotating data then is about, helping the classifier by providing additional information.

employed to predict a system's current state.[6] (Jurafsky and Martin 2020, p. 509) My reasoning here it that since we do not have a perfect human dialogue theory yet, I opted to make sure my non-textual features will hold helpful information for the classifier. After all, it is also possible that my additional features will worsen results by adding a lot of noise.

*Need type* discerns information need queries into fact need, competence need, and cooperation need. It is a rough presorting of the actual labels.

The *position of an utterance in a test person's dialogue* is again relatively close to the ground truth but is also a good proxy of modelling human conversation. In cooking, it is quite evident that needs at the beginning are different from those at the end of the process. In the beginning, I might wonder what I can cook at all. In the end, this is, hopefully, not a question anymore. However, in most practical applications, the length of the dialogue can not be known beforehand. Therefore, this feature will not be available in such a fine-grained manner but will be categorical in real-world applications.[7]

The last three mentions in the above bullet list, namely *step number, step component number,* and *recipe number*, all hold information on the recipe that a user is cooking. This, of course, also makes a difference in how the dialogue will be unfolding. Talking about Viennese sausages with mustard is simple. Going through a six-course menu for your wedding is a very different thing. Therefore what you want to cook (i.e. the recipe) and how far you have come with it already is very important for the classifier to know.

So much on the theoretical ideas behind and operationalizations of the annotated features of the cooking data. The textual voting data have not been annotated but rather come with additional features that were empirically surveyed in tandem with the open-ended questions.

[6]Of course, we do not want to predict the systems current state but rather the users information need that the system may want to address.

[7]What is still thinkable is when you have a huge amount of dialogue, you could calculate a numerical state tracker feature. This can be done by estimating for a user query how far the query is away from the average length of all dialogues. Such a value can then be above or below the mean and would be a fine-grained global measure of how long the speech assistance process has been going.

### 2.2.2 Voting data

There are many reasons why a person might change his or her mind when choosing which party to vote for. Things like party identification have proven a strong predictor of individual voting behaviour over the last century. (Campbell et al. 1960) Nonetheless, it has been in decline for several decades. (Schoen 2014, p. 505) Other factors like the candidates and specific, prevalent issues that change with every election have considerably increased their explanatory power when we want to explain how people vote. (Pappi 2003)

When predicting whether people will stay with the party they voted for the last time, such variables (i.e. features) have usually been measured using surveys. When faced with a classification task, social science and political science have long resorted to logistic regression or the support vector machine (SVM). This might have several reasons, one of which I think is that linear regression today is still the uncontested workhorse of social science statistics. Any variety of it has an easier time to become an accepted method within this discipline.[8] That does not mean that SVMs and others are not potent methods or that political science is methodologically poor. Scientists made many great discoveries using regression.

It is nonetheless true that surveys usually have questions that we can not fully harness by these methods, such as open-ended ones. However, these open-ended questions often contain the most information on issues that voters are concerned about. Moreover, if we believe that issue voting is on the rise, these questions and the information they contain should be exploited more. Only very recently have textual survey data and modern NLP methods been systematicly discovered for voter classification. (Meidinger and Aßenmacher 2021) The focus here was for a long time, and still is, data from social media.

I will use the survey data which I already described more closely in section 2.1.2. Some variables available in the data set are demographic. I include the following additional features to improve the classification: *age, education,* and *place of residence.* One more variable that is not demographic in nature is whether a voter voted his or her way because of a *candidate or a party.* As a last amender to

[8]Just as linear regression SVMs fit a line to a cloud of data points. Linear discriminant analysis - also often used in social science - is a variety of linear regression. Its goal is comparable to that of an SVM. (Backhaus et al. 2018, p. 204)

the textual input, I bring in *what party people voted for in the last two elections*, which at this point were the German federal election of 2017 and the Bavarian state election of 2013.
This gives us the following as additional features:

- Age
- Education
- Place of residence[9]
- Vote for candidate or party
- Second vote at German general election 2017
- Second vote at Bavarian state election 2013

Stöss (1997, p. 56) finds that swing voters tend to be younger, and as people age, they tend to stabilize their voting behaviour. This means that the information on age could help classifiers to predict our target labels.

Research says swing voters tend to have comparatively high education. In my case, the Bavarian state election, I also include the variable since I hope this variable will help identify the large swath of people who voted for the Green Party the first time. Green voters tend to be very well educated. This is uncontested in research and literature. (Brenke and Kritikos 2020, p. 305; Stifel 2018, p. 176-177) My hope is that the education variable will carry some information on swing voters that opt for the Greens the first time.

Using the place of residence to classify voters goes as far back as to Lipset and Rokkan (1967). Terms like the Worcester woman, the Holby City woman or the Soccer Mum have been en vogue to describe voter profiles, especially in the United Kingdom since the 2000s. (Spectator 2019; Aaronovitch 2009) What strikes is that mostly these profiles describe some urban or suburban personas that policymakers are especially keen to convince with their offers.[10]

[9]Note that this is operationalized as the place where people went to cast their vote. However, since they have to vote where they have their primary residence, this is very equivalent.

[10]Of course, there are exceptions to this. The most prominent example here are presidential elections in the United States of America, the last couple of which were decided in states like Wisconsin, Illinois, or Michigan. Presidential candidates courted blue-collar workers, which do not necessarily fit into the archetypal urban voter profile.

The reasoning behind the last two variables is probably apparent. If people have a steady party allegiance at least over the two previous elections, they might have a lower chance of becoming swing voters in 2018. Therefore, these two variables in unison can help separate swing voters from non-swing voters.

One more practical reason I chose all of these variables is that they are usually included in most political science surveys that concern voting. If you can find a reasonably good classifier using these variables, you can use it on pretty much any larger survey conducted in the last decades. Furthermore, these are questions that usually take no more than five minutes to get from people. Therefore, concise surveys could also be used to amplify classifiers using social media data.[11]

We will now carry on with the research design where I quickly give an overview of the conducted experiments. I will also layout how we go forward and formulate my hypotheses.

[11]Of course, this is delicate grounds. Not only ethically but also in the eyes of the public. Classifying and predicting voter behaviour might seem an exciting scientific challenge, but from the viewpoint of democratic theory, such techniques can be harmful. On microtargeting in Germany, see for instance, Papakyriakopoulos et al. (2017).

# 3 Research design

## 3.1 Research question

I will now describe how the design of the study is going to unfold. To this end, I will first say a few words about the overarching research question and my general discovery interest. In the second subsection of this part, I will formulate five hypotheses to be investigated by experimentation in the latter half of this study. Subsection three ends this part by sketching how I carried out my experiments.

I have already outlined my research interest in the introduction, but I want to update it now as I already had the chance to give some background knowledge to the reader. Generally speaking, I am concerned with two things here. First, I want to find out how machine learning can improve dealing with human dialogue. Second, I want to compare how transformers perform on classification tasks where little data is available. A particular interest here is how they compare with each other and with baseline models like LSTM or SVM.

> *What influence do additional numerical and categorical features have in text classification tasks when using transformer models? Is there a difference between dialogue data and non-dialogue data? How do different transformer architectures compare with each other and with other ML models in such circumstances?*

This picks up on research by Vlasov et al. (2020) from a company called Rasa. Rasa provides a backend for implementing chatbots using YAML (Yaml ain't markdown language) and markdown syntax. As the ML backbone of its software, Rasa also uses a TN based architecture. Naturally, the company is primarily interested in how these models perform on dialogue data. A chatbot is also very different from the conversational agent, as it was simulated by the in-situ experiments I described in section 2.1. Comparing transformers between dialogue tasks and non-dialogue tasks as well as modelling the human-system interaction with the taxonomy from section 2.2 is a new approach. It fills the gap of a more theory-driven use of deep learning in dialogue processing.

## 3.2 Hypotheses formulation

I have already argued above (see section 2.1.1) on dialogue complexity. Of course, this also means classification differs when there is no dialogue nature of the data at all. This is the case with the voting data. Every sample is a single document and unrelated to the other samples. Following my argumentation above, I postulate that additional features will improve classification more when dialogue complexity is negligible or even non-existent.[1] This is so because we are not faced with modelling the intricacies of human conversation but rather enrich our data with more information. There is not much risk of messing up the signal that helps the classifier predicting groups. Instead, we can be reasonably confident that the information will be of help to the classifier.

> **H1**: Using additional features to text-only classification tasks improves classification better if documents are not dialogical and can be seen as single atomic units.

What I also want to test is a less specific version of **H1**. Because there is also the question of whether multimodal transformers, using additional features, can improve classification at all. This is not self-evident for reasons other than those elaborated around **H1**.[2]

> **H2**: Multimodal transformers improve classification results compared to text-only transformers.

[1]For attempts to overcome the need to model complex dialogue flows, see Mrkšić et al. (2017).

[2]There are also many technical pitfalls lurking here. One is, for instance, the question of how to bring together (i.e. concatenate) text and non-text information. For a list of possible concatenation methods in the `multimodal-transformers` library, see Georgian (2021).

As previously stated, I want to compare transformer performance within their model family. The most pressing question here is whether the first TN that was implemented in practice, namely BERT, gives the best results on my tasks; or is it rather the case that any of the other new models like Electra or ConvBERT yields the best scores? The hypothesis's direction is hard to tell since these models specifically boast about being applicable to any problem. Therefore, I leave it up to empiricism to decide on this. Also, the literature and papers published on which model is better can not be cited here, let alone summarised. A literature review on this would probably a master's thesis of its own. (for a real quick overview, see Gillioz et al. 2020) However, it is mostly the case that new models, developed later, claim to improve performance. This reasoning leads to **H3**.

> **H3**: BERT, on average, delivers different classification results than newer transformer architectures.

Then there is the almost obligatory comparison to classical baseline models. In my case, they are split into deep learning and non-deep learning, leading to **H4** and **H5**.

> **H4**: For small corpora, pre-trained transformer architectures deliver better results on pure text classification than classical non-deep learning models.
>
> **H5**: For small corpora, pre-trained transformer architectures deliver better results on pure text classification than non-attention-based deep learning models.

Those are the five hypotheses that will guide us through the remainder of this study. All of them only have the theory and data perspective so far. Therefore, I will reformulate them more precisely in section 5.4. I will have had the chance then to describe the experiments and their implementation more closely. With a background on this, more measurable and precise hypotheses can be phrased. However, I also find it essential to give some version of the hypotheses as early in the study as possible. Also, given a broader version usually promotes comprehension. Therefore, I use this two-step approach to hypotheses formulation. Of course, I do not want to make any empirical postulation that is not strictly testable with my data or study setup.

## 3.3 Train-test split

With as little as about 2500 samples in each data set, choosing train and test sets with intricate consideration is crucial. I opted against k-fold cross-validation (CV) since this would reduce the number of sampels even further. Instead, I go for stratified holdout validation, which I repeat ten times with ten randomly chosen but constant random states. At this point, ML practitioners might become alert since the changing of random states is always suspicious to them. In case of such unbalanced labels (especially in the cooking data set), this was the only thing advisable to do and can be seen as data augmentation. For instance, the cooking test set contains 115 samples for one class and only seven for another. This is so because these are also their relative proportions by which they occur in the overall data. It is now crucial for the classification's performance, whether a hard to classify example made it into the test set or a relatively easy one was chosen. Therefore, I conduct all my experiments with a fixed but randomly chosen set of random states and average the results over these ten runs or report intervals. (López 2017; Han 2019) Furthermore, what one should not do with random states, is only trying to optimize them. That not what I am trying to do. I am getting rid of the distorting effect the random state has if such few samples are available.

## 3.4 Experiments overview

In total, I conducted 3720 experiments. Their split according to the different models is shown in column six of table 3.1 below. I just want to give a quick overview here before I go on. How I implemented the experiments, what libraries I used, and so on will be the subject of the next part.

So for now, do not be bothered by how these architectures work or how exactly they were implemented. We will come to that shortly. Even the meaning of the acronyms in the first column is not essential for the moment. Note that all experiments were done with both data sources (i.e. voting and cooking data) and ten different random states as described in section 3.3.

**Table 3.1:** Overview of conducted experiments

| Architecture | Epochs | Featurization | n pre-trained models | Additional features | n experi-ments |
|---|---|---|---|---|---|
| NB | NA | Tf-idf | NA | No | 20 |
| SVM | NA | Tf-idf | NA | No | 20 |
| RF | NA | Tf-idf | NA | No | 20 |
| MLP | 40 | Tf-idf | NA | No | 20 |
| CNN | 40 | FastText | NA | No | 20 |
| LSTM | 40 | FastText | NA | No | 20 |
| BERT | [1,2,3,4,5,6] | BERT Tokenizer | 9 | Yes | 1920 |
| DistilBERT | [1,2,3,4,5,6] | BERT Tokenizer | 2 | No | 240 |
| ConvBERT | [1,2,3,4,5,6] | BERT Tokenizer | 1 | No | 120 |
| Electra | [1,2,3,4,5,6] | Electra Tokenizer | 7 | No | 840 |
| GPT2 | [1,2,3,4,5,6] | GPT2 Tokenizer | 2 | No | 240 |
| XLM | [1,2,3,4,5,6] | XLM Tokenizer | 2 | No | 240 |

The second column shows the epochs that the models were trained for. This only makes sense for the deep learning case, of course. The third column is the number of different pre-trained models I iterated over with all experiments. The number differs depending on how many pre-trained models are available online. Mostly they are from a catalogue that comes with the Python library used in this work. There, one can look up pre-trained models, more on this in section 5.2.

As shown in the fourth column, only pre-trained BERT models were available from the multimodal-transformers library at the time of this writing. (Georgian 2021) Therefore, additional features are used only with pre-trained BERT architecture models.

# 4 Climbing the BERT mountain

BERT, a so-called transformer network, is a quite recent invention in NLP. To understand how BERT in particular and a TN in general work will be the aim of this part.

In his YouTube series and book on BERT and transformers, ML teacher Chris McCormick talks about learning and understanding these new models. He says that they are usually taught to people who first came into contact with other kinds of deep neural networks, namely convolutional neural network (CNNs) and recurrent neural networks (RNNs). This, however, is not needed and might even hinder an intuitive grasp of BERT. He calls this the BERT mountain, as shown in figure 4.1. (McCormick 2020; McCormick 2019b)

Therefore, he explains BERT in a way that does not require any beforehand understanding of other networks; where you do not have to climb the BERT mountain. It is no small didactic problem if teachers have to presume or even build sufficient knowledge before coming to the actual concept of interest. Since BERT and transformers are quite different from other NLP networks, questioning the approach of the BERT mountain is appropriate. There is also a wholly different view on the BERT mountain that I find is very well summarized by the meme on the next page.
I do not think much interpretation is needed here. Understanding what came before a groundbreaking invention like BERT is most likely essential to understanding the invention itself. Both argumentations - the BERT mountain and its opposite - have some validity. Unfortunately, I do not have space here to explain the fundamentals

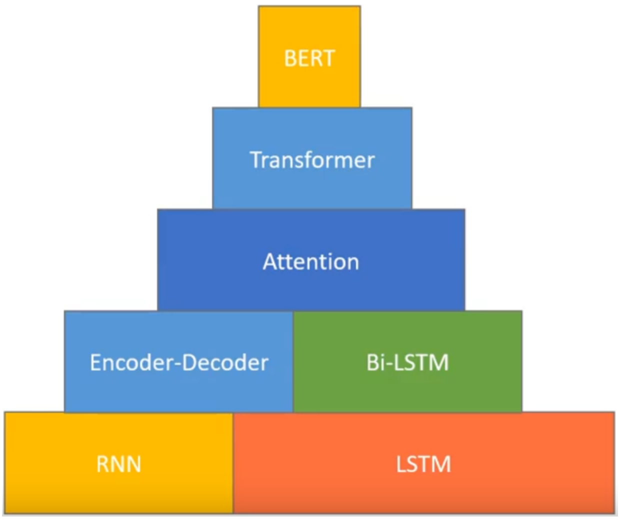


**Figure 4.1:** The BERT mountain, from McCormick (2019a).

**Figure 4.2:** Cheating your way up the BERT mountain.

of ML and deep learning. I refer the reader to some great books covering everything up to but not including transformers: Raschka and Mirjalili (2017), Aggarwal (2018), Goldberg (2017), Kedia and Rasu (2020). I will therefore skip classifiers like naive Bayes (NB), random forest (RF), support vector machine, CNN, and long short-term memory (LSTM) networks. Also not explained here are the featurization methods for textual data a will use later, namely term frequency-inverse document frequency (tf-idf) and FastText. On FastText, see Joulin et al. (2016), Bojanowski et al. (2017), and Kedia and Rasu (2020). It is a method for word vectorization based on Word2Vec. (Mikolov et al. 2013) So basically, we skip the bottom third of the BERT mountain seen in figure 4.1 and come straight to everything directly related to BERT and transformers.

## 4.1 Transformer Networks - or what BERT is

Before I carry on, I quickly want to sketch what this subsection will be about. Transformers are a relatively recent development in ML. A transformer makes use of the concept of transfer learning, which is applying a model trained on a different task and different data. There are various kinds of transfer learning, and I will introduce them in the next section.

A transformer is composed of an encoder and a decoder. The encoder enriches the input data and transforms it so that the decoder can perform a given task on the data as good as possible. For instance, the decoder could enhance our input word embeddings and the decoder would perform the actual classification. Both encoder and decoder are composed of blocks which in turn are composed of layers and sublayers. How many of these components are present in a transformer varies. For example, the base version of BERT that we will use has 12 layers. (Huggingface 2021a) I will soon describe in more detail how encoder and decoder work in transformers.

Transfer learning and encoder-decoder layouts have not been new when transformers came around. What has been quite a breakthrough is the invention of attention by Bahdanau et al. (2016). Vaswani et al. (2017) then used this concept in their paper *Attention is all you need* to propose its use in a neural network. Attention is crucial to understanding transformers, and I will go to some length explaining it in this subpart. For now, imagine it as giving the network the possibility to determine which words in a sequence belong together and form meaning together to produce a particular output. Especially bear in mind that this is theoretically possible over arbitrarily long distances, meaning it does not matter how many other words are between two words that belong together. This is an essential part of human language that RNNs and CNNs have never fully mastered. In the penultimate part of this section, we will dig deeper into attention.

In the final part of this section, I will walk through the vanilla transformer network step by step.

### 4.1.1 The concepts of transfer learning and fine-tuning - or how BERT is trained

Although the transformer and transfer learning sound quite similar, the latter has not been invented with the former. Word2Vec, GloVe, FastText and other word embeddings already do transfer learning. They train on one task and domain and try to transfer the knowledge embedded in their vectors into other domains and tasks. However, these embeddings are context-free. For instance, the word *bank* could be the place where you deposit your money, or it could be the thing you sit on in the park.

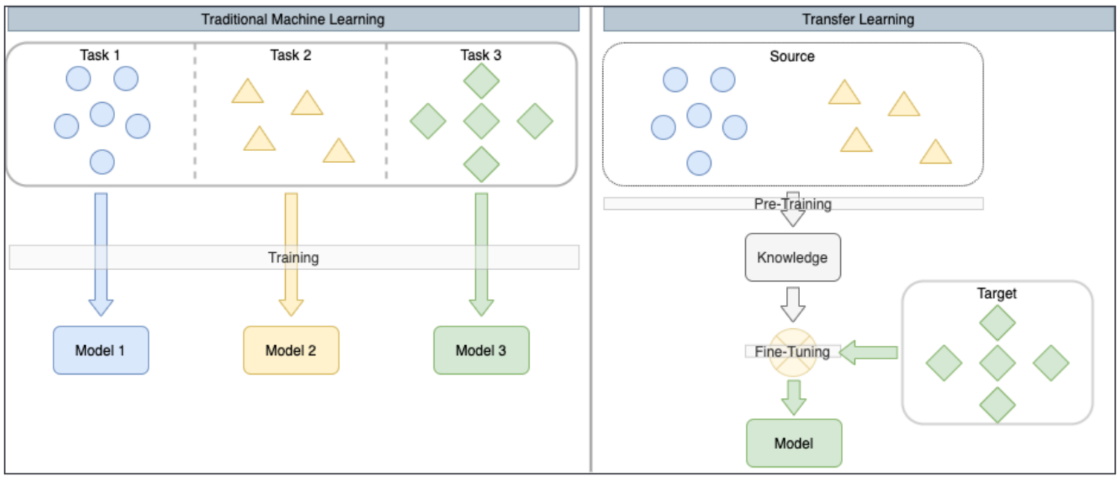


**Figure 4.3:** Illustration of transfer learning, from Bansal (2021, p. 106).

Bansal (2021) splits transfer learning into three main categories, two of which I deem particularly relevant to understanding transformers. There is *domain adaptation, multi-task learning*, and *sequential learning*. What he generally understands by transfer learning can be seen in figure 4.3. While on the left side, we see traditional ML models being built anew on every task and every data set, the right side shows what happens if we incorporate transfer learning. We have a bigger data set that we pre-train a model on. We then use that pre-trained model and the knowledge incorporated within it to fine-tune the model for a new task and with new data.

