Master Thesis

Enhancing Stance Prediction by Utilizing Party Manifestos

Author

Lea Schulz-Vanheyden

Supervisor

Matthias Aßenmacher
Department of Statistics

Department of Statistics

Ludwig-Maximilians-Universität München

Munich, 8th of May 2023

Abstract

Recent advancements in Natural Language Processing (NLP) have wide-ranging implications, including in politics, where language models can be used to give voters useful information as they navigate a complicated political landscape. For example, a model could assist users by predicting which parties align with statements chosen by the user, aiding in informed decision-making. This thesis explores the effectiveness of incorporating information from political party manifestos to enhance the accuracy of large language models that predict party stances. The proposed model takes user-generated statements and relevant passages from party manifestos as inputs to determine the political party that is most likely to agree with the statement. The thesis evaluates the performance of different semantic search techniques for identifying relevant passages and various input patterns for conveying the additional information to the model. Two NLP models, ELECTRA and BERT, are compared to determine their effectiveness in this task. The study found that incorporating information from party manifestos did not substantially improve the model's ability to predict party stances, although some semantic search techniques provide better results than others. Further research is needed to determine the effectiveness of different input patterns, models, and hyperparameters, and to investigate how various other context types can be incorporated into the model.

Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
ELECTRA	Efficiently Learning an Encoder that Classifies Token Replacements Accurately
ELMo	Embeddings from Language Models
GAN	Generative Adversarial Network
GPT	Generative Pretrained Transformer
GRU	Gated Recurring Unit
LM	Language Model
LLM	Large Language Model
LSTM	Long Short-Term Memory
MLM	Masked Language Model
NLP	Natural Language Processing
NN	Neural Network
NSP	Next Sentence Prediction
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RTD	Replaced Token Detection
SBERT	Sentence-BERT

1. Introduction

Recent years have seen a rapid advancement in the field of Natural Language Processing (NLP), largely because of developments in neural networks, deep learning methods and availability of computing power. The ability of machines to accurately decipher and process human language, one of the most intricate and nuanced forms of communication, makes NLP an important field of study in machine learning. Due to the potential of NLP, there has been a significant investment in research and development from businesses like Google, Microsoft, and Facebook (Tech Advisor, 2019). In the years to come, how we communicate, work, and live will likely be significantly impacted by the continued development of NLP models and techniques.

There has been a lot of interest in using NLP models in all kinds of areas, including politics and education. NLP models can (among other things) be used to spot political leaning, forecast election results, and measure public opinion by examining political speeches, social media posts, and news articles. Sharma and Moh (2016) for example use sentiment analysis on Hindi Twitter in order to predict Indian elections. Recurrent Neural Networks (RNNs) can be used to detect the political position expressed in a sentence (Iyyer et al., 2014). Bradley Hayes trained a Twitter bot on speeches of Trump which is able to build tweets that sound like they were written by Trump (Murphy, 2016) and Hartmann et al. (2023) uncovers that the language model ChatGPT (OpenAI, 2022) has a pro-environmental, left-libertarian orientation, acknowledging that no model can be truly neutral.

These are just some examples of how NLP models can be used in the area of politics. While some of them, like the Trump-bot, are purely entertaining, others aim to solve problems like predicting an election or uncovering biases. Another use case is to use NLP techniques in order to aid human decision-making in regards to politics. Deciding which party to vote for is often not an easy decision. In an increasingly complex world, the opinions and views of each party are also more difficult to grasp and understand. Existing decision aids like the Wahl-O-Mat (Bundeszentrale für politische Bildung, 2021) use predefined statements to compare the parties' views. Recent advances, however,

suggest that automated interfaces based on large-scale language models could be used to create decision aids that are more human-like and effective.

A model that analyzes user-generated statements and forecasts whether each of the parties will agree with the statements is one way to accomplish this goal. Users of this model are able to gain insight into the positions and strategies of each party on a specific issue chosen by them, which enables them to make well-informed decisions on subjects that are relevant to them. Such a tool needs an advanced NLP model that can correctly predict party positions based on a relatively short input sentence from the user. To give more context to the model, it might be useful to also feed it relevant passages from the party manifestos.

This thesis extends Witte et al. (2022), where a model was fitted with the same goal. But, unlike in this thesis, they did not use the party manifestos as additional context. The aim of this thesis is to investigate whether party stance prediction can be improved by utilizing the information contained in party manifestos. The investigation is structured in three parts: (a) the exploration of several semantic search methods to find relevant passages in the manifestos, (b) the examination of optimal ways to communicate this additional information to the model (input pattern design), and (c) the comparison of performance between two big language models, ELECTRA (Clark et al., 2020) and BERT (Devlin et al., 2018). By conducting this investigation, the thesis seeks to determine which approach is most effective for enhancing party stance prediction.

This work is structured in the following way: Chapter 2 introduces the problem setting and the data in detail and provides some relevant background information. In Chapter 3 all the models and methods used are explained. There will be a general introduction to transformers and a brief overview of the most important NLP models of the last few years. The different semantic search techniques later used are described, and the basics and different approaches to input pattern design are introduced. Chapter 4 provides details on the experiments that were conducted, while in Chapter 5 the results are described and analyzed. These are further discussed in Chapter 6. Chapter 6 also gives an outlook on further research by discussing the limitations and possible new directions. Chapter 7 draws the final conclusions of this thesis.