The first, *domain adaption* is pretty much what machine learners do all the time. They gather new data for their task. If you train a model by splitting up all your data into train and test data and it performs well on the test data, that is a good starting point. However, this test set still comes from the *same* data set. The ultimate test would be to gather new data in different circumstances, for instance, take new pictures of cats in different surroundings if the task were to spot cats in pictures. *Domain adaption* will not be part of this research project. I do not have the time or resources to gather new data from voters[1] or conduct new cooking experiments.

The second kind of transfer learning, *multi-task learning*, is what BERT does. If you wish to train a model that will perform reasonably well on a whole range of tasks, you could train the model on all of these tasks. That would not leave you with one but many different models. What was done instead when building BERT was that BERT was trained on two specific tasks. Both of them do resemble practical applications to some degree, but they are not what you would come up with when trying to get the best model for a specific task.

The first task BERT was given - *called masked language modelling (MLM)* - goes like this: In every input sentence, a fraction of all words get masked, meaning BERT does not know anymore which word it is receiving as input. Another fraction gets randomly replaced by a random token from the vocabulary and the remainder of input words stays untouched. The aim is to train BERT to correctly learn the masked and swapped tokens to output the correct original sentence, although the input has been severely mutilated. (Kedia and Rasu 2020, p. 277)

The second task - *next sentence prediction (NSP)* - feeds BERT with pairs of text passages. Fifty per cent of the time, the latter passage truly belongs to the former. In other examples, the second text passage is just a randomly selected text from the whole data set. The aim of BERT here is straightforward. Do these two text passages actually belong to each other? Should the second follow the first? (Kedia and Rasu 2020, p. 278)

As can be seen in figure 4.4, in practice, BERT has to solve both tasks simultaneously. The combination of both is something that I think is quite hard to find a real-world application for. Maybe some intelligence service could be faced with very distorted documents that might resemble this task that BERT

[1]What I could do is taking the gargantuan amount of political survey data gathered in the past 100 years or so. Many of them ask the questions we were asking in our survey, and many of them include the demographic variables I use for my models in the latter part of this thesis. Bringing all this data together would still be a massive project in and of itself. One recent paper by Meidinger and Aßenmacher (2021) does something like this with survey data from the United States. They show how open-ended survey questions and pre-trained transformers can help to learn about peoples political attitudes.

was given. Or in the domain of historical documents, I could imagine finding an application like this. What I want to say is that both the tasks and certainly their combination is not an everyday application of ML and it remains elusive *why* training BERT this way was the best choice. Devlin et al. (2019) chose the tasks and their hyperparameter (e.g. the percentage of words to mask) by experimentation. They chose it in a way that the resulting model would, on average, yield the best results on a whole lot of very different tasks.

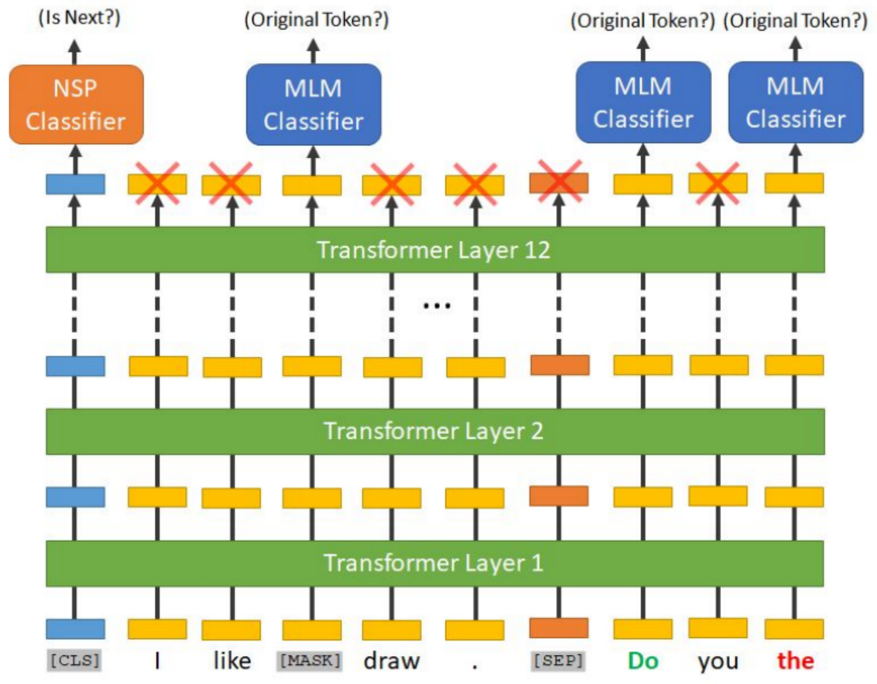


**Figure 4.4:** Cuncurrent training of BERT on MLM and NSP tasks, from McCormick (2020, p. 76).

While learning on different tasks is called *multi-task learning*, transferring a learned model to another task is called *sequential learning*. You first train your model in a stage called pre-training and then apply it to another task that could even come from another domain. So, while pre-training in BERT happens using *multi-task learning*, applying a pre-trained BERT to your specific problem is *sequential learning*. It is essential to distinguish here to know what we are talking about and make sure we all mean the same processes and concepts. I was doing a lot of *sequential learning* for this thesis and next to no *multi-task learning* since this was done for me in a pre-training process by other institutions with enough resources.

So, applying a pre-trained transformer is not just letting is loose as is on your data. There is the possibility to fine-tune it. If you have very little data, like less then 3000 samples in my case, BERT will still be pre-trained on the tasks as mentioned earlier on gigabytes and gigabytes of data for you. Fine-tuning is the process of training a transformer model again on new, mostly much fewer data. Most models I will discuss later have been trained on huge corpora whose size goes into the gigabytes or terabytes. Weights and attention scores update just as in training from scratch. The difference is, we do not start with random initialization but from a checkpoint that is stored for us by the company or team that pre-trained the models.

There are three obvious upsides about using fine-tuning:

1. Quicker Development
2. Less data required
3. Better results (McCormick 2020, pp. 16-17)

Pre-training is usually done for dozens of epochs and takes very long. (Devlin et al. 2019) With much fewer resources than a big institution like Google, the applied researcher can then take the model and train it again for a few epochs (usually two to four) on her small data set. This process is called fine-tuning and it sounds really commonplace, almost obvious, to do it that way. Before the advent of transformers, it was not since models like LSTM or others never came up with one single architecture or training task or training data that would perform as well as BERT does on so many tasks.

### 4.1.2 Encoder-decoder architectures - or how BERT (partly) looks like

Like transfer learning, the idea of encoders and decoders was not new to ML. (Cho et al. 2014) Still, this concept is vital for understanding transformers.

Before saying anything about what an encoder or a decoder is: BERT, as one particular case of a transformer network, *only* consists of an encoder. Remember what the acronym stands for: *Bidirectional Encoder Representations from Transformers.* It becomes clear that BERT's output is only what comes out of the encoder part of what Vaswani et al. (2017) proposed. These encoder representations are essentially word embeddings enriched by the encoder, which is BERT.

Figure 4.5 is a very simplified sketch of an encoder-decoder architecture. The encoder gets an input, does something with it, passes it on the decoder, which again does something with it and produces an output. In a sense, a classical NLP workflow that uses traditional ML methods also resemble this architecture. (Jabbar 2020) Tf-idf, then works as the encoder by producing something that the decoder, for instance, an NB classifier, can work with. So encoding and decoding has been part of NLP machine learning even before deep learning took hold of the field.

What is so entirely different about transformers is their general idea of how encoder and decoder should look like and that you can put their various variants together like building blocks. They need comparably little changing or hand-tailoring of features. We will look under the hub later and see what is inside the encoder and decoder of a transformer network.

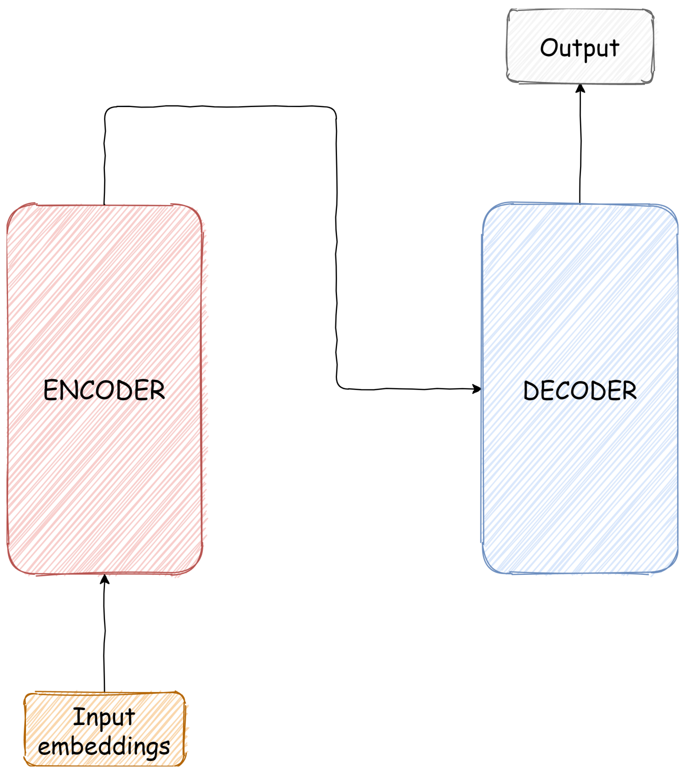


**Figure 4.5:** Sketch of encoder and decoder in a transformer, adopted from Vaswani et al. (2017).

### 4.1.3 Self-attention - or how BERT learns

Bert notices the hat, but chooses not to engage Ernie since he knows it'll lead to some ridiculous stuff.

**Figure 4.6:** Bert not paying attention

Just as in Sesamstreet Bert in the meme above, BERT (and every other transformer) can pay attention or refrain from doing so. Intuitively said, self-attention aims to measure how important words are for each other to form meaning for a certain output. This output can be another word in a translation task or a class label's likelihood in a classification task.

> "Do not be fooled by me throwing around the word 'self-attention' like it's a concept everyone should be familiar with. I had personally never come across the concept until reading the Attention is All You Need paper." (Alammar 2018)

I appreciated these lines when I first came into contact with the concept of self-attention. It is not something that comes easily to mind, especially when you come from CNNs and LSTMs and other deep neural networks.

It is probably easiest to start thinking about self-attention on a word-to-word basis. The aim is to find out how related a single word is to every other word in a sequence, including the word itself. All of this is usually done with dot products between word embeddings.[2] Self-Attention can capture whether a single word like a pronoun belongs to a noun or vice versa. (See 2019)

[2]Again, I can not go into the mathematics here. I refer the interested reader to Rothman (2021, pp. 20-31), where the author illustrates all the necessary calculations in Python using only NumPy.

According to Hagiwara (2021, p. 192), self-attention “is a mechanism for creating a summary of the input under a specific context.” In the present case, this summary is the likelihoods of the 13 and two labels in the cooking and voting task, respectively. Put differently, the input data is reflecting on itself to produce the desired output. With an example from the Bavarian state election data of 2018, I want to elaborate on this further.

> *Weil ich gehofft hätte mit dem ÖDP*[3] Kreuzchen Fr. Merkel nicht mehr zu unterstützen! **CSU** ist nach wie vor meine **erste Wahl** wegen der Stabilität für die **sie** steht.[4] (engl. Because I had hoped that by voting ÖDP I would no longer support Mrs Merkel! **CSU** is still my **first choice** because of the stability **it** stands for. My emphasis, translation P.S.)

Here we have a triplet of a noun (*CSU*), a closer description of its properties to the voter (*first choice*), as well as a pronoun referring back to the CSU (*it*). For a human reader, it is perfectly sensible to interpret the pronoun at the end of the sentence as belonging to the CSU. However, this is far from obvious for a computer or neural network, especially in tri-gram situations like this. After all, the word *sie* (engl. *it*) could also refer to *Stabilität* (engl. *stability*) or *Wahl* (engl. *choice*). Humans have domain knowledge, or I should rather say common sense knowledge of what the respondent here might be trying to get at. You can imagine how hard this will become when a knowledgeless and ignorant system tries to infer meaningful n-grams with n larger than three. Indeed, in most real-world problems, the meaning of sub-sequences of a more extended sequence usually is not fully expressed in a collection of bi-grams. However, one single self-attention mechanism can only capture the relatedness of word bi-grams. This is the reason - as we will soon see - why transformers always have more than one attention layer. I will go into this in the following section.

One especially instructive visualization from Google that I do not want to withhold here brings together everything I have said so far about encoders and

[3] *The CSU is the Christian Social Union. The party has governed Bavaria since 1957 and won the elections in 2018. The ÖDP is the Ecological Democratic Party, a small party acting mainly in Bavaria and not represented in parliament.*

[4] The sentence resembles a so-called Winograd schema. (Winograd 1972) A typical example is this: *The city council refused the demonstrators a permit because they feared violence.* (Mitchell 2019) How is fearing violence here? This resembles the question from above of who stands for *stability*, although the above example should be easier to solve for an ML model.

decoders as well as self-attention. It is an animation and can be found in this post on the Google AI blog:
`https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html`

### 4.1.4 Transformers and BERT in seven steps

Now that we understand the most fundamental concepts surrounding a transformer, I will walk through the transformer architecture seen in figure 4.7.[5] I will do so in seven steps that are annotated to the figure ($\boxed{1}$). We have already seen the red and blue blocks (encoder and decoder) of the new figure on the next page in section 4.1.2 of this work. I now open the bonnet to see what is inside. Before we go on, one little remark on terminology here. When I say layer, I mean the red and blue blocks, namely encoder and decoder. The vanilla transformer from Vaswani et al. (2017) had six layers of each. BERT from Devlin et al. (2019) has 12 encoder layers and no decoder layers. When I speak about sub-layers, I mean what is inside these blocks, namely multi-head attention, masked multi-head attention and feedforward (in yellow and orange). I will refer to the embedding layers at the very bottom of figure 4.7 as sub-layers, although they are not contained within the red and blue blocks. However, the literature uses this terminology and I follow it.

#### 1. Encoder embedding layer and positional encoding

"The input embedding sub-layer converts the input tokens to vectors of dimension d model = 512 using learned embeddings in the original Transformer model. The structure of the input embedding is classical:" (Rothman 2021, p. 8) And they also work quite classical. Original BERT only has about 30,000 words in its embedded vocabulary. Words that are not contained are modelled by n-grams that can go down to the level of single letters. (McCormick 2020, p. 20-21) The embedding

[5]Note that for the sake of clarity, there are several omissions in my figure that other authors usually display. The most important of these is the lack of normalization layers. These are usually stacked after every sub-layer and - as their telling name suggests - do some normalization on the sub-outputs. As they are not crucial to understanding the general idea and flow of transformers, I opted not to go into them in the following pages. If you wish to read up on them, I refer the interested reader to Rothman (2021, p. 6-38). There the author explains the whole workflow in detail and also goes into what the normalization layers do.

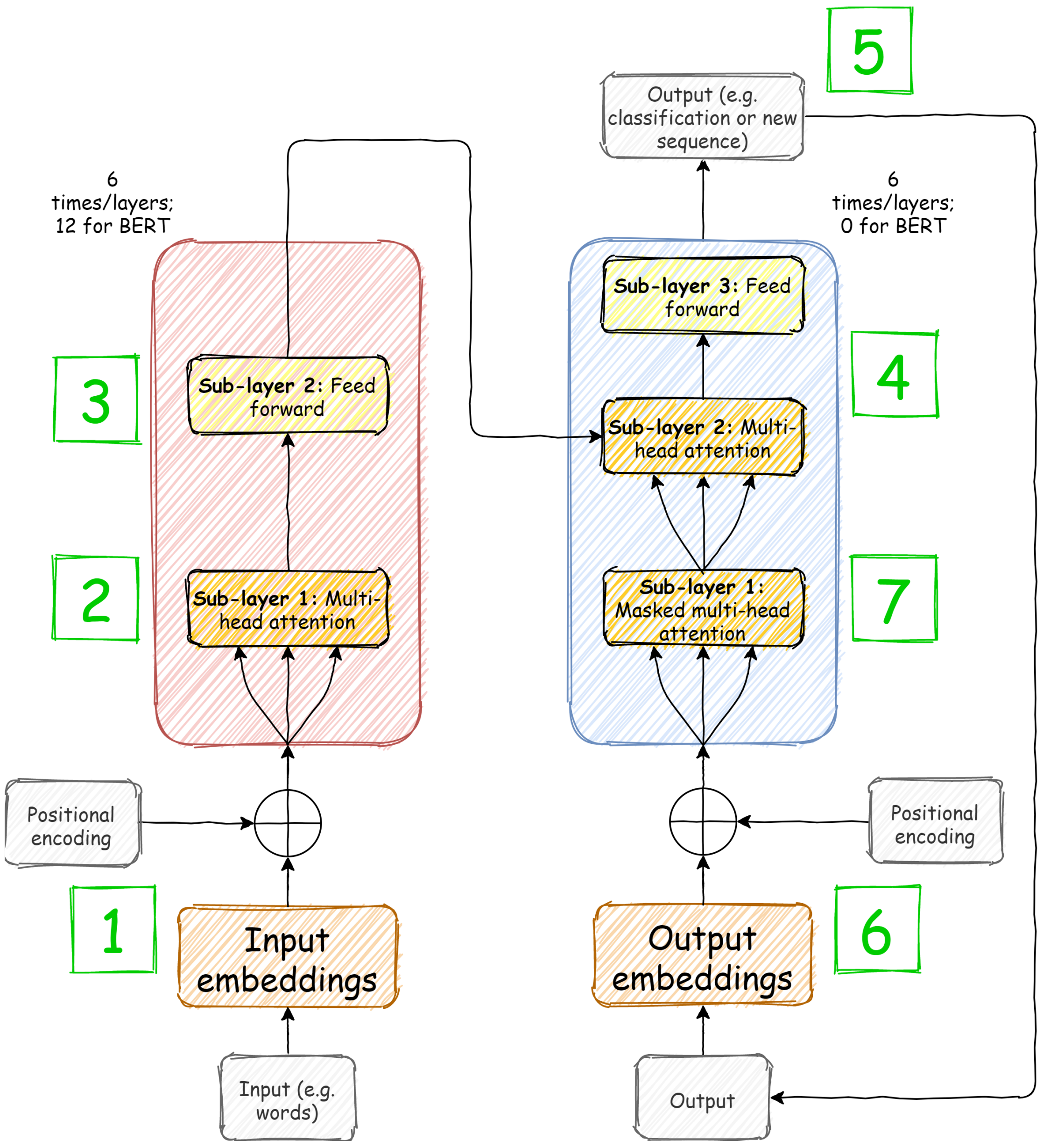

5
Output (e.g. classification or new sequence)
6 times/layers; 12 for BERT
6 times/layers; 0 for BERT
Sub-layer 3: Feed forward
4
3
Sub-layer 2: Feed forward
Sub-layer 2: Multi-head attention
2
Sub-layer 1: Multi-head attention
Sub-layer 1: Masked multi-head attention
7
Positional encoding
Positional encoding
1
Input embeddings
Output embeddings
6
Input (e.g. words)
Output

**Figure 4.7:** The transformer architecture, adopted from Vaswani et al. (2017) and Hagiwara (2021), p. 201-203.

layer produces word embedding vectors for each token (i.e. word). For every word, a 512-dimensional[6] vector is generated. (Rothman 2021, p. 8)

What is not so classical about a transformer embedding layer is that it also uses so-called positional encodings that enrich word embeddings with information on where the word is located in a sequence. It is needed since words are not

[6]This number varies depending on the specific transformer. BERT has 768 dimensions. Unless I say otherwise, in my explanations, I always talk about the general, theoretical transformer proposed by Vaswani et al. (2017).

processes sequentially but in parallel. It also enriches word embeddings further with information of which words might belong from meaning together. To illustrate what positional encoding does I adapt the following example from Rothman (2021, p. 11):

> *The **black** cat sat on the couch and the **brown** dog slept on the rug.*

In the example, the aim is to get a similarity measure for the words brown and black. Their simple word embeddings naturally have high similarity. However, they are on positions two and ten in the sequence, which makes their similarity drift apart. In a nutshell, without positional encoding, their similarity is about 0.99, while when incorporating positions, their similarity is 0.86. (Rothman 2021, p. 11-17)

I also quickly want to sketch how this works with an exemplary sentence from the voting data. I use a response one surveyed person gave to the open-ended question of why they now voted for a different party then in the last election.