2. Data and Problem Statement

Before delving into the details of the study, it is relevant to provide some context on the political landscape and the tools used to help voters understand party stances in Germany. The following sections provide a brief overview of the different parties and ideologies present in the German political landscape, as well as an introduction to the Wahl-O-Mat platform. This is followed by a discussion of the problem statement and the data used for the study.

2.1. Political Landscape in Germany

Germany is a federal state, and apart from the federal government, each of the 16 individual states hold political power. The states are responsible for certain areas, such as education, culture, and police law. There is a wide variety of parties in Germany, some of them appearing nationwide and others only in certain states. For example, there is the CSU (Christlich-Soziale Union in Bayern), which only competes in Bavaria. Together with the CDU (Christlich Demokratische Union Deutschlands), which competes everywhere but Bavaria, they form the so-called *Union*. Both parties refrain from competing in their respective other federal states and work together in the German Bundestag.
The Bundestag is Germany's parliament and the country's main legislative body. Its members are elected every 4 years, with the most recent election in 2021. Right now, there are 7 parties in the Bundestag: the *SPD* (Sozialdemokratische Partei Deutschlands) has the majority of seats with 206 out of 734. The *Union* (consisting of *CDU* and *CSU*) takes up 197 seats, the party *Bündnis 90/Die Grünen* 118, the *FDP* (Freie Demokratische Partei) 92, the *AfD* (Alternative für Deutschland) 81, the *die Linke* 39, and a small regional minority party named *Südschleswigscher Wählerverband* (SSW) takes up the last seat (Wikipedia, 2022c).
In this thesis (and in the work it builds upon), the SSW is ignored since it is a small and very regional party with no real decision-making power in the Bundestag. The CDU and CSU are treated as belonging to one group, the Union.

As the goal is to predict party stances, it is useful to have at least a basic understanding of the parties goals and politics.
The *SPD* emerged from labor parties and, as their successor, sees itself as a promoter of greater social justice. They call for the strengthening of the social market economy (in German: "Soziale Marktwirtschaft") and want an ecological and employee-friendly restructuring of society (Schubert and Klein, 2016). It could be described as center-left in the political landscape.
The CDU supports a free-market economy and is rather conservative on social issues but still supports social welfare programs. "The CSU is more conservative than the CDU, especially on social issues such as abortion, church-state matters, immigration, and the rights of Germany's many foreign residents." (Conradt, 2013). The *Union* is placed in the center-right on the political spectrum (Conradt, 2021).
Die Grünen (English: "the Greens") are, as the name suggests, an environmental party. They are concerned with ecological, economic, and social sustainability and work on issues such as the promotion of renewable energies, nuclear phase-out, agricultural turnaround, and restructuring of the tax system (Decker, 2021).
The *FDP* is an economically liberal party. It advocates less control by the state and more personal responsibility on the citizen's part. This includes, for example, the abolition of the minimum wage, the reduction of the broadcasting contribution, and simpler tax laws (Wikipedia, 2022a).
The *AfD* is a far-right party founded in 2013. According to BBC (2020), while founded as an anti-euro party, the AfD now focuses on fighting immigration and Islam with some of its members trivializing the Holocaust and being observed by the "Verfassungsschutz" (German domestic intelligence services).
Die Linke (English: "the Left") is furthest left of all the parties currently in the Bundestag. Its main focus lies on democratic socialism as an alternative to capitalism. As such, they want to increase rates for middle- and high-income taxes and spend more public money on education, research, culture and infrastructure. (Wikipedia, 2022b).

2.2. Wahl-O-Mat

The German "Wahl-O-Mat" (Bundeszentrale für politische Bildung, 2021) is a question-and-answer tool that is supposed to aid users in figuring out which political party registered for an election is closest to their own political position. A separate Wahl-O-Mat is developed for each election and posted on the internet a few weeks before the

polls open.
Marschall (2022) describes how it works: “The Wahl-O-Mat presents 38 propositions to its users, for example, ‘All public buildings should by law become non-smoking areas’ or ‘The VAT must be raised’. People visiting the website are asked to take a stand on the propositions by clicking one of three buttons: ‘agree’, ‘disagree’ or ‘neutral’. Additionally, users have the option of skipping several theses. After voting on all items on the list, the users can mark those propositions, that they consider important to them, to give them a special weight in the final calculation. Finally, the Wahl-O-Mat calculates the distances between the voter’s and the parties’ positions and displays, as the result, the party with the smallest distance - the best match. Additionally, the Wahl-O-Mat calculates the extent of agreement between the voter’s position and all parties by displaying the respective summed distances for all parties selected. Furthermore, for each proposition the users have the option to take a closer look at the relation between the parties’ positions and their own point of view. Finally, Wahl-O-Mat players can have a look at any explanation the parties may have provided concerning the positions they have taken on these propositions.”
In addition to providing information about the essential beliefs of the parties, the Wahl-O-Mat serves as an instrument for promoting political communication. Follow-up communication as an exchange (especially in social groups such as school, family,or workplace) can contribute to the formation of political opinion before elections (Rohmann, 2009). Far beyond the specific election ballot, the aim is to promote engagement with politics.
And it achieves its goal. The first Wahl-O-Mat was published in 2002 and became popular right away; the initial iteration was used about 3.6 million times, and the highest usage was achieved prior to the 2021 German Federal Election when it was used about 21.3 million times (Marschall, 2022).

2.3. Problem

While the Wahl-O-Mat (Bundeszentrale für politische Bildung, 2021) is a very useful tool for making politics tangible, get people interested in it, and also help them check which parties represent their interests, it has one major drawback. The statements are pre-defined. It is not possible for users to introduce topics they are interested in. This is exactly the problem that language models could help solve. NLP is now a crucial tool for deciphering and analyzing user-generated content across a variety of fields, and

it could also be used for political stance prediction. Thus, the goal of this study is to investigate the use of language models and NLP methods to overcome this limitation of the Wahl-O-Mat and examine the accuracy of political stance prediction by incorporating context from the party manifestos.

The goal of this thesis is to develop a classifier that predicts party stances, similar to the Wahl-O-Mat. Specifically, the classifier should predict how the six parties currently represented in the Bundestag would agree with a user-generated Wahl-O-Mat-style statement (query). While Witte et al. (2022) have already trained a similar model, this thesis aims to investigate whether incorporating party manifestos can further improve the model performance.

Since the manifestos contain information about the parties stances, being able to incorporate them into the model should (in theory) boost the performance. The first step is to extract the relevant passages from the manifestos. Since they are too long to be fed to the model as a whole, it is necessary to find the parts that relate to the user query. In this thesis, a couple of so-called semantic search techniques are described and applied to the data in order to find the sentences that contain the most semantic similarity (i.e. similar meaning) with the query. There is currently a lot of research on improving existing methods and focusing on new approaches in semantic search. It is hard to differentiate between what works and brings an improvement and what does not, especially since most of the models are tested on English data, but the language of the data used in this thesis is German. A range of ideas are tested in an effort to find the best strategy. Four newer approaches are compared with two established ones.

The second issue to address and explore is the "input pattern design". Input pattern design refers to different combinations of the parts of the input (party, query, and relevant sentences from the manifesto) that are fed into the model. It is not clear what the optimal way to do this is. In order to find out, four different patterns (and two variations) are compared with each other and also to a baseline of no context (i.e. no party manifestos, just the party and the query). Some of these patterns are inspired by human-speech-like language, while others are more structured in nature. The thesis wants to explore if adding context (the party manifestos) improves the models predictions and if so which input pattern design works best.

The last choice to be made is the model. There are a lot of NLP models that could be used for this task, and it is not straightforward which one to choose. The combinations of semantic search technique and input pattern design are applied to two models in order to be able to choose the best one. It is also of interest if some combinations perform

better depending on the model.
This research could be useful in a number of ways. If political stances in user-generated content can be accurately classified, this would allow people to determine which political parties correspond to their own beliefs and values. Citizens can get a thorough understanding of the political environment and are able to make an informed voting decision. Thus giving people a useful tool to participate in democracy.
Second, the research surrounding language models can be advanced. The use of these methods in a new and crucial area, political prediction, can be investigated. By using party manifestos as a source of context, it is possible to show how useful domain-specific knowledge is for enhancing the performance of NLP models.

2.4. Data

2.4.1. User-generated Statements

The data was collected from five different sources by Witte et al. (2022) shown in Figure 2.1:

1. *Voting Advice Application*: The Wahl-O-Mat (Bundeszentrale für politische Bildung, 2021) provides 1809 political statements for the period between 2002 and 2021. For all statements, there is an approval label for each of the six parties: "SPD", "Union", "Die Grüne", "FDP", "AfD" and "Die Linke".

2. *Party Short Manifestos*: To extend this data, 801 manually extracted statements from the 2021 party short manifestos were added. Unlike the Wahl-O-Mat data, these statements only have one label (that of the corresponding party).

3. *User Generated Texts*: 3737 statements were user-generated. Participants were asked for political statements and which parties would support those statements the most and the least (in their personal opinion). This corresponds to two labels (one positive and one negative).

   At that point, every statement in the data set is politically controversial, meaning that each statement has parties that agree and those that disagree. Users may also be drawn to issues that are truly uncontroversial among the parties, such as "Democracy should be protected". Recent studies have demonstrated the effectiveness of targeted data set enrichment as a method for enhancing model resilience and resolving

associated problems (Gupta et al., 2021; Bakhtin et al., 2022). In order to do this, a total of 281 uncontroversial statements were added with equal labels for all parties:

4. *UN Charta & Constitution*: 91 statements from the German constitution stating basic human rights and 48 statements from the United Nations Human Rights Charter.

5. *Adversarial Examples*: 142 manually created adversarial examples.