> *Da ich so einen **arroganten Sack** wie **Söder** nicht wähle. (engl. Since I do not vote for such an arrogant jerk as Söder. Translation P.S.)*[7]

Now quite the opposite, as in the first example, is true. Although the respondent might see that quite differently, the terms *arrogenter Sack (engl. arrogant jerk)* and *Söder* will not have a high cosine similarity. However, their closeness in the sequence will drive their similarity measure up; this is expressed by their respective positions of five and eight.

So, in a nutshell, the embedding layer of a transformer learns (or takes pre-trained) word embeddings *and* enriches them with a positional encoding vector to reflect every word's position in the sequence.

## 2. Encoder Sub-layer 1: Multi-head attention

In a second step, the network has a multi-head attention sublayer. We are applying n self-attention calculations, called multi-head attention. Depending on how many heads (self-attention mechanisms) it has, it can run self-attention calculations for n words in parallel. Most transformers have between eight and 12 heads. (Devlin et al. 2019; Vaswani et al. 2017)

[7]Markus Söder was the CSU (Chrisitan Social Union, the major party in bavarian politics) candidate for prime minister in 2018 and now holds that office.

Multi-head attention makes use of the possibility to parallelize self-attention computations. Since every word is paired with every other, it does not matter which word we start or end with. Consequently, we are not bound to sequential computation but can do self-attention calculations for every word in parallel. That way, computational time decreases considerably compared with inherently sequential LSTM models. One objection here might be that we do not know what to do with our eight heads when the number of words in a sentence is not a power of eight. This is handled by not assigning each head one word but rather a part of each word. (Wagner 2020, chapter 8)

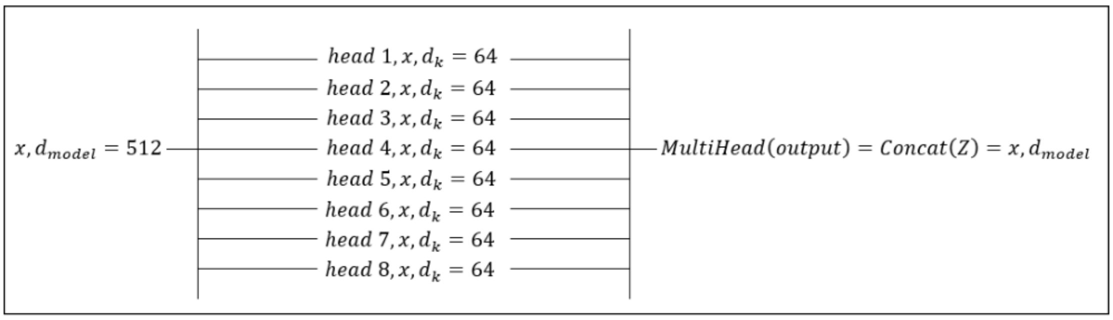


**Figure 4.8:** Multi-head attention with with eight words, from Rothman (2021, p. 18).

In figure 4.8 above the sentence has eight words with 512 dimensions each. We could let each of the heads do the calculations for one word's 512 dimensions. However, we could also divide the 512 dimensions of all words by the number of heads and let every head do the calculations on 64 dimensions of every word in the sentence. This not only solves the problem of sentence length that is not a power of eigth since it simply takes the number of words out of the equation. It also gives the heads a notion of dimensionality-competence. Say head one always does the first 64 dimensions of all words in a text and the first 64 dimensions of word embeddings always contain the emotionality of that word. Of course that is not the way it is but stay with me for the sake of the argument. If it were that way then we would end up with head one always doing the emotion-calculations of words, head two always doing the rationality-calculations of words and so on. Every head gets its competence. However, it is hard to tell wether this is what is happening inside. Vig (2019) and Vig (2021) visualise what happens when the attention-heads do their calculations.

When this process is finished, embeddings get concatenated back into their 512-dimensional form for every word. These eight vectors (derived from the number of words in the sentence), in turn, get transformed into one vector with 512 dimensions that constitute the output of the muli-head attention sub-layer. (Rothman 2021, p. 32)

Self-attention can capture - as stated above - whether a single word like a pronoun belongs to a noun or vice versa. Running multi-head attention once computes self-attention only once, meaning the network only learns how one word relates to every other word in one particular way.

However, in most real-world problems, the meaning of sub-sequences of a longer sequence usually is not fully expressed in a collection of bi-grams. In our example from the Bavarian state election of 2018, the three words *arrogant jerk Söder* belong together. However, one single self-attention mechanism for one word can only capture the relatedness word bi-grams. This is what multi-head attention does. It calculates self-attention for every word. That the process is being run in parallel for the whole sentence or text does not mean that there is more than one self-attention being calculated for every single word.

The encoder layer gets repeated n times where the input to every next layer is the previous layer's output. This way, every layer learns different aspects of how a word relates to other words.

Figure 4.9 below visualizes self-attention for layers four and 12 of a BERT model using the example sentence concerning Markus Söder. As we can see, layer four learns the connections of *Sack (engl. jerk)* with the words *einen (Engl. an)* and *arrogant*. Layer 12[8] instead learns the meaning of Sack together with the words that layer four already picked up as well as *wie (engl. as)* and *Söder*.[9]

What is noteworthy here are the ## suffixed to some of the words. That is BERTs way of dealing with out of vocabulary (OOV) words. With 30,000 terms, BERTs vocabulary is relatively small. However, when encountering unknown words, the model splits it in n-grams using the WordPiece algorithm first outlined by Schuster and Nakajima (2012). (see also McCormick 2020, p. 20-21; Huggingface 2021b)

So up to the second of our seven steps, we ran parallelized multi-head self-attention on every input word of the sequence.

[8]Remember, BERT has 12 encoder layers and no decoder layers. Therefore, all we see in figure 4.9 is self-attention produced by encoder layers.

[9]Note that learning here is not necessarily cumulative, which means that subsequent layers do not have to include n-grams already learned by previous layers. It is just the way it is in this particular example.

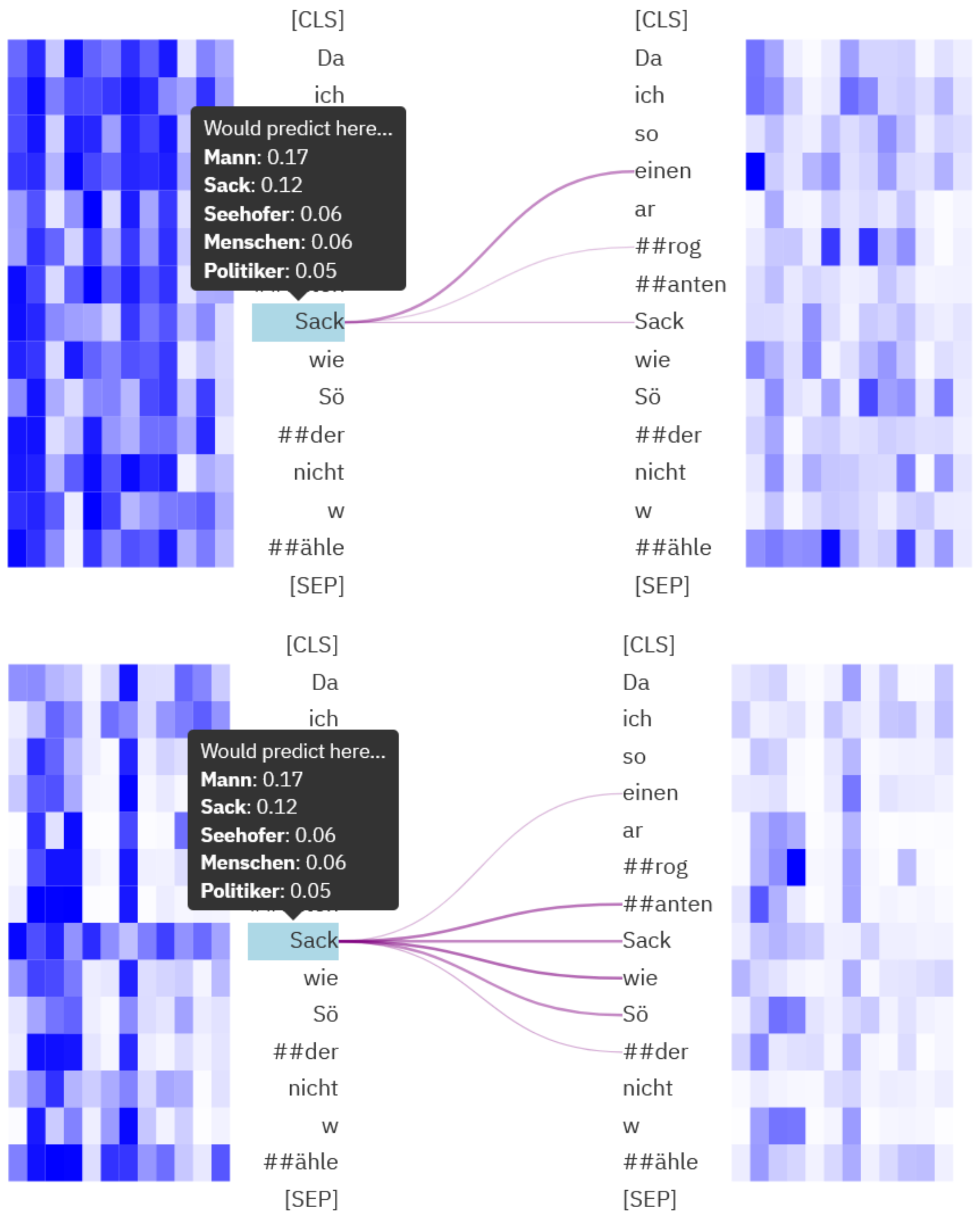


**Figure 4.9:** Layer 4 (top) and 12 (bottom) from a pre-trained german BERT model, visualized using the exBERT software by Hoover et al. (2020).

### 3. Encoder Sub-layer 2: Feedforward network

Now comes a feed-forward neural network (FFN) as a sub-layer within the encoder layer.[10] There are some traditional elements in the transformer architecture that are

[10]Here, the literature is not always a hundred per cent clear. Some authors equate an FFN with an MLP; others do not. In this work, with FFN, I mean a fully connected feed-forward neural

just as crucial as self-attention. The FFN sub-layer in the traditional transformer proposed by Vaswani et al. (2017) is two layers deep and uses ReLU as its activation function.

Steps one, two, and three are what makes up the encoder layer. They encode the input word embeddings and enrich them with additional information. The remaining steps are part of the transformer's decoder side and - in our case - classify[11] the input embeddings into categories. As BERT only is an encoder, BERT ends here. The hugginface `transformers` library that we will use later also holds classes like `BertForSequenceClassification` which implement a decoder on top of BERT.

### Steps 4 to 7 - the decoder layer

What happens in steps four, five, six and seven is better explained in unison than in discrete steps. The fourth step handles sublayer two and sublayer three of the decoder. These are multi-head attention and feed-forward sublayers, respectively and their functioning here is equivalent to what they do in the network's encoder block. The encoder's output (at step three) is fed right into the middle of the decoder, where the decoder produces some output. This output now is *fed back to the decoder embedding layer, which passes it on to sublayer 1 of the decoder.* This sublayer is a so-called masked multi-head attention sublayer (step seven). The vanilla transformer and other models like GPT2 now model a left-to-right reading-like situation. This means the output is fed back to the decoder sequentially and the decoder does not 'know' what word comes to the right (or left depending on the language) of the current word.[12] The reasoning was that humans also process language this way when they read or listen to other people speaking. (Rothman 2021, p. 173-175)

BERT did away with this and that is why BERT is only an encoder.[13] (Bansal 2021, p. 173) I interpret this so that the argument around BERT was that humans do not process sentences in a perfect real-time manner, at least not consciously. Ideally, we wait until our counterpart is finished speaking until we make a final sense of what they have said. Equivalently we do not pause after every word

network.

[11]We can change the decoder to do other things, like produce a new sequence of words. In most use-cases, the decoder's needed changes are minor to apply a transformer to a different task.

[12]Note that this is a sequential process, but multi-headed attention calculation within each block stays parallel.

[13]BERT modifies this sort of training task as already described in this thesis, see section 4.1.1.

when reading a sentence to ponder what the sentence might mean up to here. Instead, we read the whole sentence and infer its meaning when done reading. That does not mean that we do not guess or know beforehand how a piece of language might continue. Generally, we use the whole sentence to understand what it means. (Rothman 2021, p.43)

I think BERT has a case here when reflecting on human language processing in the most naturalistic way. No one could know beforehand that making the process more human-like would improve its performance. However, the whole idea of neural networks has been invented from a biological blueprint. Therefore, it is reasonable to try to carry on in modelling with this blueprint in mind.

The decoder layer again loops six times (i.e. has six layers) in the vanilla transformer. The output generated after the last loop is not fed back into the network anymore but is retained as the final result of whatever the task was (e.g. classification or a new sequence in a translation task).

*We are there at last, we have climbed the BERT mountain.* In the next part, I will describe how I implemented what you have read during your ascent to the BERT mountain. Afterwards, we will do statistical analyses of the cooking and voting task results and see what they mean for the five hypotheses.

# 5 Preparing the experiments

With the theoretical knowledge and background from parts two, three, and four in mind, I will now describe how I implemented my experiments. I have already formulated some hypotheses in section 3.2. To make them strictly empirically testable, I will also refine them in this part since their current wording is still very broad. I will start with an overview of the baseline experiments.

I made 120 experiments for the baseline models, 840 for the multimodal transformers and 2760 for the normal transformers. The number of experiments was driven by several considerations. First of all, my focus is transformers and how they compete with each other. Second, baseline experiments were rather costly to run in terms of computational time; for instance, there is no implementation for RF models on GPUs. So baselines ran on my laptop's CPU, which made them quite slow. Third, I 'only' had four GPUs at my disposal on which I ran the text-only and multimodal transformer models in parallel.

For that I used a service provided by the University of Regensburg. I had four GeForce RTX 2080 Ti, a GPU currently benchmarked among the top 20. (videocardbenchmark.net 2021) It took several days to run all the experiments - some fails included, I admit. I did not track running time precisely, but I can say it would have taken me a whole day to run all the experiments on the four GPUs, i.e. four days had I had only one available. Considering the GPU's price and benchmark, this is nothing to be done on a local notebook, at least not if it is not a high-end gaming notebook or something comparable. However, a days runtime in a cloud service is quite affordable since the hourly cost is usually between 50 Cents and two Euros. I want to say here that although fine-tuning is much cheaper than training from scratch, the computational resources needed are still considerable. If you already know beforehand what model you are going to need, then just fine-tuning that one model will take half an hour at the most. This is seldom the case, and usually, you will have to iterate over dozens if not hundreds of models. Therefore, this is undoubtedly a deficit of transformers in general and research is needed here.

## 5.1 Baseline models

NB, SVM, and RF were all implemented with scikit-learn. A best classifier was searched by optimizing hyperparameters for recall and precision on both tasks. As a featurization method here, tf-idf was employed. Table 5.1 shows the hyperparameters that the implementation grid searched over for SVM and RF. There was no grid search for NB, since there are no hyperparameters for this classifier. Optimization happens anew for every experiment run, i.e. for every random state and every task. Therefore, there could be different hyperparameters for every experiment since for each run the best was selected to classify the test set.

**Table 5.1:** Sweeped RF and SVM hyperparameters.

| Hyperparameter | Random forest | Support vector machine |
|---|---|---|
| Bootstrap | [True] | |
| max_depth | [80, 90, 100, 110] | |
| max_features | [2, 3] | |
| min_samples_leaf | [3, 4, 5] | |
| min_samples_split | [8, 10, 12] | |
| n_estimators | [100, 200, 300, 1000] | |
| kernel | | [rbf, linear, poly] |
| gamma | | [1e-3, 1e-4] |
| C | | [1, 10, 100, 1000] |

Note: As parameter names I use those expected by the scikit-learn library. The SVM hyperparameter gamma was only iterated over with the radial basis function (rbf) kernel.

As a featurization method for the multi-layer perceptron, I also used tf-idf. This is not the most common approach since an MLP can easily digest high dimensional word embeddings. However, I was curious to see a neural network's performance using a more straightforward featurization method. I put together the MLP in some iterations of trial and error, where I also experimented with some non-sequential topologies. After some reasonable guesswork and best efforts, the network that made the best classifications is the one that can be seen on the left in figure 5.1. Concerning the other two baseline neural networks, I based my topologies on Kim (2014) and Dickson (2019) for the CNN as well as Chollet (2018) for the LSTM network. The implemented CNN is in the middle, the LSTM on the right of figure 5.1 below. I used FastText word embeddings for the LSTM and CNN.

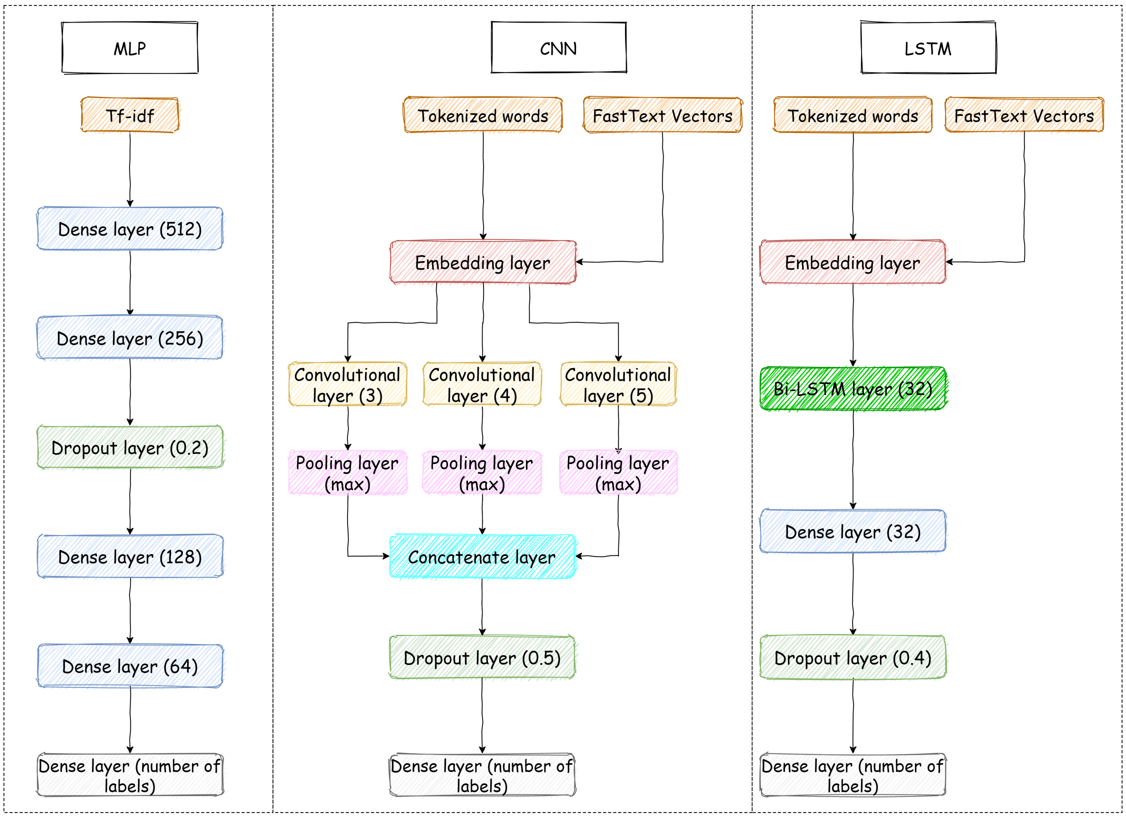


**Figure 5.1:** Implemented baseline neural networks.

I implemented the neural networks with the widely know TenforFlow API named Keras. As was already reported in section 3.4, MLP, CNN, and LSTM were trained for 40 epochs each. That led to minor overfitting, especially with the MLP model. Future experiments on the data should consider this. CNN and LSTM both have a trainable embedding layer where the FastText embedding matrix sets the initial weights.