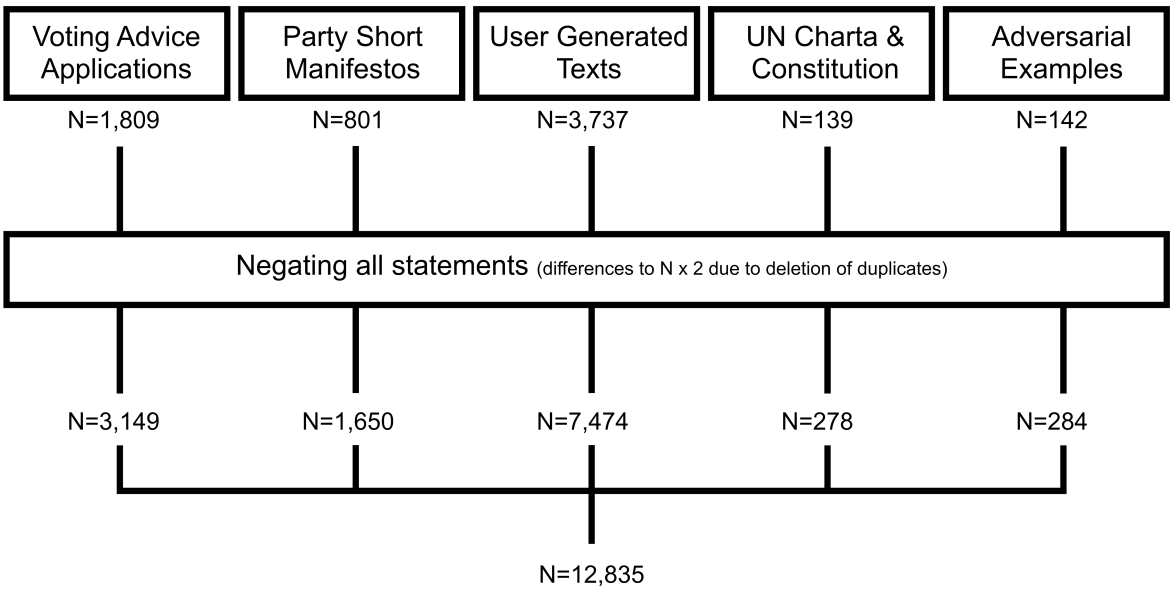

Figure 2.1.: The different data sources.

Another possible issue is that one word can change a sentence's meaning and, consequently, its labeling (for example, "Taxes should be raised" vs. "Taxes should not be raised"). Such negated phrases are generated automatically, and labels are assigned in accordance, bringing the total number of labeled statements in the final data set to 12835.

2.4.2. Party Manifestos

The data used in this study also consists of the official 2021 election programs for the Bundestag election in Germany. The party manifestos from the six already mentioned parties were downloaded and split into sentences. Note that, because the CSU is part of the Union parliamentary group at the federal level, the CSU statements were included in the CDU file to simplify the data collection process.

There are 1386 sentences in the SPD manifesto, 2602 in the CDU+CSU manifestos, 3689 in “die Grünen”, 2224 in the FDP, 1607 in the AfD, and 4511 in “die Linke”. As a result, there are 16019 sentences total in all party manifestos.

3. Methodological Background

This Chapter is divided into a number of sections that explore different facets of NLP relevant to this thesis. The first section provides a very brief overview of NLP in general and discusses subjects mentioned later like neural networks, language models, tokenization, and embeddings. The next section delves into the concept of transformers, which have revolutionized the field of NLP by enabling state-of-the-art performance on various language tasks. The discussion of pre-trained language models follows, which have gained popularity in recent years as a result of their outstanding results on a variety of language tasks. Several pre-trained language models, ELMo (Peters et al., 2018), GPT (Radford et al., 2018), BERT (Devlin et al., 2018), ELECTRA (Clark et al., 2020), and T5 (Xue et al., 2020), are introduced. The Chapter then covers various semantic search techniques, SBERT (Reimers and Gurevych, 2019), BERT-Flow (Li et al., 2020), Whitening (Huang et al., 2021), SBERT-WK (Wang and Kuo, 2020), and IS-BERT (Zhang et al., 2020), which are used to extract relevant information from texts. The Chapter ends with a section on input pattern design, which discusses how to format input for NLP models to produce predictions.

3.1. Natural Language Processing

NLP is a branch of Artificial Intelligence (AI) that focuses on instructing computers to understand, interpret, and generate human language. The fundamental ideas and methods of NLP, such as neural networks, tokenization, and embeddings, are briefly introduced in the sections that follow.

Neural Networks

Adapted from the design and function of the human brain, Neural Networks (NNs) are a specific kind of machine learning model. Neural networks are made up of interconnected layers of nodes that process and transform data. Each node receives input from the

previous layer's nodes and applies a non-linear function of the sum of its inputs to produce an output that is passed on to the nodes in the next layer. The weights of the connections between the nodes are adjusted during training to reduce the error between the predicted and true outputs. As a result, the NN can recognize patterns and relationships in the data and use them to accurately predict or categorize new data.

Language Models

Language models (LMs) are models that focus on comprehending and/or producing human language. To learn the statistical patterns and structures of language, LMs are trained on massive amounts of text data, such as books, articles, and web pages. A wide range of tasks, including text generation, machine translation, sentiment analysis, question-answering, and more, can be performed using LMs. They serve as a foundation for comprehending and processing human language in a way that closely resembles human cognition, making them a crucial component of many NLP applications.

Tokenization

Tokenization is the process of breaking down text into smaller units, called tokens, which can be individual words, subwords, or even special tokens such as end-of-sentence markers. Tokens are the basic building blocks of text and are used as input to various NLP algorithms. For instance, in the sentence "I love natural language processing", tokenization might result in tokens such as "I", "love", "natural", "language", and "processing". Tokens offer a way to represent text in a structured format that NLP algorithms can easily process and analyze. Tokenization is an essential step in many NLP tasks because it lays the groundwork for additional text analysis and processing.

Embeddings

Because NLP models need numerical representations to process the data, text data cannot be directly input into these models. Here is where embeddings come into play. Word embeddings, sentence embeddings, and document embeddings are numerical representations of text that capture the context and meaning of the text in a continuous vector space. In order for algorithms to effectively analyze, comprehend, and produce human language, these embeddings act as a link between the unstructured text data and the structured numerical data that NLP models can process. One key property of good

embeddings is that similar words should have embeddings that are close to each other in the vector space. For instance, words like "cat" and "dog" should be closer to one another in an embedding space than "cat" and "language". For many NLP tasks, the similarity or relatedness between words must be captured, and this proximity in the embedding space does just that.

3.2. Transformers

Moving forward to discuss an important LM architecture, let us delve into the concept of transformers. LMs, as previously stated, are machine learning models that seek to comprehend text in natural languages. Because natural language contains complex and long-term dependencies that go beyond the single word the model is looking at, they need some sort of memory to store and retrieve data in order to carry out this task. The models are only able to produce contextually appropriate responses by keeping track of previously read text. Memory also enables LMs to deal with ambiguity and uncertainty by taking various interpretations of a given text into account (given what has happened before). Without memory, LMs would struggle to capture the variety and complexity of human language.

RNNs (Recurrent Neural Networks) (Elman, 1990), LSTMs (Long Short-Term Memory) (Hochreiter and Schmidhuber, 1997), or GRUs (Gated Recurring Units) (Cho et al., 2014) are different ways to introduce some sort of artificial memory into a model.

The self-attention mechanism that is used for transformers in "Attention is all you need" (Vaswani et al., 2017) is another way to enable memory. Attention as a concept was first introduced by Bahdanau et al. (2014). Unlike other mechanisms, attention has (in theory) an infinite window of memory, i.e. it doesn't suffer from long-term memory loss. It enables the model to remember everything that might be important, no matter how long ago it was processed. The goal of attention is to find the relevant parts of the input. In order to, for example, translate a sentence, some words are more important than others for predicting the target word. The model needs to learn what these words are and pay more attention to them.

Following the architecture in Vaswani et al. (2017) the attention mechanism has three key components: the query q, the key k and the value v. An answer on Stackoverflow by the user dontloo (2019) gives a good analogy: "The key/value/query concept is analogous to retrieval systems. For example, when you search for videos on Youtube, the search engine will map your query (text in the search bar) against a set of keys (video title,

description, etc.) associated with candidate videos in their database, then present you the best matched videos (values)."

Since attention is calculated on a set of queries simultaneously, the vectors q, k and v become the matrices Q, K and V. The query and key matrices then undergo dot product matrix multiplication to produce a score matrix. This matrix has a score for each word pair that determines how much focus should be put on that word when looking at the other word. After scaling the matrix QK^T and using a softmax, it consists only of values between 0 and 1 that determine the amount of attention for every word. This attention matrix is then multiplied with the values V to get the output. The following matrix notation shows the full formula to get the embeddings using attention from Q, K and V (d_K is the dimension of K used for scaling):

$$\text{Attention(Q,K,V)} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right)V \tag{3.1}$$

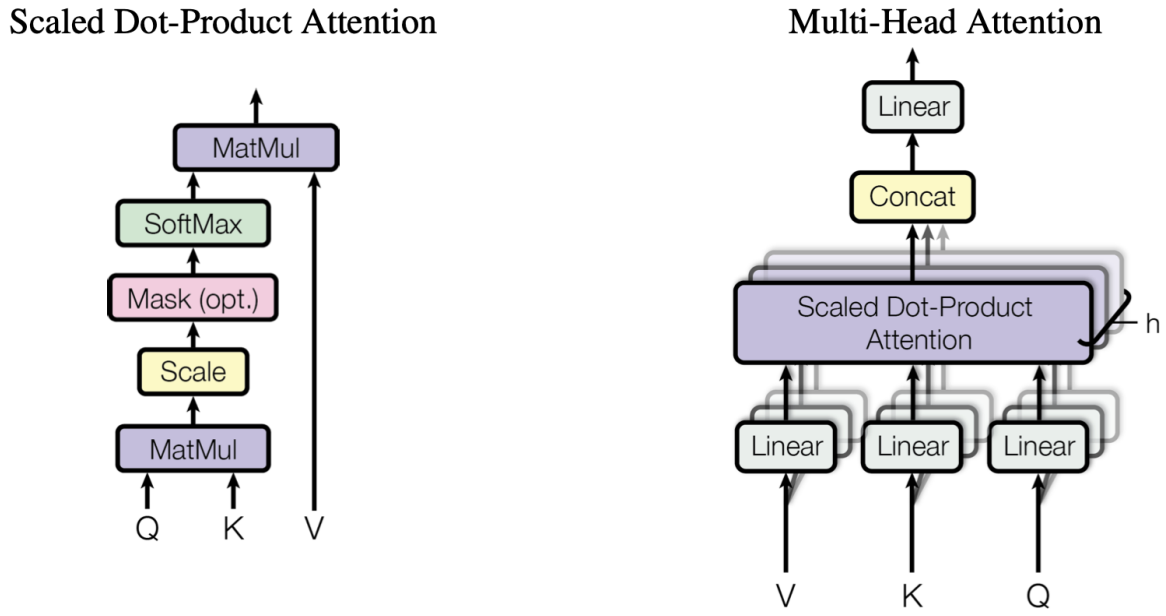

Figure 3.1.: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention. Figure obtained from Vaswani et al. (2017).

This is the so-called *scaled dot-product attention*. Whereas Vaswani et al. (2017) use so-called *multi-head attention* to build transformers. Rather than calculating a single attention, they split Q, K and V into h vectors and apply self-attention to each of them. This means that the model has h different heads, each of which should learn something

different. This is supposed to give the model more representation power. The different heads are computed in parallel and concatenated at the end. The final values are obtained by linearly projecting the concatenated values. Both scaled dot-product and multi-head attention are visualized in Figure 3.1.