In the CNN, we see filter layers (i.e. convolutional layers) that apply the automated feature generating process. Filter size varies from three to five. For every such layer, the network applies max pooling and concatenates the three resulting vectors together. After this, Kim (2014) puts one dropout layer as well as a dense layer for doing the final classification.

The LSTM has a bidirectional layer at its core, which means that it not only reads every sentence from left to right. Additionally, every sentence is also read in reverse from right to left to increase performance. After the bidirectional LSTM layer, there is a dense layer, a dropout layer and another dense layer to do the ultimate classification.

## 5.2 Text-only transformers

Huggingface is an NLP company and one of its main projects is the `transformers` library for Python. It holds classes by which pre-trained models provided by third parties can be loaded and fine-tuned. Say, Google happens to have a huge cooking dialogue corpus lying around somewhere. They could train a BERT from scratch, using that corpus, which usually takes days. Google could then upload this BERT cooking model to the hugging face models library, where any researcher can download it. Unfortunately, Google does not have such a corpus with cooking dialogue data. Instead, most models are trained on Wikipedia, newspapers, news craws or historical corpora, as can be seen in table 5.2. It lists all the 23 pre-trained transformers I found on the internet and the hugginface library[1] that I deem might do a decent job on the tasks. Of some models, there is a large and a base version. The large version usually has 24 layers and 16 attention heads, while the base version has 12 layers and attention heads. (Rothman 2021, p. 46) Whether a model is large or base can be read from the model name in the second column of the table. If it does neither say large nor base, the model only comes in one flavour.

For this study, I found six broader topology-classes of pre-trained transformers, namely BERT, Electra, ConvBERT, DistilBERT, GPT2, and XLM. I have already explained what BERT is. The other five are also transformers but have some variation.

[1]To look up one of the models in table 5.2, go to `https://huggingface.co/` and enter the second column's content in the search bar on the top of the website.

**Electra** is a so-called generative adversarial network (GAN). Unfortunately, we do not have time or space here to dig into its inner workings. For our purposes, it shall be sufficient that two networks are trained in a sort of competition with each other where the one tries to fool the other. Electra can also be used for sequence classification. (Clark et al. 2020; Brownlee 2019)

The authors of **ConvBERT** say the initial BERT model had some deficits concerning the attention mechanism. They say BERT too heavily concentrates on the larger self-attention picture and therefore loses sight of local dependencies. (Jiang et al. 2021) I do not know how much impact this would have with such short sequences as in the tasks here, but as there are pre-trained models available in German, it is worthwhile to try it.

**DistilBERT** is a light version of BERT that tries to retain the results while reducing computational cost. The authors aimed to make BERT cheaper to pre-train and fine-tune. (Sanh et al. 2020)

**GPT2** stands for Generative Pre-trained Transformer 2. It is a causal or sequential model and simulates a reading-like model. Therefore, being at a particular word, GPT2 does not 'know' what words come afterwards. This is different from BERT, which always processes the whole sentence at once. (Radford et al. 2018) GPT2 was designed for language generation, but just as any architecture in the transformers universe, it can be repurposed for sequence classification. Contrary to BERT - which is only an encoder block - GPT2 has only a decoder and no encoder. Its largest variant has a mindboggling 1.5 billion parameters. (Alammar 2019) I am not going to use that one, though.

**XLM** means Cross-lingual Language Model Pretraining. Those models are trained not only in German but in many different languages. There are two versions available. One pre-trains BERT-like, with masked language modelling (MLM). The other uses causal language modelling (CLM), which is the prediction of the next token. (Lample and Conneau 2019) The two versions can be distinguished in the second column of table 5.2.

I believe these models could yield good results mainly because they were trained on German corpora. Some I included for curiosity reasons. For instance, one GPT2 model was already fine-tuned on Johann Wolfgang von Goethe's Faust. There was also some reasoning behind this, of course. Faust is a tragedy and a theatre piece and, therefore, basically a long dialogue between different people. Using an already fine-tuned model to tune it again is nothing I have seen so far. However, I could not find any advice against it. I would assume that it leads to overfitting if the data you use for tuning, again and again, are very similar because that amounts to simply training a model for too long. In my case, the data could not be more different, and I only fine-tune the GPT2 model twice. The idea is, cooking data will bring in some knowledge on the domain, while Faust data already provided the model with some knowledge on the nature of the human dialogue. This is entirely up for grabs and we will see how it goes.

**Table 5.2:** Overview of all pre-trained models used for experiments.

| Model | Path in models library | Train data | Trained by |
|---|---|---|---|
| GBert | deepset/gbert-large | Wikipedia articles, Common Crawl, Open Legal Data | deepset.ai |
| GBert | deepset/gbert-base | Wikipedia articles, Common Crawl, Open Legal Data | deepset.ai |
| BERT | bert-base-german-cased | Wikipedia articles, Open Legal Data, News articles | DBMDZ |
| BERT | bert-base-multilingual-uncased | Articles of the 100 most frequent Wikipedia languages | Devlin et al. (2019) |
| BERT | dbmdz/bert-base-german-europeana-cased | Europeana Newspapers | DBMDZ |
| BERT | bert-base-german-dbmdz-cased | Wikipedia articles, EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl,News Crawl | DBMDZ |
| BERT | bert-base-german-dbmdz-uncased | Wikipedia articles, EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl,News Crawl | DBMDZ |
| BERT | dbmdz/bert-base-german-europeana-uncased | Europeana Newspapers | DBMDZ |
| BERT | redewiedergabe/bert-base-historical-german-rw-cased | Fictional and non-fictional German texts written between 1840 and 1920 | Redewiedergabe |
| GElectra | deepset/gelectra-base | Wikipedia articles, Common Crawl, Open Legal Data | Chan et al. (2020) |
| GElectra | deepset/gelectra-large | Wikipedia articles, Common Crawl, Open Legal Data | Chan et al. (2020) |
| GElectra | deepset/gelectra-base-generator | Wikipedia articles, Common Crawl, Open Legal Data | Chan et al. (2020) |
| GElectra | deepset/gelectra-large-generator | Wikipedia articles, Common Crawl, Open Legal Data | Chan et al. (2020) |
| Electra | dbmdz/electra-base-german-europeana-cased-discriminator | Europeana Newspapers | DBMDZ |
| Electra | dbmdz/electra-base-german-europeana-cased-generator | Europeana Newspapers | DBMDZ |
| Electra | german-nlp-group/electra-base-german-uncased | Wikipedia forums, Wikipedia articles Common Crawl, News Crawl, Subtitles | German NLP group |
| ConvBERT | dbmdz/convbert-base-german-europeana-cased | Europeana Newspapers | DBMDZ |
| DistilBERT | distilbert-base-german-cased | Wikipedia articles | DBMDZ |
| DistilBERT | dbmdz/distilbert-base-german-europeana-cased | Europeana Newspapers | DBMDZ |
| GPT2 | dbmdz/german-gpt2 | Wikipedia articles, EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl,News Crawl | DBMDZ |
| GPT2 | dbmdz/german-gpt2-faust | Same as row above but fine-tuned on Goethes Faust | DBMDZ |
| XLM | xlm-mlm-ende-1024 | Englisch and german Wikipedia articles | Huggingface |
| XLM | xlm-clm-ende-1024 | Englisch and german Wikipedia articles | Huggingface |

Note: For more information on the models go to https://huggingface.co/ and enter the content of the second column into the search field at the top of the webpage. DBMDZ is a team of NLP researchers at the Bavarian State Library. German NLP group and deepset.ai are companies. Redewiedergabe is a project funded by the German Research Foundation (DFG). GBERT stands for GermanBERT and GElectra for GermanElectra.

## 5.3 Multimodal transformers

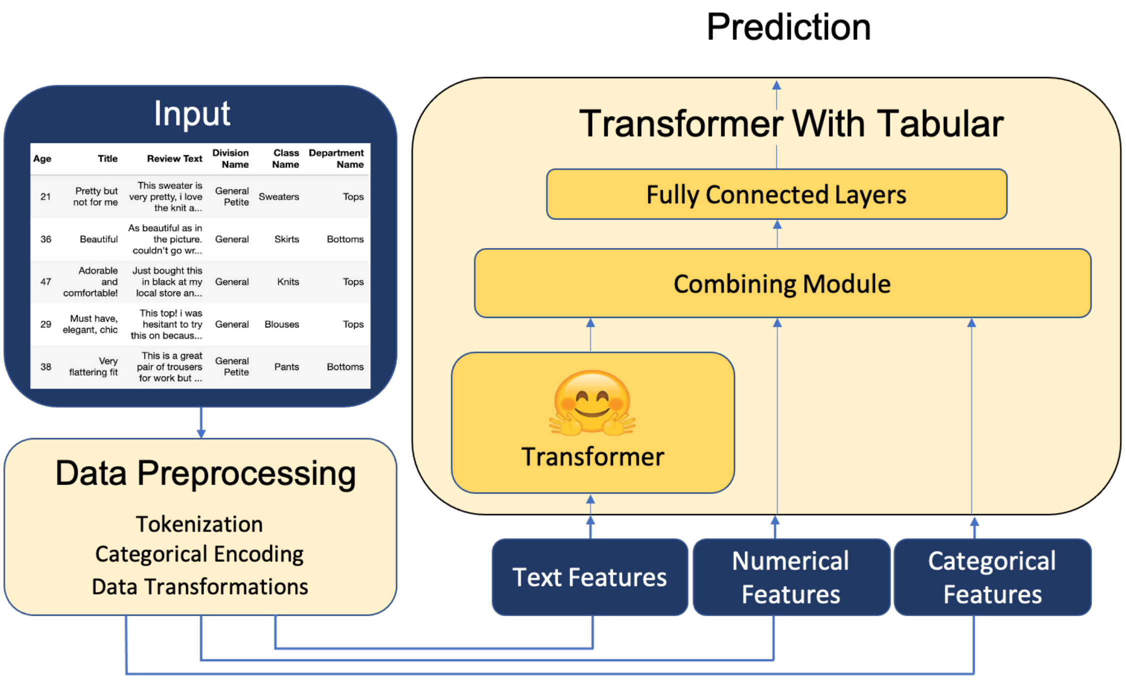
Last but not least, there are also multimodal transformers. These models enable the researcher to add additional features to her transformer models, such as numerical or categorical ones. Which ones these are, I elaborated on in section 2.2. The aim is apparent, providing additional contextual information to the NLP classifier. The library I use for this is called `multimodal-transformers`. Its developer is the company Georgian that also provides an open-source project on Github.[2]

Figure 5.2 shows the workflow one runs through when using the library. Most of the figure is already covered by implementing text-only transformers using the huggingface library. In practice, only the additional feature columns and a concatenation function have to be declared. This function tells the models how to combine textual and non-textual features. Since there are categorical and numerical additional features in the data, there is only one option in the `multimodal-transformers` library. It uses MLPs to concatenate categorical and numerical features and concatenates the result with the output from the text-only transformer. This output is fed into one final classification layer. (Gu 2020; Georgian 2021)

What should also be added here is that at the time of this writing, not all of the pre-trained models from table 5.2 work within the `multimodal-transformers` library. For the German language, only BERT topology models are available. Their corresponding names in column two of table 5.2 are the following:

- bert-base-german-cased
- bert-base-multilingual-uncased
- dbmdz/bert-base-german-europeana-cased
- bert-base-german-dbmdz-cased
- bert-base-german-dbmdz-uncased
- dbmdz/bert-base-german-europeana-uncased
- redewiedergabe/bert-base-historical-german-rw-cased

[2]`https://github.com/georgian-io/Multimodal-Toolkit`



| Age | Title | Review Text | Division Name | Class Name | Department Name |
|---|---|---|---|---|---|
| 21 | Pretty but not for me | This sweater is very pretty, i love the knit a... | General Petite | Sweaters | Tops |
| 36 | Beautiful | As beautiful as in the picture. couldn't go wr... | General | Skirts | Bottoms |
| 47 | Adorable and comfortable! | Just bought this in black at my local store an... | General | Knits | Tops |
| 29 | Must have, elegant, chic | This top! i was hesitant to try this on becaus... | General | Blouses | Tops |
| 38 | Very flattering fit | This is a great pair of trousers for work but ... | General Petite | Pants | Bottoms |

**Figure 5.2:** Workflow with the `multimodal-transformers` library, from Georgian (2021).

## 5.4 Specifying the hypotheses

I will now rephrase the hypotheses I formulated so far. The reason for this is that their wording is still shaped largely by the theoretical background I gave so far. However, to be strictly testable, they have to be more precise. Let us recall **H1**.

> **H1**: Using additional features to text-only classification tasks improves classification better if documents are not dialogical and can be seen as single atomic units.

Building on my elaborations since I proposed this hypothesis, I now reformulate it in a more specific way to make it testable within the framework of this study.

> **H1.1**: The average difference in weighted F1 scores between text-only transformers and multimodal transformers is significantly higher for the cooking data than for the voting data.

This means I measure how much better multimodal classification is compared with text-only classification. I do this by subtracting F1 scores for both from each other. I also do this for each task separately and then compare the computed values for cooking and voting. I will generally use weighted F1 scores as a measure for the classification's goodness.

For **H2** I postulated the following:

> **H2**: Multimodal transformers improve classification results compared to text-only transformers.

This now becomes:

> **H2.1**: Weighted F1 scores for multimodal transformers are significantly higher than for text-only transformers.

Now that we know about BERT and all the other transformers, we can handle **H3** a lot better and improve its wording for the study.

> **H3**: BERT, on average, delivers different classification results than newer transformer architectures.

This turns into:

> **H3.1**: The means of the weighted F1 scores of all experiments using text-only BERT models are significantly different from the mean of all other transformer models.

BERT here means models that use the same topology as the one proposed by Devlin et al. (2019). These are the first nine lines of table 5.2. Therefore, **H3.1** essentially compares the BERT models with the other models in the table.

Hypotheses **H5** and **H4** compare text-only transformers with classical ML models and other deep learning models respectively.

> **H4**: For small corpora, pre-trained transformer architectures deliver better results on pure text classification than classical non-deep learning models.
>
> **H5**: For small corpora, pre-trained transformer architectures deliver better results on pure text classification than non-attention-based deep learning models.

Compared with the other hypothesis, the last two are already quite precise. Nonetheless, I rephrase them and turn them into the following:

**H4.1**: On pure text classification of both tasks - voting and cooking, the average of pre-trained text-only transformer model's weighted F1 scores are significantly higher than those of classical non-deep-learning models.

**H5.1**: On pure text classification of both tasks - voting and cooking, the average of pre-trained text-only transformer model's' weighted F1 scores are significantly higher than those of non-attention-based deep learning models.

Some of the hypotheses may sound much more complicated now, but making testable empirical statements is no easy task either.

In the penultimate part of the study, I will do statistical analyses with the experimental output data. I will start with some plots to give a feeling for what happens in the data. I then proceed hypotheses-wise but will also make some digressions where I think it might be interesting.

# 6 Statistical analysis of experimental output data

## 6.1 First exploratory plots

Plotting is the most catchy and accessible method in data analysis. The problem is that there is usually a myriad of plots you could and would want to show. Unfortunately, you are usually constrained to a handful. Therefore, I start my statistical analysis with four of them to demonstrate what was happening during the experiments.

Let us jump right in and have a look at the following plot on the next page. The plot is rather complex. But when we are done with it, I hope to have given a good first understanding of the output data. The plot has three rows. The top is my 120 baseline models. The middle shows text-only transformers where classification was done just based on textual input. At the bottom, we see multimodal models, where apart from textual input, I also included other variables (i.e. features) like the Wandke annotations. Every dot is one F1 score weighted by target label support[1], coming from one experiment. Bear in mind that I mix the voting task and cooking task results here. They should, of course, be analysed separately and if together, they should be compared in some way. Also, I do not group by hyperparameters

[1]This means that the F1 score of different target labels influence the displayed global F1 score for the experiment run differently depending on how many samples the class has. This is especially important since I have an extremely unbalanced data set for the cooking task.

here. In the plot, you see models trained with one epoch as well as such that ran for six epochs. This makes the plot not a detailed view, but its purpose is to give only an idea of the output data as a starter. I will come to more detailed plots later.

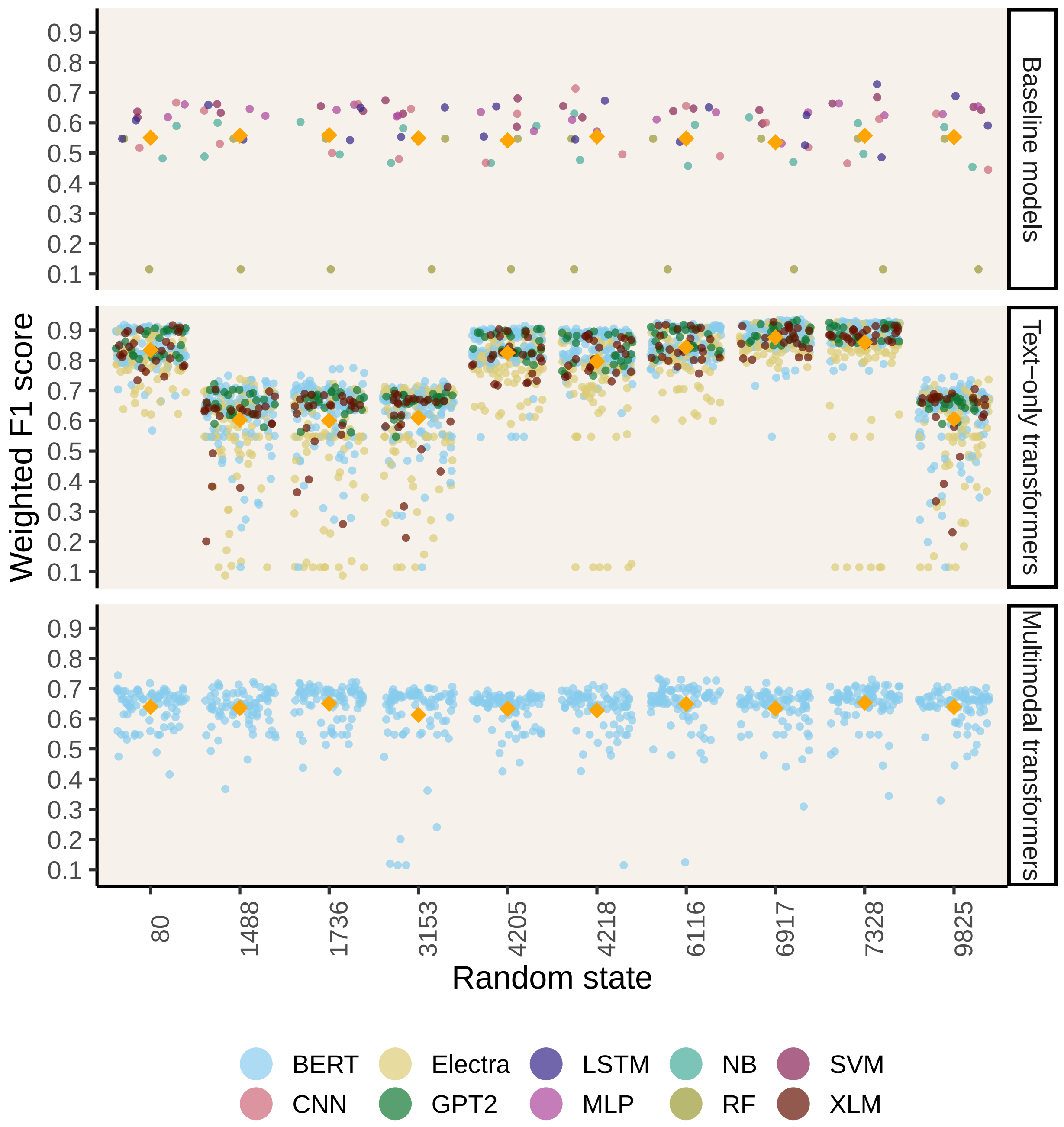


**Figure 6.1:** Comparison of all experiments over random states and model type. Note: Plot based on all 3720 experiments.

The horizontal axis shows random states. As I laid out in section 3.3, I ran every experiment ten times, with ten different random states. On the vertical axis, I plotted the F1 scores of all 3720 experiments. Apart from that, I coloured every dot according to the architecture used to produce the F1 score. For the sake of clarity, I summarised BERT, ConvBERT, and DistilBERT under BERT. This will be split up later. As we can see at the bottom, there are only pre-trained BERT models available for multimodal transformers. In the middle, we have various models, like BERT, Electra, GPT2, and XLM. At the top, we see the colours for NB, SVM, RF, MLP, CNN, and LSTM.