This encoder-decoder architecture is the second important concept needed to understand transformers. Introduced in Sutskever et al. (2014) this architecture is used in many NLP models including transformers. It is used to deal with the problem of in- and output sequences being of different lengths. Imagine, for example, the simple task of translating from one language to another. It will often be the case that the translation does not have the same number of sentences/words/characters as the input. That is where the encoder-decoder comes into play. First, the encoder model transforms the input into a vector of a fixed size. This vector is supposed to contain all of the information from the input. A second model, the decoder, then takes this vector as an input and predicts the outcome. As a result the input and the output sequences don't have to be of the same lengths.

Putting those two ideas together leads to the final transformer architecture, as shown in Figure 3.2. The basic idea behind each step will now be briefly explained, starting with the encoder. As explained previously, the encoder is used to map the input sequences into continuous representations that hold all of the learned information from the input together with context added through attention that will help the decoder focus on the important tokens.

Since NNs can only deal with numerical input and not with text, the texts must be converted. First they are tokenized, i.e. divided into individual words or meaningful sub-words. These tokens are then transformed into embeddings, i.e. numerical vectors that are representations of the text input. The absence of recurrence or convolution in the model would cause it to loose information on the sequence's order. That is why the next step is to inject positional information into the embeddings. Vaswani et al. (2017) do this using sine and cosine functions, creating vectors depending on the time step, and adding those vectors to the embeddings.

Next comes the encoder layer (Vaswani et al. (2017) use a stack of $N = 6$ identical layers). It starts with multi-head self-attention. Self-attention is a special type of attention that allows the model to relate each token in the input sequence to the other tokens in that sequence. This is done so the model knows which tokens to pay attention to when dealing with a certain token. There is a residual connection (He et al., 2016) around the multi-head self-attention, followed by a normalization (Ba et al., 2016). That

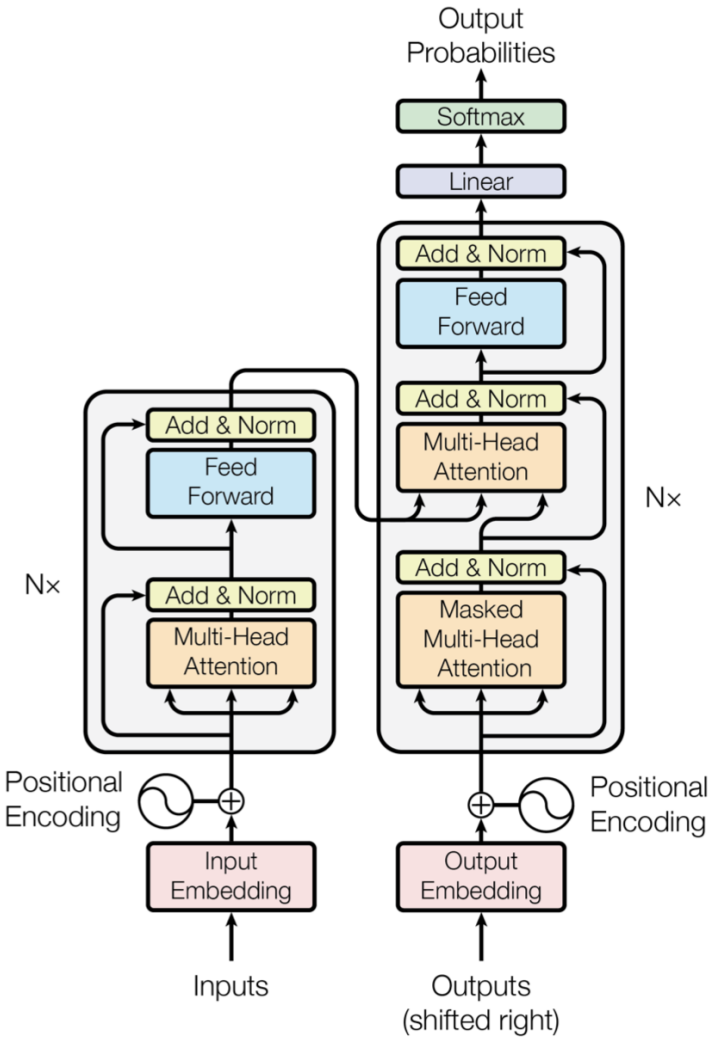

Figure 3.2.: Transformer-model architecture. Figure obtained from Vaswani et al. (2017).

means that the output from the multi-head attention gets added to the embeddings (with positional encoding), and together they are normalized. This then gets fed into a point-wise feed-forward NN consisting of two linear layers with a ReLU activation (Nair and Hinton, 2010) in between them. There is also a residual connection around, which gets added to the output of the feed-forward network and is normalized.
Next is the decoder. Vaswani et al. (2017) stack $N = 6$ identical layers. Each consists of three sub-layers, each with a residual connection around it, followed by layer normalization. The decoder gets two inputs: the true outputs (what it should predict) and the output from the encoder. Before being fed into the decoder, the true outputs are being embedded and positional encodings are added analogous to to the inputs for the encoder. Next, a