What does the plot tell us? It gives a first clue that normal transformers, which only process textual input (such as the open-ended survey response or the utterance during the cooking experiment) perform best. However, we also see that they are also most prone to variation according to different random states. Of course, random state is not a hyperparameter to tune for, but it nonetheless influences model performance. Therefore, volatility in this respect is not a desired property of a classification model. Text-only transformers seem to have two levels of best performance. That is F1 scores around 70 and around 90 per cent. Depending on the random states, they perform either on par or better than both other model families, at least as far as this first exploratory plot can tell us.

One can also zero in on the text-only transformers in the middle row of the plot. Here no perfectly distinct pattern of how the points cluster within each of the ten groups can be seen. If you would really want to spot any patterns here, one might say that Electra usually is the worst performing model while still also reaching high scores. Furthermore, the Electra colour usually does not reach as high as BERTs bright blue. However, I would rather say that models in the middle pane are scattering pretty much all over the realized score range from .1 to .9 while no model clusters in one score region.[2]

Let us now have a closer look at the baseline models at the top of the plot. They are by far fewer experiments than the others, for reasons I already elaborated. Still, 120 experiments over ten groups are statistically interpretable. What stands out right away here is the poor performance of all random forest models. They reach F1 scores of around .1 and .55 for the cooking and voting task, respectively. Worse

[2]GPT2 is exempt from this. Its range is narrow for other reasons that I will come to shortly.

scores would be tough to achieve. You would probably have to train the model to do wrong classifications to worsen them further. This is so since below .1 and .55 would equate coin tosses, i.e. random classification, for binary and 13 label tasks.

With the RF models, I am inclined to say that this is not due to their inherent inaptness to do well on these tasks. It is probably the featurization method I used with RF, namely tf-idf, which is too low dimensional for the forest. Also, the hyperparameter sweep I did for RF models might have been too narrow. In the end, I can only speculate why RF yields such dire results. More experiments would probably be needed to investigate that.

The other experiments in the baseline row border the 70 per cent score mark. They are comparable to the multimodal models in the bottom or to the worse performing groups in the middle, where text-only transformers were used.

In the next plot 6.2, I divided by classification task, namely swing voter classification or cooking information need classification. The picture does not change too much, except in the middle row, where we can see that text-only transformers seem to perform a tiny bit worse on the binary classification. Why that might be so, we can only guess for now. After all, we do not know whether it only looks that way in the plot or if there is an actual statistically significant difference here. Also, a binary task is not necessarily easier than a multiclass problem. Empirically, in this case, the reason is that there is a trade-off between correctly predicting swing voters or non-swing voters. There are only two options, and their correct classification is, to some degree, mutually exclusive. After that, the classifier has nowhere to go. In the multi-label problem, when losing ground on some class, the classifier has many options to choose from where it can make up for some lost ground. Speaking in the optimization language, the learned function has more options on how to maximize. Therefore, it is easier for the classifier to find a reasonable local optimum during searching.

What can also be seen is that in the binary problem, no model performs worse than a coin toss. It comes as no surprise since that is what we are here for in the first place. If the F1 score would be worse than chance on any input data, the classifier should be questioned. Even the screwed-up RF model for the cooking task is not worse than chance, which in that case would be $\frac{1}{13}$ or .08. We see that the RF in all cases performs around .11, at least a bit better than throwing a 13 sided dice. Again, the multimodal transformers seem to stabilize results between the two tasks. Whereas the means (the orange diamond) in the other two horizontal panels are either lower for the cooking or the voting task, the bottom results are the same for both tasks. I already said that text-only transformers seem to perform minimaly worse on the binary classification. On the other hand, the baseline models seem to have a more challenging time on the multilabel problem. Do not be fooled by the mean here in the top left facet, which is heavily skewed by the lousy performing RF. Nonetheless, if we ignore comparing the means in the top row here, we can see that NB and LSTM perform better on the voting data problem.

One more thing that we see in the middle row is that for the text-only transformers are that the range of scattering holds for both tasks and all models within them. There is no clear pattern where we could say that this or that model can only be seen within some vertical axis range.

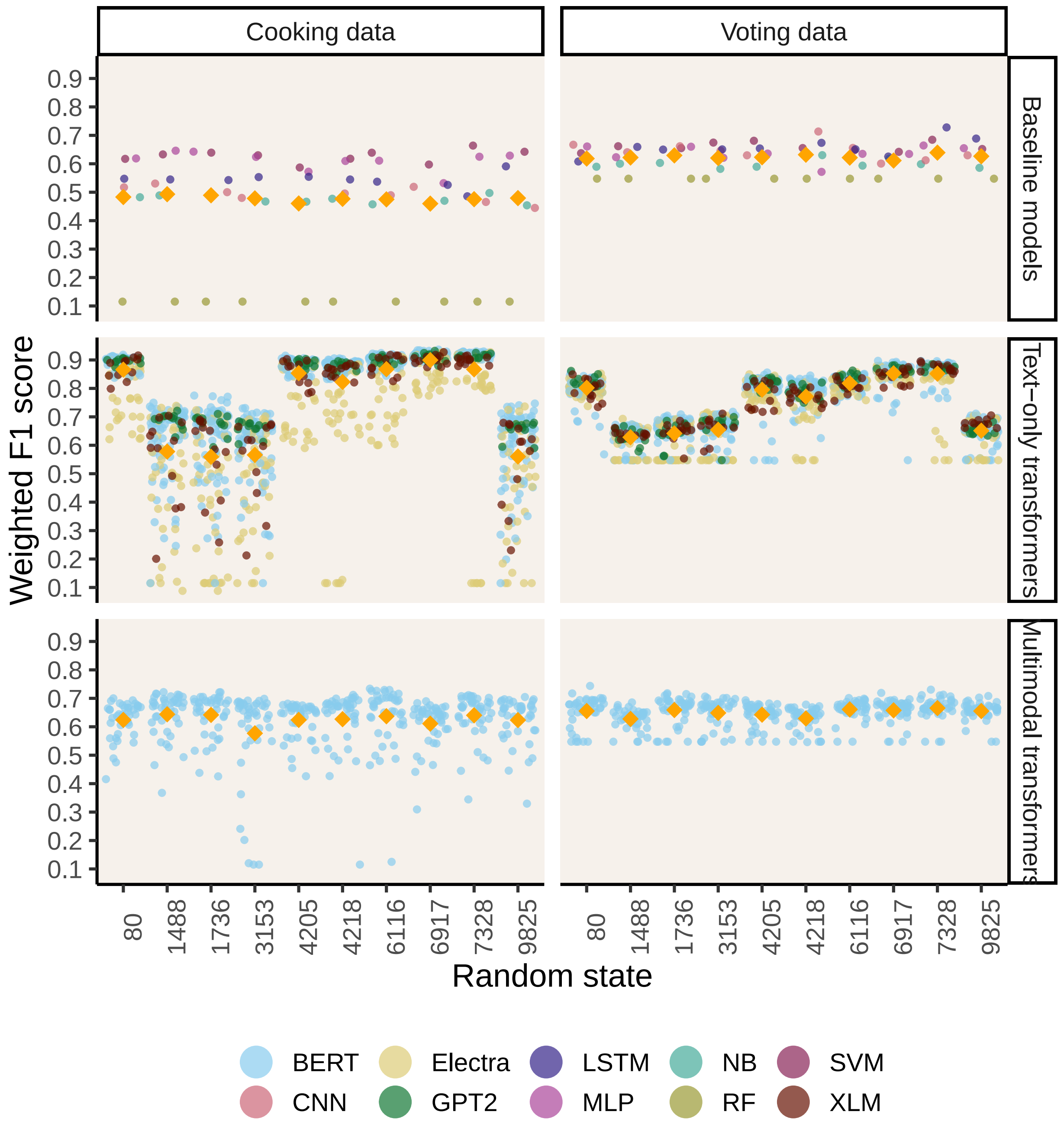


**Figure 6.2:** Comparison of all experiments over random states, model type, and task. Note: Plot based on all 3720 experiments.

In the third plot in figure 6.3, we zoom into the middle row and have a closer look at the normal transformers. The effect of random states exists, and it is considerable. Judging from the previous two figures, the effect can be as large as 20 per cent in F1 score difference. One group of random states produces results that are .15 to .20 better than those of the other. Running ten experiments shows an almost half-half split. There are four random states with lower results and six with higher. This hints that there will be a 50/50 split in the long run as you repeat these experiments more often. This argument is strengthened by the fact that the split holds for all of 2760 text-only transformer experiments.

Therefore, in plot 6.3, I average over random states to zero in on the effects, the number of epochs has on classification results. It comes as no surprise that more training improves model performance. In the cooking task, the effect is much more pronounced than in the voting task. In the former, the weighted F1 scores increase from .66 with one epoch to .77 with four epochs. The curve flattens after that. Training for more than four epochs does not improve results further. That is in line with Devlin et al. (2019), who advise fine-tuning BERT models for two to four epochs, which seems to be a good choice for other transformer models as well. On the right of the plot, we see the voting task. There F1 scores, on average, only improve by .02 or .03 when comparing one epoch training versus three epoch training. The measure even goes down if you fine-tune the models too long. We clearly see that fine-tuning time is a much more crucial factor in the multilabel task than in the binary task.

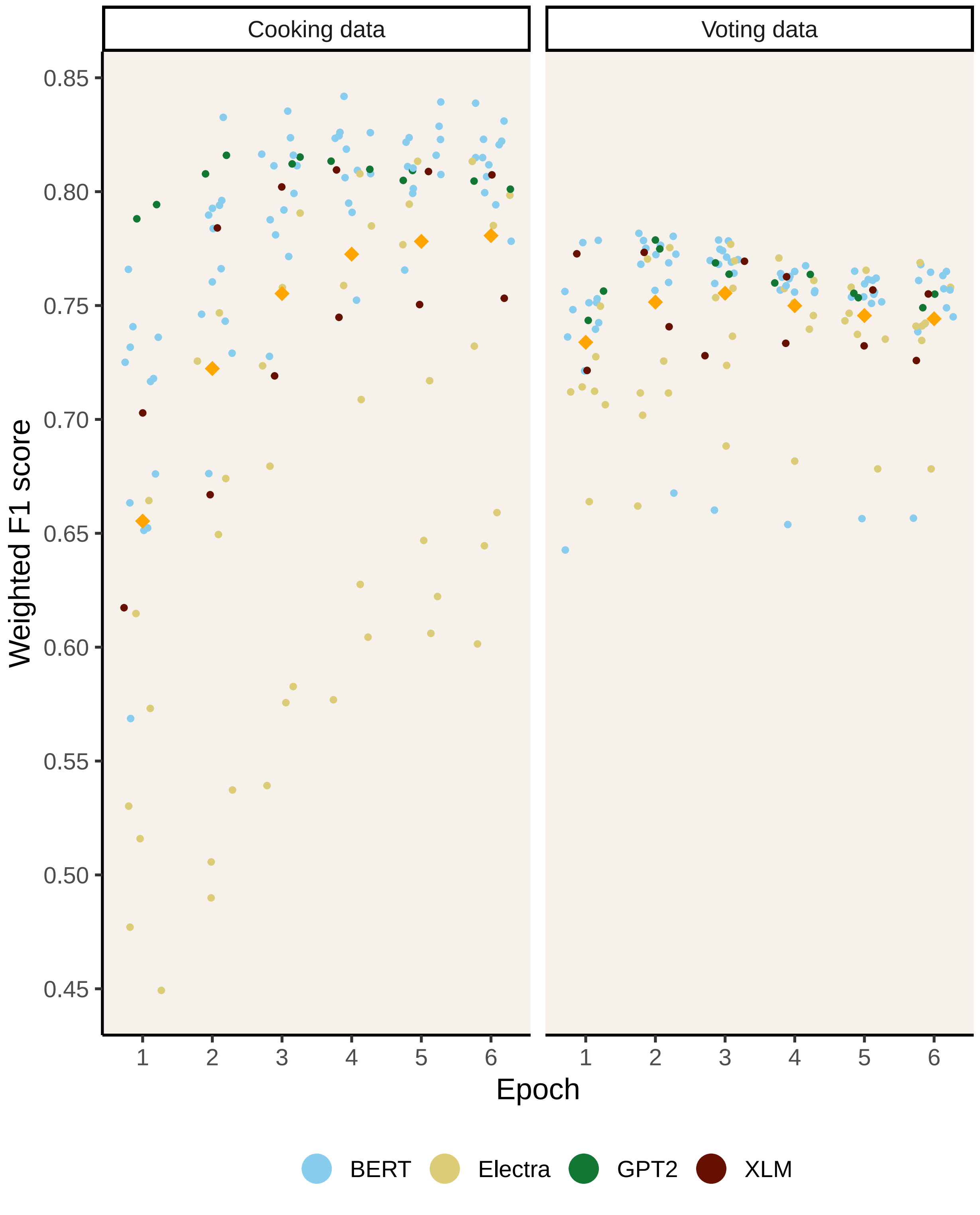


**Figure 6.3:** Comparison of all text-only transformer experiments by task. Note: Plot based on 2240 text-only experiments. F1 scores were averaged over random states.

Now that we have looked into the 'model family view', meaning we only compared broader architecture groups, we can now drill down even more. As I already pointed out, there is not only the difference between BERT, Electra and so on; there are also different pre-trained versions of BERT. This difference has been obscured by my plots so far. In plot 6.4 I compare BERT, ConvBERT and DistilBERT.

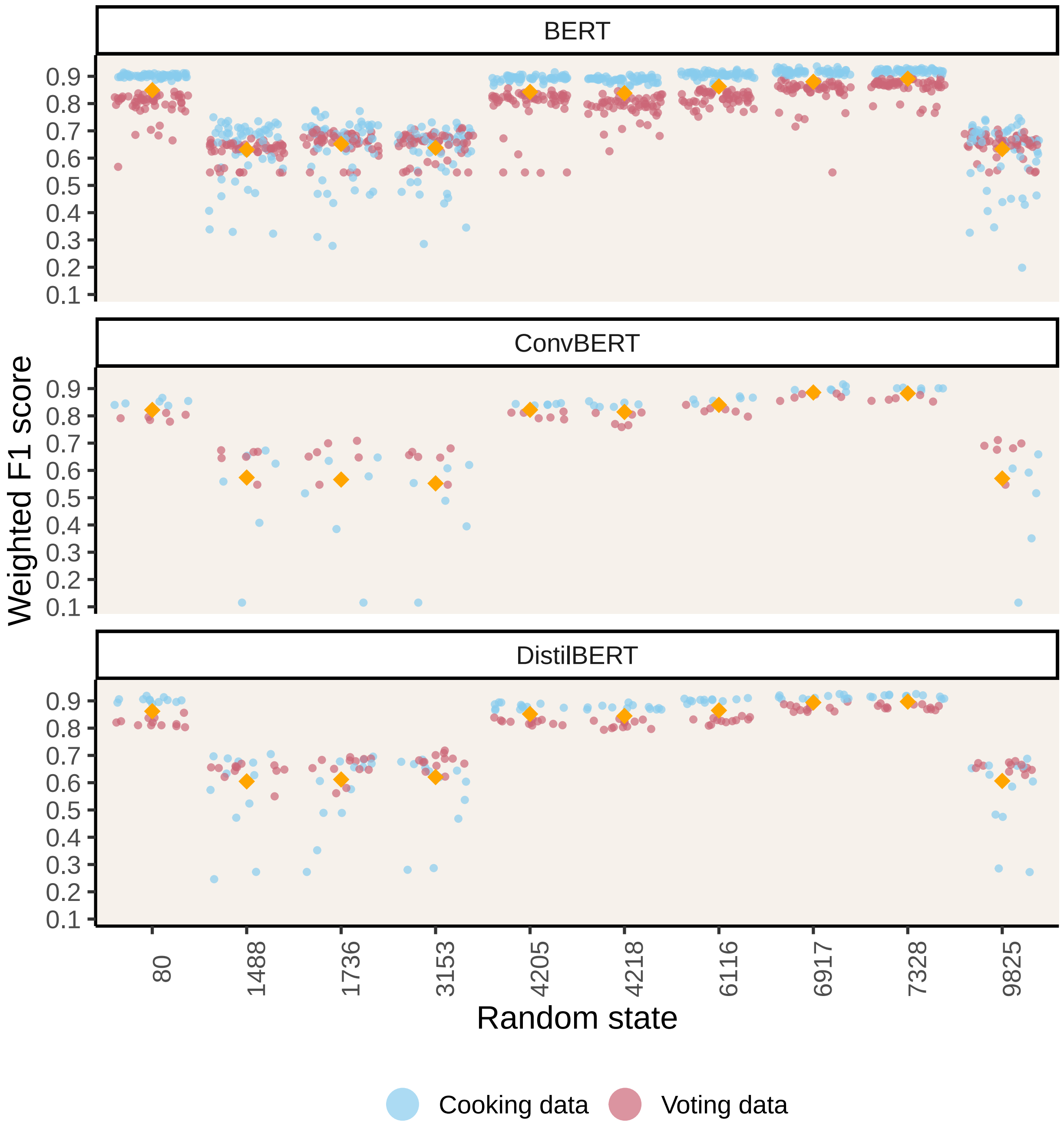


**Figure 6.4:** Comparison of all text-only BERT, ConvBERT and DistilBERT. Note: Plot based on 1440 text-only experiments.

The random states effect again holds for all three architectures and at first glance, they seem to yield quite comparable results. When looking closer, one sees that this is only true for DistilBERT and BERT. The F1 score of ConvBERT is a little lower across the board. Remember from section 5.2 that ConvBERT made some changes to better address local dependencies in sentences. This degrades the classification. The examples from the voting data in section 4.1 show that the larger self-attention picture is crucial since there can be long-range dependencies in the data. That is also true for the cooking data.

What also strikes here is that for some random states voting task and cooking task results overlap, while they separate for others. They even switch position in the very right of the middle row, meaning ConvBERT here does better on the voting task. However, the larger picture shows the models are mostly doing better on the cooking data. Their results are also generally very comparable, with only minimally worse scores for ConvBERT.

## 6.2 Testing the hypotheses

There would be a lot more to analyse. To keep it straightforward, I will progress hypothesis-wise from here on. As a measure of how good the classification is, as already said, I will use weighted F1 scores. I know there is a myriad of other measures one could look at. However, going into more than one score for so many experiments is beyond the scope of this study. I chose weighted F1 since it is widely known and accepted. It also combines precision and recall into one measure. I also preferred the weighted version over the macro version since the labels are very unbalanced.

### 6.2.1 Hypothesis 1

Although I might repeat myself, here is **H1.1** again:

> **H1.1**: The average difference in weighted F1 scores between text-only transformers and multimodal transformers is significant higher for the cooking task then for the voting task

From the five hypotheses **H1.1** is arguably most difficult to test. Therefore, I propose two ways to do so.

**Test 1** For the first test, I calculate the differences between text-only and multimodal transformers for both tasks. This is no simple setup and I want to make sure we are all on the same page on this. I visualise what I mean it in figure 6.5 below. The sketch shows two experiments. Both receive as input the same random state and the same pre-trained model. One is a text-only experiment. Therefore, it receives no additional categorical or numerical information. Experiments are run on the cooking and voting task and produce weighted F1 scores for both. I then have four F1 scores; two for voting and two for cooking, two multimodal and two text-only, namely *text-only-cooking, text-only-voting, multimodal-cooking, and multimodal-voting.* I now subtract multimodal-cooking from text-only-cooking and multimodal-voting from text-only-voting. The result is two differences, one compares voting task and one compares cooking task. These are the differences for testing the hypothesis. Their interpretation is that if the one is higher than the other, text-only models performed better on that task than on the other.