masked multi-headed self-attention layer is applied. Future tokens shouldn't be taken into account by the decoder since it creates a sequence token by token and is autoregressive. That is why this attention layer is masked, a mask (setting attentions to $-\infty$) is placed on future tokens to prevent them from influencing predictions. This mask is added to the score matrix before scaling it, which leads the attention scores for future tokens to be zero and the model essentially ignoring those tokens. Apart from the masking, the process is the same as for the multi-headed self-attention layer in the encoder.
Next is the multi-head "encoder-decoder attention". The output of the encoder serves as the source for the keys and values, while the queries are derived from the preceding decoder layer. Therefore, each location in the decoder can pay attention to every position in the input sequence, allowing the decoder to decide which encoder input is important. The feed-forward layer is built in the same way as in the encoder. Finally, the output goes through a linear layer that acts as a classifier, and after softmax is applied, it outputs the probabilities between 0 and 1 for each token.
By demonstrating that self-attention could be used instead of recurrent and convolutional neural network architectures, the transformer architecture revolutionized the field of NLP. LMs' accuracy and speed both significantly increased as a result of this innovation, enabling state-of-the-art performance on a variety of language tasks. Some of the most cutting-edge LMs to date, including GPT-3 (Brown et al., 2020) and BERT (Devlin et al., 2018), were created using the transformer architecture, which has come to be the de facto standard for many applications of NLP.

3.3. Pre-trained Language Models

Pre-training is a widely used technique for NLP transfer learning tasks that involves training a large LM on tons of text data before fine-tuning it for a particular task. Pre-training aims to learn a rich representation of a language's underlying linguistic relationships and patterns. The model can then be fine-tuned using this representation as a strong starting point to perform a specific task, such as political stance prediction. Using a pre-trained model offers several benefits over training a model from scratch. The large amounts of text data used have already taught the Pre-trained model a lot of information that can be used to boost performance on the target task. When compared to training a model from scratch, this can save time and computational resources while also improving overall performance.
In recent years, several pre-trained LMs have been introduced that have shown remarkable

performance on a range of language tasks. ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), T5 (Xue et al., 2020), ELECTRA (Clark et al., 2020), and various GPT models (Radford et al., 2018, 2019; Brown et al., 2020) are some examples of these models. Following is a brief introduction to each of the different models, followed by some solutions on how to use them when handling non-English (specifically German) text data.

3.3.1. ELMo

Peters et al. (2018) introduce ELMo (**E**mbeddings from **L**anguage **Mo**dels), a bidirectional LM that combines a forward and a backward LM. Bidirectional means that the model can use information from both sides (what happened before and after the current token). ELMo uses two BiLSTMs, which consist of two separate unidirectional LSTMs, to achieve its bidirectionality. This is not only a "shallow" concatenation of independently trained unidirectional LMs (Devlin et al., 2018), but the sequential nature of the LSTMs means that ELMo a) is not fully parallelizable and b) fails to capture long-range dependencies during contextualization.

3.3.2. GPT models

OpenAI created a group of deep learning models known as the GPT (**G**enerative **P**re-trained **T**ransformer) model family. The first model in the series, GPT-1, was introduced in June 2018 (Radford et al., 2018) and was followed by GPT-2, GPT-3 and ChatGPT (Radford et al., 2019; Brown et al., 2020; OpenAI, 2022).
GPT-1 (Radford et al., 2018) is a transformer that combines multiple decoder blocks with a standard LM objective for pre-training and can then be fine-tuned with task-specific data. The architecture supported transfer learning and could carry out a variety of NLP tasks with little tweaking. GPT-1 demonstrated the value of pre-training a model and then adapting the fine-tuning depending on the task, which improves the model's generalization. The model provided a starting point for other models that could more fully realize this potential with larger datasets and more parameters. But while GPT-1 solved some of the problems of previously released ELMo, it is still just unidirectionally contextual.
Using a larger dataset and adding more parameters to the model were the main developments in the GPT-2 model that was introduced a year later (Radford et al., 2019). But there were also some other changes. The goal of GPT-2 was to learn multiple, different

tasks using the same unsupervised model. This is called task conditioning, the model is supposed to produce a different output for the same input if given a different task. In order to condition a LM on a task, examples or plain-language instructions are given to the model. Being able to task condition makes GPT-2 capable of zero-shot learning and zero-shot task transfer. Zero-shot learning refers to the ability of a model to recognize and classify objects or concepts it has never seen before. Zero-shot task transfer means that the model is able to perform a task without being specifically trained on it. The input was formatted so that the model had to comprehend the task at hand and respond appropriately. For instance, in a question-answering task, the input would be the question followed by a question mark (?), indicating that the model was expected to generate an answer. Without any prior experience with that specific task, the model was created to understand the nature of the task and provide accurate answers.

Open AI developed the GPT-3 (Brown et al., 2020) model with 175 billion parameters, which has 100 times more parameters than GPT-2, in an effort to create an extremely robust and powerful LM that would require little to no fine-tuning and only a few demonstrations to comprehend tasks and carry them out. GPT-3 is trained to be capable of zero-shot and few-shot learning. Few-shot learning refers to the ability to perform a task with only a few examples. This means that users can provide a few examples and a description of the task, and the model can use this information to generate new outputs. In addition to growing in size and getting better at learning, GPT-3 has a wider range of tasks it can complete. This includes coding, summarizing, translating languages, and answering questions. The model can produce realistic text in a range of genres and styles, making it useful even for literary or artistic endeavors like writing poetry or fiction.

ChatGPT (OpenAI, 2022) is a web app to generate natural language texts for chatbot applications. Due to its specialized optimization for dialogue, which enables it to carry out tasks like explaining code or writing poems in a more chat-like manner, and its accessibility to the public it quickly rose to popularity. With 20 billion parameters, ChatGPT has been specifically designed to generate human-like text in the context of chatbot conversations and has been trained with specific datasets of chatbot interactions.