I opt to select only those pre-trained transformers that are present for text-only and multimodal models. This is limited by the latter, as pointed out in section 5.3. Figure 6.6 shows the empirical distribution of the differences on the left and means with their 95 per cent confidence intervals (CI) on the right. A reading example for a fictional point in the boxplots helps to understand what is going on here. Say there is a data point plotted at exactly 0.0 on the vertical axis. That means that the F1 score difference between the text-only experiment and its multimodal counterpart is zero, i.e. their classification was equally good or bad. Another example: A value of -0.1 for the experiment using the pre-trained model `bert-base-german-cased` and training for one epoch, means that here the multimodal model performed 0.1

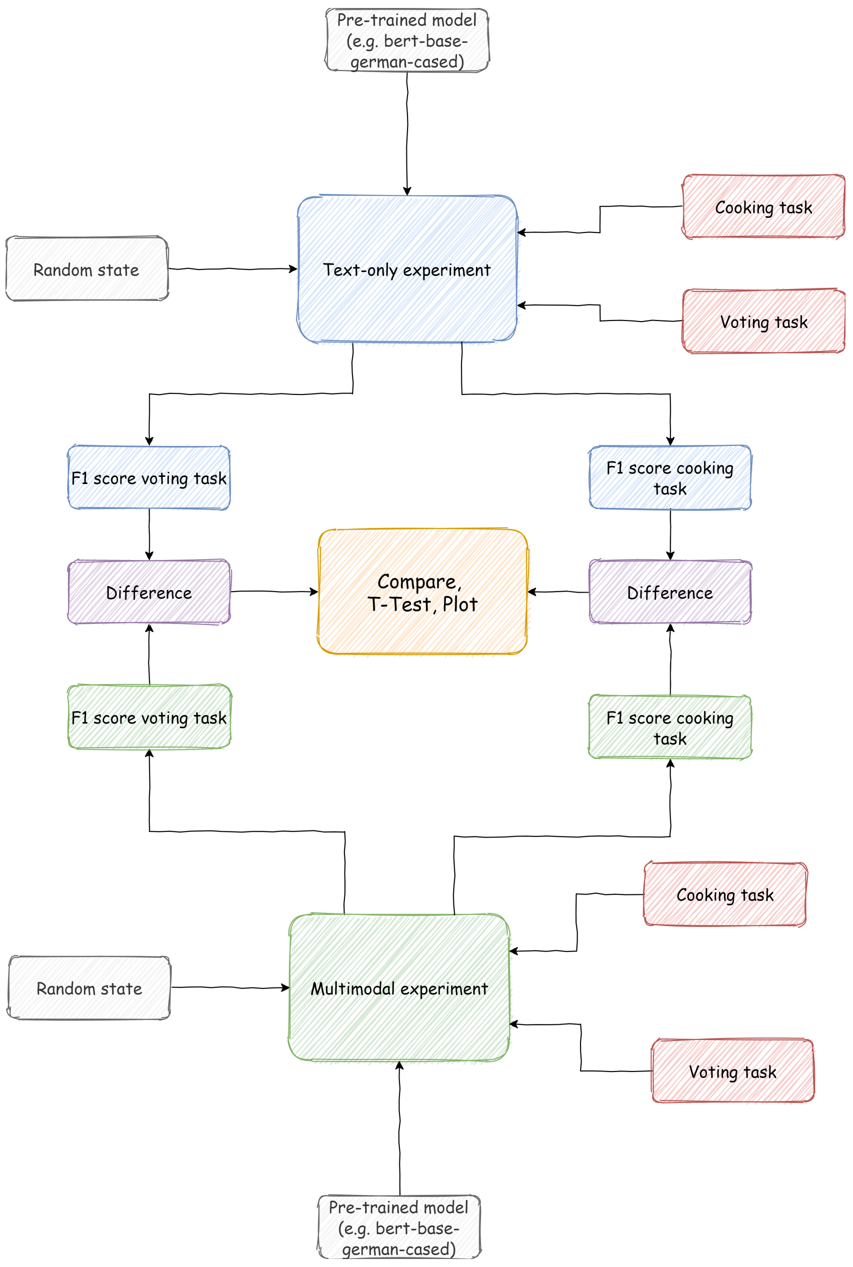


**Figure 6.5:** Test setup for H1.1.

F1 score points *better* then the text-only transformer. All needed experiments have been carried out in text-only and multimodal versions. The gap you see in the middle of the cloud of data points is due to the effect of the random states that I already pointed out when describing plot 6.1. The effect is present for both tasks and should not distort any signal when testing H1. The overall picture is clear. In both tasks, text-only models perform better on average. That is so because most of the data cloud is above zero, which means that the F1 score for most text-only models is higher than for their multimodal counterpart.

What has to be tested is whether the cooking task's mean is lower (i.e. multimodal models performed better on this task) than the mean for voting data. That is not the case and can easily be seen in the left of figure 6.6. A one-sided T-test confirms this with a p-value of 1 (see table 6.1). The means are significantly different from one another but in the opposite direction as the hypothesis proposes. Therefore, multimodal models, compared to text-only models, improve performance more in the voting task and not in the cooking task.

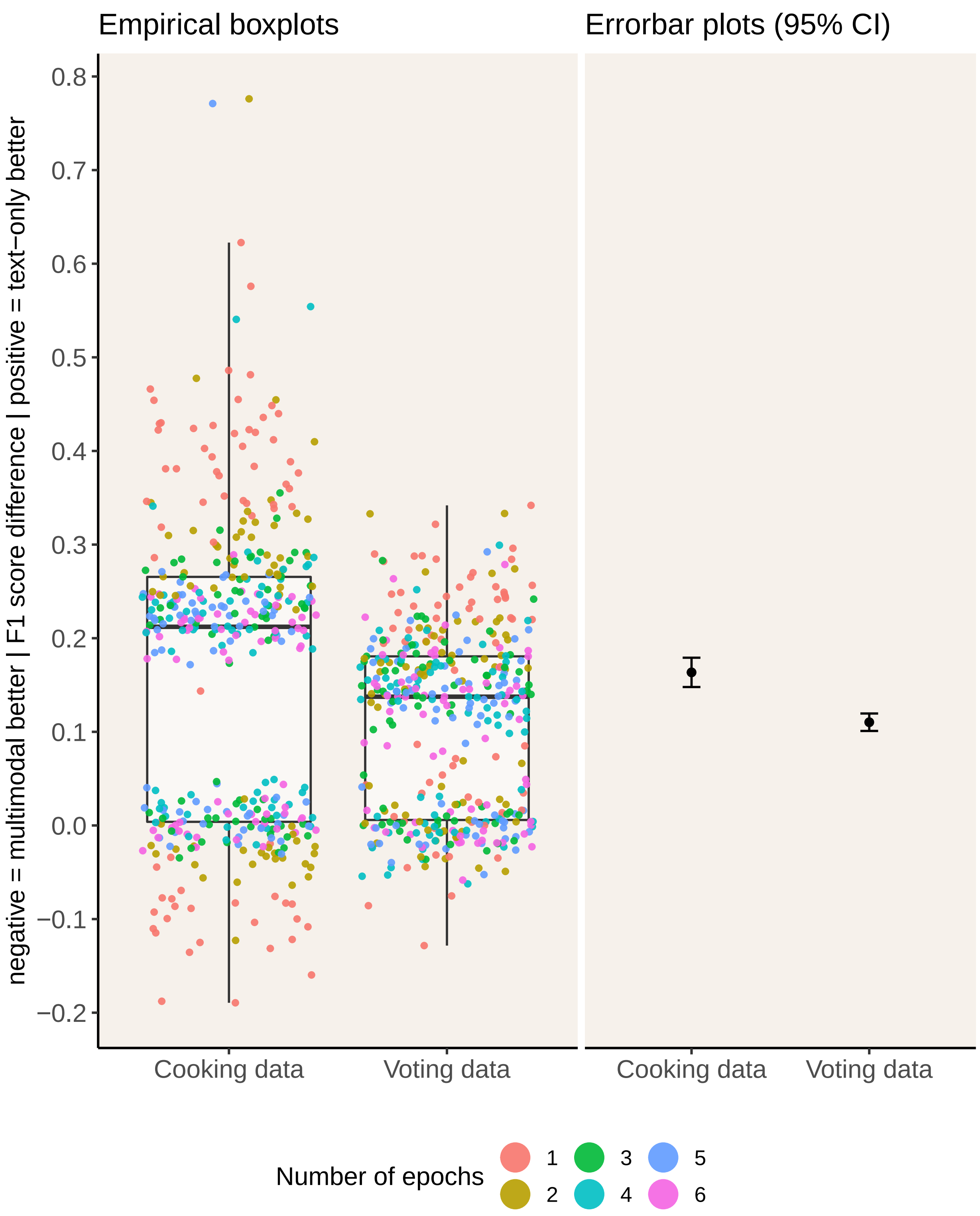


**Figure 6.6:** Comparison between text-only and multimodal BERT models. Note: Plot based on 840 text-only BERT experiments and 840 multimodal BERT experiments.

**Table 6.1:** T-Test comparing differences between text-only and multimodal transformers - voting versus cooking task.

| T value | p value | Mean cooking | Mean voting | Lower 95% CI | Mean difference | Upper 95% CI |
|---|---|---|---|---|---|---|
| 5.739 | 1 | 0.164 | 0.11 | 0.035 | 0.053 | 0.071 |

Note: n=1680.

I tested the one-sided hypothesis that the true mean for the cooking task is *lower* than for the voting task. The error bar plot above already told us that this stands no chance of being significant. The T-test above also confirms this. The point estimate says in the cooking task, text-only models, on average, perform .053 F1 score points better. Since F1 goes from 0 to 1, you could also express this as percentages. That does not seem like a huge effect. However, when you consider the number of experiments, it nonetheless quite substantial.

It should also be noted that *means are positive in both tasks.* That also indicates that text-only models generally performed better. However, this is not strictly what the hypothesis is about. It is about whether multimodal models improved performance in the cooking task better than in the voting task. Therefore, I only care if the cooking task mean is significantly lower, then the voting task mean. Whether it is positive or not is secondary to the hypothesis.

Against my approach, one could undoubtedly bring forward that both tasks are not comparable. The one is binary, the other multi-label. I see that argument and this is why I propose a second test to **H1.1**. I turn the cooking data results into a binary problem by only retaining the labels for preparation and ingredient.[3]

**Test 2** I take preparation and ingredient as the two labels since they have the most samples, which keeps the two task's sample sizes comparable. It also made it easier for the cooking task classifiers to yield good and stable results on the two labels. That, on the other hand, means it is an all in test for the hypothesis. If it fails here, it is improbable to be confirmed on any other combination of labels from

[3]If one has the F1 scores of all 13 labels, one can easily calculate the weighted F1 scores for only two labels. The number of classes and samples taken into account has to be changed. Of course, this is also not the perfect approach. In future experiments, comparing more similar tasks might be a step in the right direction. We will see shortly why comparing such unalike tasks still yielded some valuable insights.

the cooking data. It does also not take away the dialogical nature of the cooking task. Every information the classifier gets about this is given by annotated features in the first place; and these features are retained for the two labels. Since the plot from figure 6.6 hardly changes, I only report the T-test results below.

**Table 6.2:** T-Test comparing differences between text-only and multimodal transformers - voting versus cooking task - version with both tasks binary.

| T value | p value | Mean cooking | Mean voting | Lower 95% CI | Mean difference | Upper 95% CI |
|---:|---:|---:|---:|---:|---:|---:|
| 2.807 | 0.997 | 0.132 | 0.11 | 0.006 | 0.022 | 0.038 |

Note: n=1680.

The difference decreases by about half and the CI also narrows considerably. Nonetheless, the mean stays higher for the cooking task. Compared with text-only models, multimodal models still performed worse on the cooking task than on the voting task. This holds even as the number of target labels becomes the same. *In the light of these findings, I reject* **H1.1**.

### 6.2.2 Hypothesis 2

**H2.1** picks up here and compares both kinds of transformers directly.

> **H2.1**: Weighted F1 scores for multimodal transformers are significantly higher than for text-only transformers.

Other then **H1.1**, **H2.1** is not concerned with the difference between multimodal and text-only transformers over the two tasks. It directly focuses on comparing both variant's F1 scores. Therefore, this hypothesis does not compare tasks anymore.

The reason I chose both datasets is that they make a good contrast. If one finds results to be equal with both datasets, the reasons for this can not be the different characteristics of the data sets. The general picture is that transformers that use additional features apart from text perform worse on both tasks. Plot 6.7 reiterates the finding from plot 6.1. Here I zeroed in on the BERTs that are available for text-only and multimodal. We can see the performance of the same algorithms and the same models on two very different tasks and we see that the kind of the task does not matter; results look very alike. The influence of the number of epochs one trains the model differs a little, but that is it. Two different tasks, same performance. This, in turn, means that all the differences the two tasks have can not be the reason for these results since different task characteristics must lead to different results. One possibility would be that task characteristics cancel each other out perfectly. However, that is very unlikely.

*It also reinforces the findings around* **H1.1** that for the classifiers, it does not matter whether the data comes from a dialogue or not, i.e. whether one includes the additional features in the cooking task. One more reason for this could also be that the features we annotated do not capture the complexities of human dialogue well enough. However, I find it unlikely that they carry no additional information whatsoever and are only noise for the classifier. Therefore, again, and since I applied the same models to both tasks, the characteristics that differentiate both tasks can not be why I get those same results. What these reasons are I can only speculate on. More digging into the output data and probably more experiments are needed here. I will give an outlook on that in the last part.

I am not going to reproduce a plot of the kind I did for **H1.1** here since it would be redundant, and anyone can figure from plot 6.7 what it would show. It

**Table 6.3:** T-Test comparing weighted F1 scores of text-only and multimodal transformers.

| T value | p value | Mean cooking | Mean voting | Lower 95% CI | Mean difference | Upper 95% CI |
|---|---|---|---|---|---|---|
| -26.382 | 1 | 0.638 | 0.774 | -0.147 | -0.137 | -0.126 |

Note: n=1680.

would show text-only having higher average performance then multimodal models. The T-test in table 6.3 confirms this.

Again, as already anticipated after testing **H1.1**, there is no possibility of the one mean being higher than the other. **H2.1** *has to be rejected for this study.*

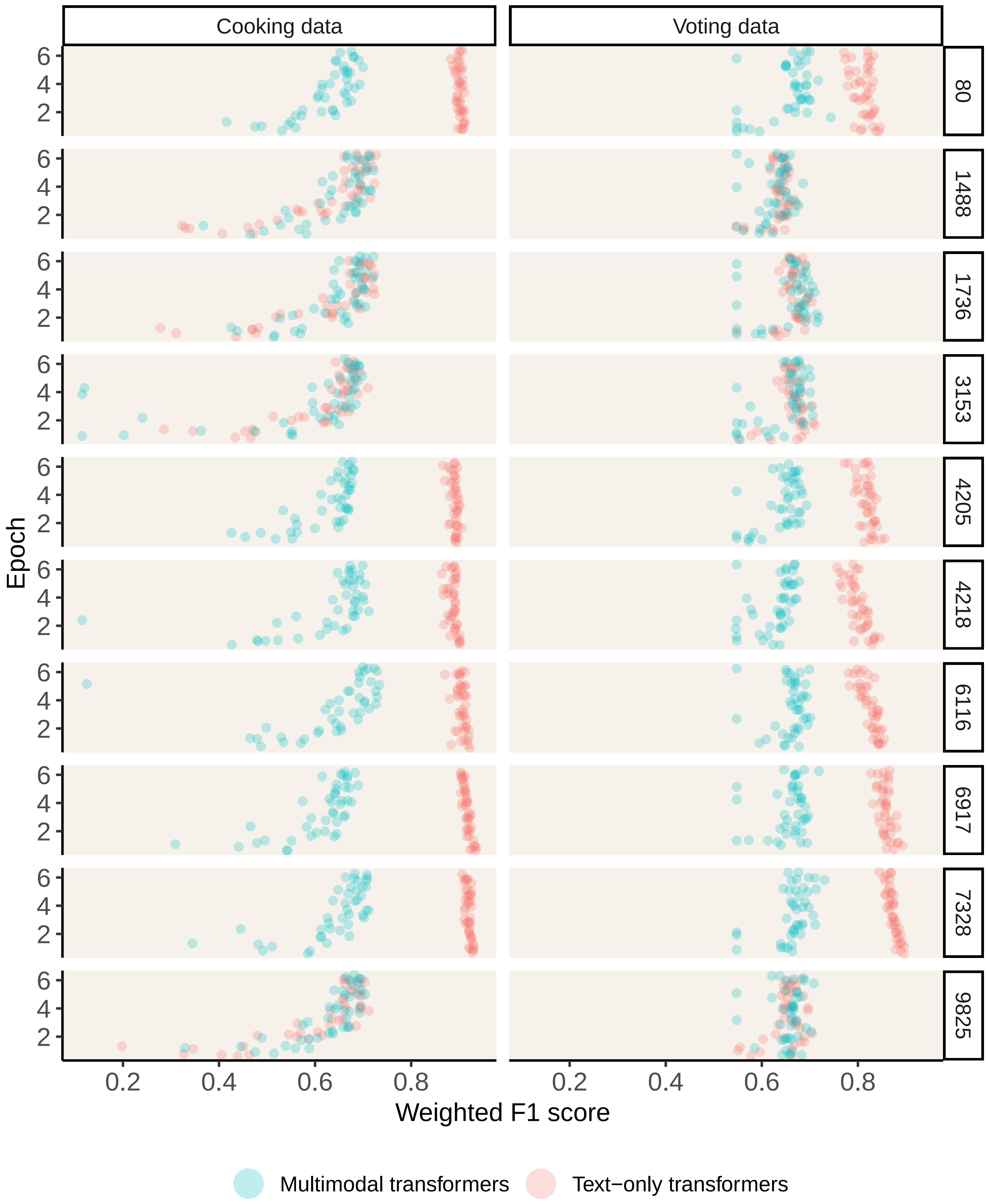


**Figure 6.7:** Scatterplot comparing text-only and multimodal transformers. Note: Plot based on 840 text-only BERT experiments and 840 multimodal BERT experiments.

### 6.2.3 Hypothesis 3

Up to here, I focused on F1 differences between tasks as well as multimodal vs text-only models. **H3.1** addresses whether we can see improvements when departing from the vanilla BERT topology.

> **H3.1**: The means of the weighted F1 scores of all experiments using text-only vanilla BERT models are significantly different from the mean of all other transformer models.

One could test this with yet another T-test where BERT and not BERT are being compared. However, it would be rather crude to lump all the other transformer models together like this. As I elaborated in section 5.2, they are all very different in their topology and approach to modelling the human language. Therefore, I will run a one-way analysis of variance (ANOVA) to check whether all these means differ. Furthermore, although the hypothesis does not say anything about the effect's direction, I am also interested in that. To this end, Tukey's *honestly significant difference* (HSD) test is the best approach. It extends the idea of the omnibus test from an ANOVA by computing pairwise comparisons of all means to see which mean is larger or smaller than the other. (Abdi and Williams 2010) At the same time, the HSD test corrects for the problem of the familywise error that would arise when one would simply make many T-tests. (Haynes 2013; Field and Iles 2016, p. 708)

Note that here comes into play an issue in implementation. For technical reasons, I used a batch size of one for the GPT2 models. This is why GPT2 is a good deal better than the others in plot 6.8 and why its distribution is narrower. This did not influence **H1.1** and **H2.1** because I did not compare GPT2 there. The bias introduced by this was equally split to all groups compared in **H1.1** and **H2.1**.

Plot 6.8 shows the empirical distributions as violin plots and errorbar plots to compare the six transformer topologies employed in this this study. Almost all the violins have a comparable range. The only exception here is GPT2, for reasons already explained. I will, therefore, not make any judgements towards GPT2 and only give some tendencies. The confidence intervals of the topologies are mostly overlapping. However, Electra only overlaps with ConvBERT.

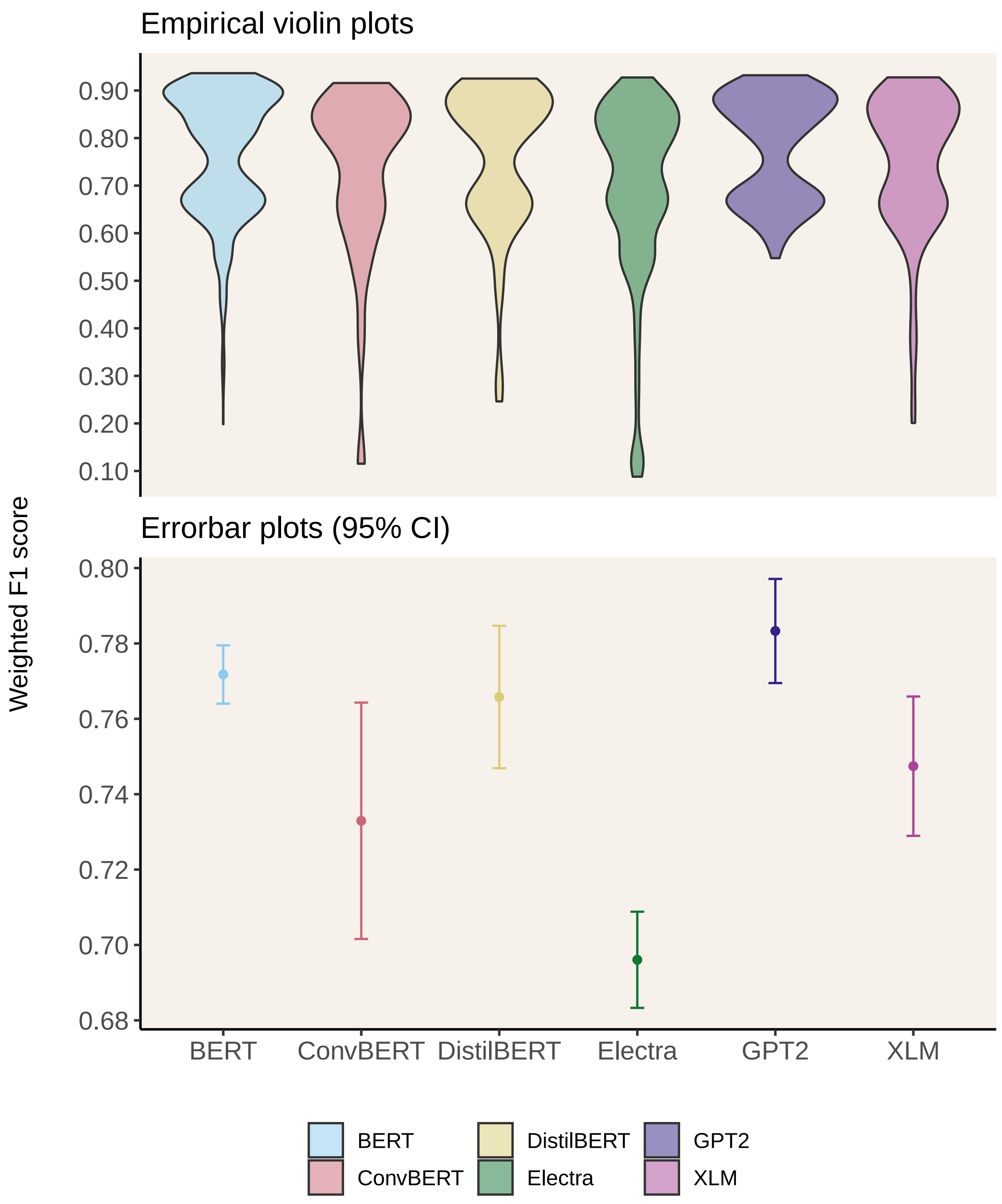


**Figure 6.8:** Comparison between transformer topologies. Note: Plot based on 2760 text-only transformer experiments.

Let us have a look at the one-way ANOVA in table 6.4, where topology (i.e. BERT, ConvBERT, etc.) is the independent grouping variable.

**Table 6.4:** One-way ANOVA results for comparison between transformers.

| | Df | SS | MSS | F value | p value |
|---|---|---|---|---|---|
| Model | 5 | 3.259 | 0.652 | 27.791 | 0 |
| Residual | 2754 | 64.598 | 0.023 | NA | NA |

Note: n=2760. For the number of experiments per group, see the table in section 3.4.

Hypothesis **H3.1** can be confirmed when only looking at the ANOVA. However, as already stated, I am not quite content with an omnibus test here that merely looks at whether all the means are different. Therefore, have a look at table 6.5.

**Table 6.5:** Tukey's HSD Test for all possible pairwise comparisons of transformer topologies.

| | Comparison | Lower 95% CI | Mean difference | Upper 95% CI | p value |
|---|---|---|---|---|---|
| 1 | ConvBERT-BERT | -0.081 | -0.039 | 0.003 | 0.090 |
| 2 | DistilBERT-BERT | -0.037 | -0.006 | 0.025 | 0.994 |
| 3 | Electra-BERT | -0.096 | -0.076 | -0.056 | 0.000 |
| 4 | GPT2-BERT | -0.020 | 0.012 | 0.043 | 0.899 |
| 5 | XLM-BERT | -0.055 | -0.024 | 0.007 | 0.226 |
| 6 | DistilBERT-ConvBERT | -0.016 | 0.033 | 0.082 | 0.391 |
| 7 | Electra-ConvBERT | -0.080 | -0.037 | 0.006 | 0.134 |
| 8 | GPT2-ConvBERT | 0.002 | 0.050 | 0.099 | 0.039 |
| 9 | XLM-ConvBERT | -0.034 | 0.014 | 0.063 | 0.959 |
| 10 | Electra-DistilBERT | -0.102 | -0.070 | -0.038 | 0.000 |
| 11 | GPT2-DistilBERT | -0.022 | 0.018 | 0.057 | 0.811 |
| 12 | XLM-DistilBERT | -0.058 | -0.018 | 0.022 | 0.778 |
| 13 | GPT2-Electra | 0.055 | 0.087 | 0.119 | 0.000 |
| 14 | XLM-Electra | 0.019 | 0.051 | 0.083 | 0.000 |
| 15 | XLM-GPT2 | -0.076 | -0.036 | 0.004 | 0.106 |

Note: n=2760. For the number of experiments per group, see the table in section 3.4.

Looking at the HSD test, the results for **H3.1** are not as clear anymore. A positive value in the third column means that the first-named model in column one was better. A negative value indicates that the one named second performed better. For instance, in line three is says Electra-BERT. The mean difference overall between Electra and BERT experiments here is -0.076. On average BERT performed 7.6 weighted F1 score points better on both classification tasks. Column two and four give the 95 per cent confidence interval of the respective means. Line three in the given example is also the only one that is highly significant and includes BERT. Significant at the five per cent level is also line eight, which compares GPT2 and ConvBERT. However, for reasons elaborated above, this should not be overstated. This performance comparison would not be significant, if both models would be more strictly comparable.

Highly significant is the pair in line 13, where also GPT2 is compared, this time with Electra. Although GPT2 results should be taken with a grain of salt, I am convinced that this finding would hold even when GPT2 and Electra were implemented so as they could be compared more strictly.

One last thing that can be taken at face value is XLM versus Electra, where again XLM outperforms Electra by more than five per cent on average. Electra is also generally the least performing model. So the usage of generative adversarial networks on classification might not be the best choice.

What does all of this mean for **H3.1**? I am inclined to reject it since vanilla BERT performed significantly better in only one single comparison. However, since only two or three from 13 comparisons were significant at all, one could also argue that BERT is still among the top scorers; and that is what **H3.1** was all about. *Therefore, I neither reject nor confirm* **H3.1** *for this study. My methodology and experiments were not suited to test it thoroughly enough to make any reliable statement here.*

### 6.2.4 Hypotheses 4 and 5

My last two hypotheses are best tested in one sweep since they concern comparisons of the three large groups, the ones of which I showed a plot in figure 6.1 at the very beginning of this part. There is no empirical ML thesis without testing your baseline classification results against the rest. **H4.1** does this for non-deep-learning baseline models. **H5.1** makes the comparison of transformers with baseline deep-learning models, namely MLP, CNN, and LSTM.

> **H4.1**: On pure text classification of both tasks - voting and cooking, the average of pre-trained text-only transformer model's weighted F1 scores are significantly higher than those of classical non-deep-learning models.
>
> **H5.1**: On pure text classification of both tasks - voting and cooking, the average of pre-trained text-only transformer model's weighted F1 scores are significantly higher than those of non-attention-based deep learning models.

For **H5.1**, there are 60 experiments in total. I ran classifiers NB, RF, and SVM for ten different random states on two different tasks. These 60 baseline experiments will be compared against 2760 text-only transformer experiments. That is a pretty uneven distribution of observations. Nonetheless, both groups are large enough to employ methods of inferential statistics on them. I am aware of the pitfalls when comparing groups with such a different number of observations, which increases the likelihood that there are no equal variances between them. It can not be denied that compared with the group comparisons made so far in this study, variances and number of observations per group are highly different here. Compared with the others, confidence intervals are extremely narrow for the rightmost group in the 6.9. That is, due to the sheer number of data points in that group. Nonetheless, the effect size is large and p values are low enough to consider the results robust.

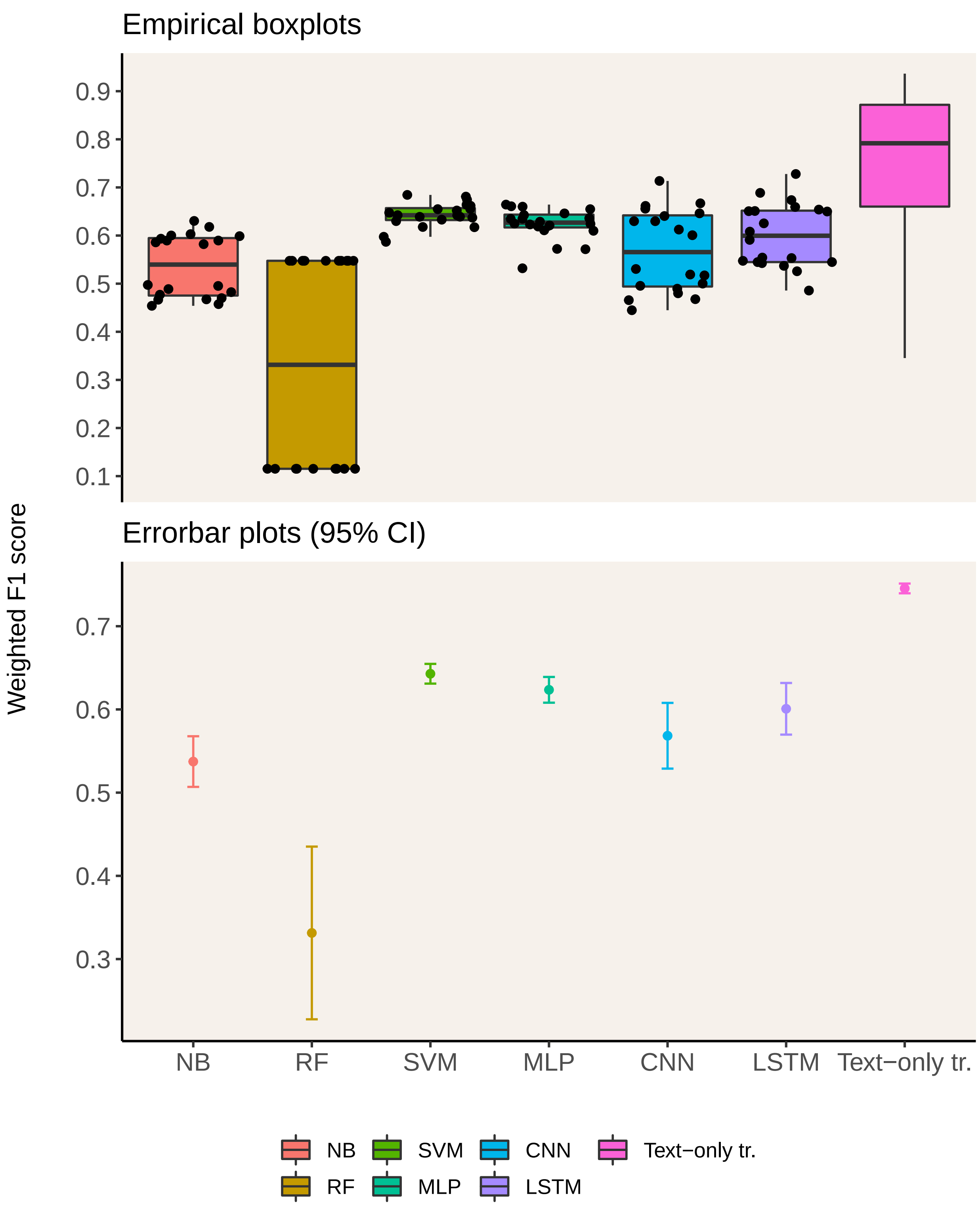


**Figure 6.9:** Comparison between text-only transformers and baseline models. Note: Plot based on 2880 baseline and text-only experiments.

Table 6.6 shows the HSD test results for all the pairwise comparisons between the baseline models and transformer models. The result is pretty straightforward. On average, transformer networks performed better than any of the baseline models.

**Table 6.6:** Tukey's HSD Test for text-only versus baseline comparison.

| | Comparison | Lower 95% CI | Mean difference | Upper 95% CI | p value |
|---|---|---|---|---|---|
| 1 | Text-only tr.-NB | 0.105 | 0.208 | 0.311 | 0.000 |
| 2 | Text-only tr.-RF | 0.311 | 0.414 | 0.517 | 0.000 |
| 3 | Text-only tr.-SVM | 0.000 | 0.103 | 0.205 | 0.051 |
| 4 | Text-only tr.-MLP | 0.019 | 0.122 | 0.225 | 0.009 |
| 5 | Text-only tr.-CNN | 0.074 | 0.177 | 0.280 | 0.000 |
| 6 | Text-only tr.-LSTM | 0.042 | 0.145 | 0.247 | 0.001 |

Note: n=2880. For the number of experiments per group, see the table in section 3.4.

Only the comparison with SVM borders statistical significance with a p-value of .051 and should not be taken at face value. All the others are significant at a generally accepted level. What also strikes is that none of the confidence intervals (except SVM) includes zero, which means that it is improbable that the actual effect is equal to zero. *Therefore,* **H4.1** *and* **H5.1** *can be accepted.*

### 6.2.5 Summary of hypothesis tests

Looking at mere counts, I rejected two of my hypotheses and accepted another two. One was left undecided. However, **H1.1** and **H2.1**, the two I rejected, were the most interesting and critical for this study. Both hypotheses were not simply rejected, but the result turned out to be quite the opposite of what I anticipated. So at least there is some insight.

**H1.1** showed that using additional features improved classification more when dealing with a non-dialogical corpus. Thus, the idea of modelling human dialogue flow for the cooking data could not improve performance. I ruled out that this effect might be due to the unequal number of labels in both tasks. That adding the annotated features does not bring about better results was also shown for just the cooking task using graphical and inferential statistical analysis.

In **H2.1**, I compared transformers only being fed with textual input versus those that also receive categorical and numerical input. The result here was that the former perform either better or on par with the latter. This is highly dependent on the random state, which comes as no surprise since there are very few samples for some labels. I elaborated on this in section 3.3. The findings also hold for both classification tasks.

**H3.1** tried to grasp whether BERT is still the best performing topology among the ever-growing range of transformers. I found some circumstantial evidence of this but nothing to make any reliable statements on the hypothesis. Therefore, I have neither rejected nor confirmed **H3.1**.

I both confirmed **H4.1** and **H5.1**, proofing that off the shelf models can outperform hand-tailored neural networks and ML classifiers. This also proves the power of transfer learning and pre-training, which I explained in section 4.1.1 . Certainly, one could improve results by driving the use of those two concepts even further, for instance, by training models from scratch for certain domains. I will come to that shortly in the last part of my thesis, where I will give some thoughts on how my experiments and analysis can be improved.

## 6.3 Some more findings

In the previous part, I quickly went into the differences between the two pre-trained GPT2 models you can see in table 5.2. One was trained from scratch on a classical large text corpus. The other GPT2 model from this table is the same as the first but already fine-tuned on Goethe's Faust. I have already gone into what my reasoning here was. There might be some merit to using the second model over the first on the cooking task. A look into this revealed mean F1 scores for both models being virtually equal. My reasoning that the big corpus might engrain a general knowledge of language into the model while Faust gives it an understanding of human dialogue was obviously to far fetched. Also, fine-tuning a model twice does not seem to be something by which one can accumulate knowledge. The idea that you could grab what you need for your final model from different small corpora - what cooking is here, what a human dialogue is there - and accumulate this in more than one fine-tuning process does not seem to work. I am not surprised by this since it was likely that signals will overlap or cancel each other out. However, I find it was worth a try since it could be tested easily here.

The focus of this analysis has been the cooking task and that the voting task served as a touchstone for its evaluation. Still, there are some insights into swing voters to be gathered, and I do not want to let this go under the radar entirely. I did not formulate any hypotheses since this would have gone far beyond the scope of this thesis.

First of all, what drives people and what is on their minds makes them decide whether to change their voting behaviour. This is reflected by the observation of text-only transformers outperforming their multimodal counterparts on the voting task. Regardless of the the pre-trained model used, variables like gender, age, or education did not improve classification but even added noise to the discerning signal and worsened results. Therefore, it is a legitimate deduction to say, the issues and problems people named in the two open-ended questions (see section 2.1 carried the signal for the classifiers.

Second, and this picks up on my first point, every classifier had a lot easier time classifying swing voters than non-swing voters. The best model achieved a .93 F1 score on the swing voter label. The one that did best on the no swing voter label only had a .84 F1 score. That again shows the signal lay within the issues stated in the open-ended questions because swing voters have more to complain about and therefore are better manageable for the classifier to recognize.

Third, following Meidinger and Aßenmacher (2021), I proofed pre-trained models like BERT can yield excellent results even on small survey data. In future research, it would be interesting to see how the models work on political science tasks on a large scale, i.e. with big data from sources like social media, legislation or party program corpora. In ML, results are always a function of the input data, especially their size.

# 7 Conclusion

## 7.1 Summary

I have given a summary of my analysis at the end of the previous part, so there is no need to do it again. What I want to do instead is revisit my research question. Let us recall it from section 3.1:

> *What influence do additional numerical and categorical features have in text classification tasks when using transformer models? Is there a difference between dialogue data and non-dialogue data? How do different transformer architectures compare with each other and with other ML models in such circumstances?*

As for the first part of the question, I found additional features to have little or even unwanted influence on classification results when using transformers. That can be due to the features or the transformers themselves, i.e., they are really doing better with textual input.

The second part can be answered with yes, although not in the direction I anticipated. The annotations I had for the dialogue data improved results less than the non-dialogue data's empirical features. This hints that human conversation is more than just trying to model its flow as a sequence of stages.

The third subquestion comes out as anticipated. Off the shelf transformers generally performed better than all the other baseline models. Plus, they are easier to implement and need less data for deployment.

From a technical perspective, working with the huggingface `transformers` library was very smooth and exceptionally well documented. This is not so much true for the `multimodal-transformers` library, which is a comparably recent release. I also hope huggingface's models library, where people and institutions can upload their from-scratch-trained models, will grow at its present rate. At the time of this writing, all the 6500 pre-trained models available were uploaded or updated within the last year.

## 7.2 Reflection

One is always wiser after the event. With the benefit of hindsight, there are some things in this study, as probably in every study, I would advise my past self to approach differently next time.

1. Get more data sets for comparison. Survey data is easy to come by. Most German survey data is available from GESIS, the Leibniz Insitute for Social Science. It bundles pretty much every political survey conducted in Germany. It would still be quite a bit of work to research the appropriate survey data there and include it into the experiments, but I think it would be worth it. Especially for getting a sensible 13 label non-dialogical classification task with enough samples, survey data is probably a good way to go.
2. For space reasons, I did not investigate single labels or single samples but only looked at weighted F1 scores. Such a close look was not my primary aim either. I also explicitly mitigated the effect that one label or sample has an overwhelming influence on the F1 scores in the long run. I described this in 3.3. Nonetheless, it would certainly be worthwhile for future research. I will elaborate on this in the next section a bit.
3. Many ML studies make demands for more data in the end. In the case of this study, it is also a valid call. Of course, the aim was also to see how transformers perform on little data. However, in the long run, when going beyond these 13 labels in the cooking task, there is no way of getting around a demand for more data. That is so because, in a real guided cooking assistant, these 13 labels are only just the very beginning. In a real-world setting, you would probably have hundreds if not thousands of ways an assistant can

interpret user input. All these ways need to be present in the data. Otherwise, no model, however good, will be of help. I have some ideas on gathering extensive data on cooking dialogues without conducting more experiments like the one I described in 2.1.

## 7.3 Outlook

What could one do next? There are three major departures from this study. A straightforward step would be to examine why performance over random states differed so much in some experiments. One would probably first look into which samples landed in the test set of the very sparsely populated labels. That is different for different random states. Then you could qualitatively or with linguistic tools compare these sentences. What distinguishes them on a qualitative level and how could that explain the varying performance I have observed?

As a second path forward, there is more investigation into how the multimodal transformers work. I explained their working in 5.3. However, there are certainly a lot more and better ways to concatenate textual and other features. As of now, there was only one method available from the library I used. Adjusting this method or implementing your own would be worth looking into. Picking up on this, one could also look into how random vectors as additional features compare with those I used for this study. If the same results emerge, it would indicate that the annotations are only noise to the classifiers.

Another idea would be to train a BERT from scratch yourself on data from the cooking domain. I also think it would be possible in reasonable time with the four GPUs I used. Of course, this would involve scraping a lot of cooking-related textual data from somewhere. What comes to my mind first here is websites like *Chefkoch.de* and transcripts from cooking television shows. These shows often follow a pattern. An experienced cook does the cooking and explains what he or she is doing to the show's host. In other formats, there are several candidates and the cook keeps asking what they are trying to cook. In any case, they are dialogical cooking data. Also, using TV commercial shows on cooking equipment would be thinkable. You can find videos like these on the internet, especially on YouTube, where German subtitles are often available. If you can find enough video material and scrape the subtitles, you can gather cooking dialogues very quickly. It is only an idea that I would find interesting, and the cooking show transcript data would certainly not be perfect. Nevertheless, enlarging the corpus I used here is very expansive and for training from scratch, you need a lot of data. Here is an example from the German show *Die Küchenschlacht* (ZDF 2015):

> **Showmaster**: Äh Hähnchenbrust, Mango und Feldsalat, wie machste die? (engl. Uh, chicken breast, mango and lamb's lettuce, how do you prepare those? *Translation P.S.*)
>
> **Candidate**: Die Hähnchenbrust? Ähm, Salz und Pfeffer. Die brat ich erstmal an, schneid sie dann in Streifen und in der Zwischenzeit mach' ich die Marinade vom Feldsalat fertig. (engl. The chicken breast? Um, salt and pepper. I fry it first, then cut it into strips and in the meantime I make the marinade for the lamb's lettuce. *Translation P.S.*)

That is undoubtedly closer to the domain and type of text input we saw in this thesis than Wikipedia or some news crawls dump. There is also a realistic possibility to crawl a decent amount of such data.

At the risk of stating only the obvious, but there is a myriad of experimental research that needs to be done in neural NLP. There has to be a way to finally get a grip on these models, maybe even some general theory of neural networks and their behaviour. The way to get there, I think, is painstaking experimentation. Before Newton came up with some laws of gravity and before Einstein could refine them, millions of researchers over centuries carried out experiments on this one single subject, gravity. Theory builds on experimentation and experimentation builds on theory; it is a virtuous cycle, self-amplifying, hopefully leading to a better understanding of the world. However - to give a piece of personal opinion in the end - I think a theory has to be the ultimate goal. Experimentation is only self-serving up to some point. We might go on and on developing ever better ML models, but without a concise theory, we will lose track sooner or later. When at some point, you have $2^{20}$ algorithms for text classification, you need a grander theoretical scaffolding to guide you. Otherwise, no human in a lifetime will be able to sort her way through all of them and make a well-founded decision on which to use.

# References

Aaronovitch, David (2009). “Can Tories Win over Holby Woman?” In: *The Times*. URL: https://www.thetimes.co.uk/article/can-tories-win-over-holby-woman-km29kb69zc5 (visited on 03/09/2021).

Abdi, Herve and Lynne Williams (2010). “Honestly Significant Difference (HSD) Test”. In: Salkind, Neil. *Encyclopedia of Research Design*. SAGE Publications. DOI: 10.4135/9781412961288.n181.

Aggarwal, Charu C. (2018). *Machine Learning for Text*. Springer International Publishing. DOI: 10.1007/978-3-319-73531-3.

Alammar, Jay (2018). *The Illustrated Transformer*. URL: http://jalammar.github.io/illustrated-transformer/ (visited on 12/23/2020).

— (2019). *The Illustrated GPT-2*. URL: https://jalammar.github.io/illustrated-gpt2/#part-2-illustrated-self-attention (visited on 03/13/2021).

Amanova, Dilafruz, Volha Petukhova, and Dietrich Klakow (2016). “Creating Annotated Dialogue Resources: Cross-Domain Dialogue Act Classification”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*. European Language Resources Association (ELRA), pp. 111–117. URL: https://www.aclweb.org/anthology/L16-1017.

Backhaus, Klaus et al. (2018). “Diskriminanzanalyse”. In: *Multivariate Analysemethoden*. Springer, pp. 203–266. DOI: 10.1007/978-3-662-56655-8_5.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2016). *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv: 1409.0473 [cs, stat]. URL: http://arxiv.org/abs/1409.0473 (visited on 03/03/2021).

Bansal, Ashish (2021). *Advanced Natural Language Processing with TensorFlow 2 : Build Effective Real-World NLP Applications Using NER, RNNs, Seq2seq Models, Transformers, and More*. Packt Publishing.

Bayerischer Landtag, Bayerischer Landtag (2021). *Wahlsystem*. URL: https://www.bayern.landtag.de/abgeordnete/landtagswahl-2018-1/ (visited on 03/11/2021).

Bojanowski, Piotr et al. (2017). *Enriching Word Vectors with Subword Information*. arXiv: 1607.04606 [cs]. URL: http://arxiv.org/abs/1607.04606 (visited on 11/26/2020).

Brenke, Karl and Alexander S. Kritikos (2020). “Wohin Die Wählerschaft Bei Der Bundestagswahl 2017 Wanderte”. In: *DIW Wochenbericht*. DOI: 10.18723/DIW_WB:2020-17-1.

Brownlee, Jason (June 16, 2019). *A Gentle Introduction to Generative Adversarial Networks (GANs)*. Machine Learning Mastery. URL: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (visited on 03/02/2021).

Campbell, Angus et al. (1960). *The American Voter*. University of Chicago Press.

Chan, Branden, Stefan Schweter, and Timo Möller (2020). *German's Next Language Model.* arXiv: `2010.10906 [cs]`. URL: `http://arxiv.org/abs/2010.10906` (visited on 03/01/2021).

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, pp. 1724–1734. DOI: `10.3115/v1/D14-1179`.

Chollet, François (2018). *Deep Learning with Python.* Shelter Island, New York: Manning Publications.

Clark, Kevin et al. (2020). *ELECTRA: Pre-Training Text Encoders as Discriminators Rather Than Generators.* arXiv: `2003.10555 [cs]`. URL: `http://arxiv.org/abs/2003.10555` (visited on 03/02/2021).

Devlin, Jacob et al. (2019). *BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.* arXiv: `1810.04805 [cs]`. URL: `http://arxiv.org/abs/1810.04805` (visited on 03/01/2021).

Dickson, Hamish (2019). *CNN for Sentence Classification by Yoon Kim.* URL: `https://kaggle.com/hamishdickson/cnn-for-sentence-classification-by-yoon-kim` (visited on 03/13/2021).

Field, Andy P. and James Iles (2016). *An Adventure in Statistics: The Reality Enigma.* SAGE Publications.

Frummet, Alexander (2019). *CookversationalSearch.* GitHub. URL: `https://github.com/AlexFrummet/CookversationalSearch` (visited on 03/09/2021).

Frummet, Alexander, David Elsweiler, and Bernd Ludwig (2019). "Detecting Domain-Specific Information Needs in Conversational Search Dialogues". In: *Proceedings of the 3rd Workshop on Natural Language for Artificial Intelligence.* DOI: `10.5283/epub.41138`.

Georgian (2021). *Multimodal-Toolkit.* Georgian. URL: `https://github.com/georgian-io/Multimodal-Toolkit` (visited on 03/10/2021).

Gillioz, Anthony et al. (2020). "Overview of the Transformer-Based Models for NLP Tasks". In: *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems*, pp. 179–183. DOI: `10.15439/2020F20`. URL: `https://annals-csis.org/proceedings/2020/drp/20.html` (visited on 03/10/2021).

Goldberg, Yoav (2017). *Neural Network Methods for Natural Language Processing.* Morgan & Claypool.

Gu, Ken (2020). *How to Incorporate Tabular Data with HuggingFace Transformers.* Medium. URL: `https://medium.com/georgian-impact-blog/how-to-incorporate-tabular-data-with-huggingface-transformers-b70ac45fcfb4` (visited on 03/13/2021).

Hagiwara, Masato (2021). *Real-World Natural Language Processing: Practical Applications with Deep Learning.* O'REILLY.

Han, Ler Wei (2019). *Manipulating Machine Learning Results with Random State.* Medium. URL: `https://towardsdatascience.com/manipulating-machine-learning-results-with-random-state-2a6f49b31081` (visited on 03/02/2021).

Haynes, Winston (2013). “Tukey’s Test”. In: *Encyclopedia of Systems Biology.* Ed. by Werner Dubitzky et al. Springer, pp. 2303–2304. DOI: 10.1007/978-1-4419-9863-7_1212.

Heinrich, Tassilo and Melanie Walter-Rogg, eds. (in appearance). *The State Election 2018 in Bavaria – Analyses of Electoral Behavior and Political Culture in the Free State.* Springer.

Henkel, Steffi (2007). *Entwicklung von Assistenzkonzepten Unter Verschieden Ressourcenreichen Bedingungen.* URL: http://www.prometei.de/fileadmin/prometei.de/publikationen/Henkel_FSP8_Diplomarbeit.pdf.

Hoover, Benjamin, Hendrik Strobelt, and Sebastian Gehrmann (2020). “exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations.* Association for Computational Linguistics, pp. 187–196. DOI: 10.18653/v1/2020.acl-demos.22.

Huggingface (2021a). *Pretrained Models — Transformers 4.3.0 Documentation.* URL: https://huggingface.co/transformers/pretrained_models.html (visited on 03/04/2021).

— (2021b). *Summary of the Tokenizers — Transformers 4.3.0 Documentation.* URL: https://huggingface.co/transformers/tokenizer_summary.html (visited on 03/08/2021).

Jabbar, Haris (2020). “Chapter 13 Natural Language Generation”. In: *Modern Approaches in Natural Language Processing.* URL: https://compstat-lmu.github.io/seminar_nlp_ss20/ (visited on 07/03/2021).

Jiang, Zihang et al. (2021). *ConvBERT: Improving BERT with Span-Based Dynamic Convolution.* arXiv: 2008.02496 [cs]. URL: http://arxiv.org/abs/2008.02496 (visited on 03/13/2021).

Joulin, Armand et al. (2016). *Bag of Tricks for Efficient Text Classification.* arXiv: 1607.01759 [cs]. URL: http://arxiv.org/abs/1607.01759 (visited on 12/03/2020).

Jurafsky, Dan and James H. Martin (2020). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* 3. ed., draft. URL: https://web.stanford.edu/~jurafsky/slp3/.

Kedia, Aman and Mayank Rasu (2020). *Hands-On Python Natural Language Processing.* Packt Publishing.

Kim, Yoon (2014). “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, pp. 1746–1751. DOI: 10.3115/v1/D14-1181.

Lample, Guillaume and Alexis Conneau (2019). *Cross-Lingual Language Model Pretraining.* arXiv: 1901.07291 [cs]. URL: http://arxiv.org/abs/1901.07291 (visited on 03/13/2021).

Lipset, Seymour Martin and Stein Rokkan, eds. (1967). *Cleavage Structures, Party Systems, and Voter Alignments. An Introduction.* Free Press.

López, Daniel (2017). *Is Random State a Parameter to Tune?* URL: https://stats.stackexchange.com/q/264008.

Malchanau, Andrei (2019). “Cognitive Architecture of Multimodal Multidimensional Dialogue Management”. DOI: 10.22028/D291-27856.

McCormick, Chris, director (2019a). *BERT Research - Ep. 1 - Key Concepts & Sources.* URL: `https://www.youtube.com/watch?v=FK1PCK1uFrc&list=PLam9sigHPGwOBuH4_4fr-XvDbe5uneaf6` (visited on 01/25/2021).

— (2019b). *BERT Research Series.* URL: `https://www.youtube.com/playlist?list=PLam9sigHPGwOBuH4_4fr-XvDbe5uneaf6` (visited on 03/05/2021).

— (2020). *The Inner Workings of BERT.* URL: `www.chrismccormick.ai`.

Meidinger, Maximilian and Matthias Aßenmacher (2021). "A New Benchmark for NLP in Social Sciences: Evaluating the Usefulness of Pre-Trained Language Models for Classifying Open-Ended Survey Responses:" in: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence.* SCITEPRESS, pp. 866–873. DOI: `10.5220/0010255108660873`.

Mikolov, Tomas et al. (2013). *Efficient Estimation of Word Representations in Vector Space.* arXiv: `1301.3781 [cs]`. URL: `http://arxiv.org/abs/1301.3781` (visited on 11/28/2020).

Mitchell, Melanie (2019). *Artificial Intelligence: A Guide for Thinking Humans.* Penguin Books.

Mrkšić, Nikola et al. (2017). *Neural Belief Tracker: Data-Driven Dialogue State Tracking.* arXiv: `1606.03777 [cs]`. URL: `http://arxiv.org/abs/1606.03777` (visited on 03/10/2021).

Papakyriakopoulos, Orestis et al. (2017). "Social Media und Microtargeting in Deutschland". In: *Informatik-Spektrum* 40.4, pp. 327–335. DOI: `10.1007/s00287-017-1051-4`.

Pappi, Franz Urban (2003). "Wählen als Akt expressiver Präferenzoffenbarung. Eine Anwendung der Conjoint-Analyse auf die Wahl zur Hamburger Bürgerschaft am 21. September 1997". In: *Politische Vierteljahresschrift* 44.1, pp. 110–111. DOI: `10.1007/s11615-003-0018-z`.

Radford, Alec et al. (2018). "Language Models Are Unsupervised Multitask Learners". In: URL: `https://paperswithcode.com/paper/language-models-are-unsupervised-multitask`.

Raschka, Sebastian and Vahid Mirjalili (2017). *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow.* Packt Publishing.

Rather, Mudasir (2018). "Information Needs of Users in the Tech Savvy Environment and the Influencing Factors". In: *Encyclopedia of Information Science and Technology.* IGI Global. DOI: `10.4018/978-1-5225-2255-3.ch197`.

Rothman, Denis (2021). *Transformers for Natural Language Processing Build Innovative Deep Neural Network Architectures for NLP with Python, Pytorch, TensorFlow, BERT, RoBERTa, and More.* Packt Publishing.

Sanh, Victor et al. (2020). *DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter.* arXiv: `1910.01108 [cs]`. URL: `http://arxiv.org/abs/1910.01108` (visited on 03/13/2021).

Schoen, Harald (2014). "Wechselwahl". In: *Handbuch Wahlforschung.* Ed. by Jürgen W. Falter and Harald Schoen. Springer, pp. 489–522. DOI: `10.1007/978-3-658-08240-6_11`.

Schoen, Harald and Cornelia Weins (2014). “Der sozialpsychologische Ansatz zur Erkärung von Wahlverhalten”. In: *Handbuch Wahlforschung*. Ed. by Jürgen W. Falter and Harald Schoen. Springer, pp. 241–329. DOI: `10.1007/978-3-658-08240-6_7`.

Schuster, Mike and Kaisuke Nakajima (Mar. 2012). “Japanese and Korean Voice Search”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5149–5152. DOI: `10.1109/ICASSP.2012.6289079`.

See, Abigail, director (2019). *NLP with Deep Learning. Lecture 8 – Translation, Seq2Seq, Attention*. URL: `https://www.youtube.com/watch?v=XXtpJxZBa2c&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z&index=8` (visited on 03/12/2021).

Spectator (2019). *Do ‘Workington Man’ and ‘Worcester Woman’ Decide Elections?* URL: `https://www.spectator.co.uk/article/do-workington-man-and-worcester-woman-decide-elections-` (visited on 03/09/2021).

Stifel, Andreas (2018). *Vom erfolgreichen Scheitern einer Bewegung: Bündnis 90/Die Grünen als politische Partei und soziokulturelles Phänomen*. Springer.

Stöss, Richard (1997). *Stabilität im Umbruch*. Springer. DOI: `10.1007/978-3-322-85109-3`.

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. arXiv: `1706.03762 [cs]`. URL: `http://arxiv.org/abs/1706.03762` (visited on 03/03/2021).

videocardbenchmark.net (2021). *Video Card (GPU) Benchmark Charts - Model List*. URL: `https://www.videocardbenchmark.net/gpu_list.php` (visited on 03/16/2021).

Vig, Jesse (2019). *Visualizing Attention in Transformer-Based Language Representation Models*. arXiv: `1904.02679 [cs, stat]`. URL: `http://arxiv.org/abs/1904.02679` (visited on 03/19/2021).

— (2021). *Deconstructing BERT, Part 2: Visualizing the Inner Workings of Attention*. Medium. URL: `https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1` (visited on 03/02/2021).

Vlasov, Vladimir, Johannes E. M. Mosig, and Alan Nichol (2020). *Dialogue Transformers*. arXiv: `1910.00486 [cs]`. URL: `http://arxiv.org/abs/1910.00486` (visited on 11/26/2020).

Wagner, Joshua (2020). “Chapter 8 Attention and Self-Attention for NLP”. In: *Modern Approaches in Natural Language Processing*. URL: `https://compstat-lmu.github.io/seminar_nlp_ss20/` (visited on 07/03/2021).

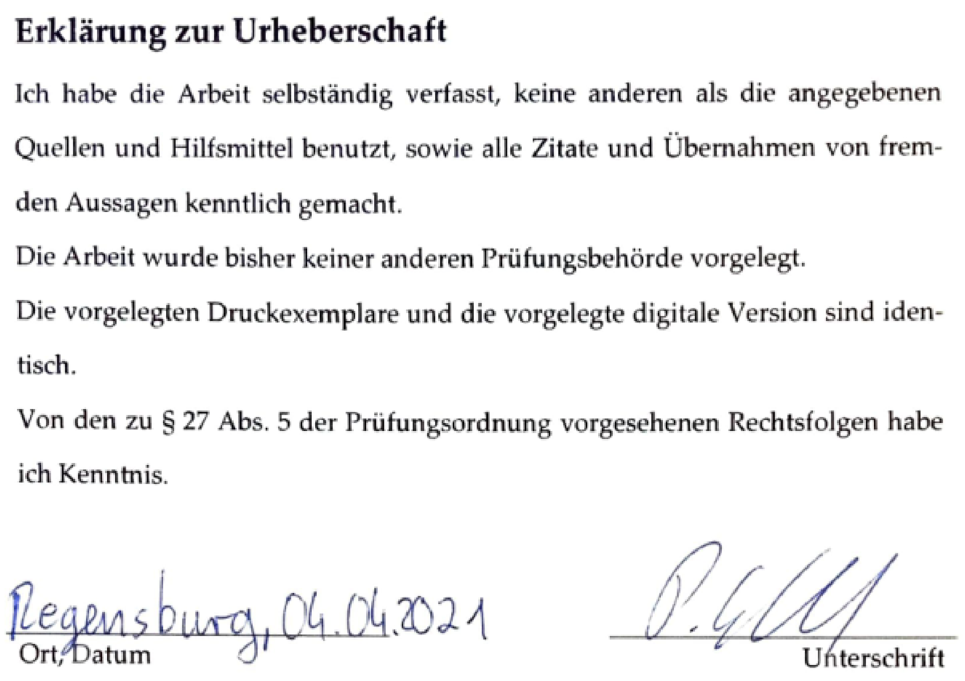
Wandke, Hartmut (Mar. 2005). “Assistance in Human–Machine Interaction: A Conceptual Framework and a Proposal for a Taxonomy”. In: *Theoretical Issues in Ergonomics Science* 6.2, pp. 129–155. DOI: `10.1080/1463922042000295669`.

Westle, Bettina and Markus Tausendpfund, eds. (2019). *Politisches Wissen: Relevanz, Messung Und Befunde*. Politisches Wissen. Springer. DOI: `10.1007/978-3-658-23787-5`.

Wikipedia (2020). *Information Needs*. In: URL: `https://en.wikipedia.org/w/index.php?title=Information_needs&oldid=969535804` (visited on 03/10/2021).

Winograd, Terry (1972). “Understanding Natural Language”. In: *Cognitive Psychology* 3.1, pp. 1–191. DOI: `10.1016/0010-0285(72)90002-3`.

ZDF, director (2015). *Die Küchenschlacht (ZDF, 26.06.2015)*. URL: `https://www.youtube.com/watch?v=O2a1Pg99ymo` (visited on 03/16/2021).



## Erklärung zur Urheberschaft

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate und Übernahmen von fremden Aussagen kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Die vorgelegten Druckexemplare und die vorgelegte digitale Version sind identisch.

Von den zu § 27 Abs. 5 der Prüfungsordnung vorgesehenen Rechtsfolgen habe ich Kenntnis.

| | |
|---|---|
| Regensburg, 04.04.2021 | [illegible] |
| Ort, Datum | Unterschrift |