3.3.3. BERT

BERT stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers and was proposed to solve these existing problems. Introduced by Devlin et al. (2018), it completely replaces recurrent architectures with self-attention by using multiple transformer encoder blocks. The unidirectionality constraint is alleviated by pre-training using a Masked

Language Model (MLM) objective.
Devlin et al. (2018) reports results on two models: $BERT_{BASE}$ and $BERT_{LARGE}$ which have the same general architecture and training steps but differ in size. $BERT_{BASE}$ has 12 layers (transformer blocks), a hidden size of 768, and 12 self-attention heads, while $BERT_{LARGE}$ has 24 layers, a hidden size of 1024, and 16 self-attention heads.
BERTs training can be divided into two steps: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over two tasks. The first task is the already mentioned MLM. 15% of the tokens are masked at random using [MASK] tokens, and the model is then trained to predict those masked tokens. This makes BERT bidirectional but at the same time solves the problem of standard bidirectional conditional models where each word indirectly "sees itself" (Devlin et al., 2018).
The second task is Next Sentence Prediction (NSP). As an input, they use two sentences. 50% of the time, they actually follow each other in the training data, and the other half is just two random sentences from the corpus. The model is trained to detect if the second sentence follows the first. This allows the model to learn the relationship between two sentences.
The pre-trained model has a general understanding of language and then gets fine-tuned depending on the task it is supposed to complete. This is a simple matter of just using task-specific in- and outputs. The input could, for example, be a hypothesis-premise pair that should be classified as true or false. To get the output, the token representing the sequence ([CLS]) is fed into a classification layer. The model then gets initialized with the parameters learned from pre-training and fine-tunes them while training on this (new) downstream task. Since not a whole new model has to be trained, this makes fine-tuning BERT relatively inexpensive and also applicable to a lot of different tasks.

3.3.4. ELECTRA

The name ELECTRA stands for **E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately. The model was introduced by Clark et al. (2020). Unlike BERT, ELECTRA learns from all of the tokens, which is due to a different pre-training objective. It replaces the old MLM task with a new task called Replaced Token Detection (RTD). Additionally, ELECTRA makes use of a novel generator-discriminator architecture that is inspired by the GAN (Generative Adversarial Network) framework used in computer vision.
The generator is an encoder that learns to replace [MASK] tokens (this is a typical MLM objective). The discriminator network then predicts whether each token in the sequence

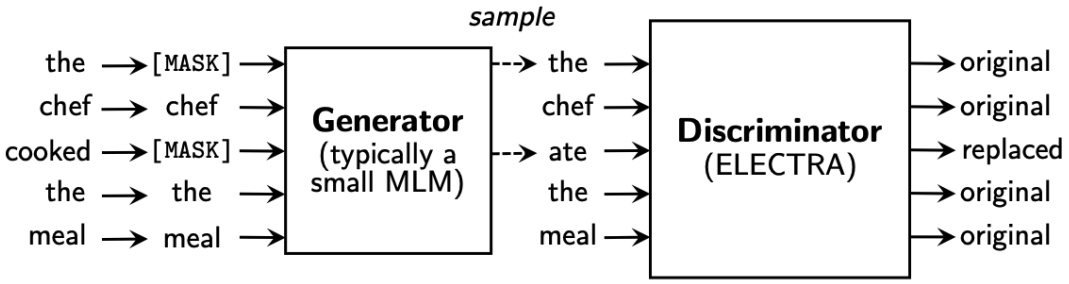

Figure 3.3.: Generator-Discriminator Architecture used for ELECTRA. Figure obtained from Clark et al. (2020).

is original (real) or replaced by the generator (fake). This is the novel RTD task. Instead of just predicting the original masked token, the model learns to predict whether a token in a sequence has been replaced by a fake token. This pre-training objective is more compute-efficient than MLM and results in improved performance on subsequent tasks (Clark et al., 2020).
Clark et al. (2020) have demonstrated that this new pre-training task is effective. Because RTD defines the task over all input tokens rather than just a select few that are masked out, it is found to be more efficient than MLM. Given the same model size, data, and compute, the contextual representations learned by ELECTRA perform better than those learned by BERT.

3.3.5. T5

The T5 model is a text-to-text transfer learning framework based on transformers that can carry out a variety of NLP tasks, such as question answering, summarizing, and translation, among others. It was first introduced by Raffel et al. (2020). T5 employs a unified architecture as opposed to earlier models, allowing it to be trained on various tasks just by altering the prefix added to the input sequence. With this method, the model can handle multiple tasks with a single architecture.
The model itself is similar to the original transformer proposed by Vaswani et al. (2017) but removes the layer norm bias, places the layer normalization outside the residual path, and uses a different position embedding scheme (Raffel et al., 2020). In contrast to models like BERT or ELECTRA, T5 is a "text-to-text" model, which means that it has been trained to map input text sequences to output text sequences. As opposed to many

other NLP models, which are trained for particular tasks like classification, translation, or question-answering, this makes T5 general-purpose. In order to be able to carry out the different tasks, T5 employs a "task prefix", which identifies the precise task being carried out, enabling the model to be more precisely tuned for a particular task (more on this in section 3.5.2). Figure 3.4 shows how the task prefixes work together with the text-to-text framework.

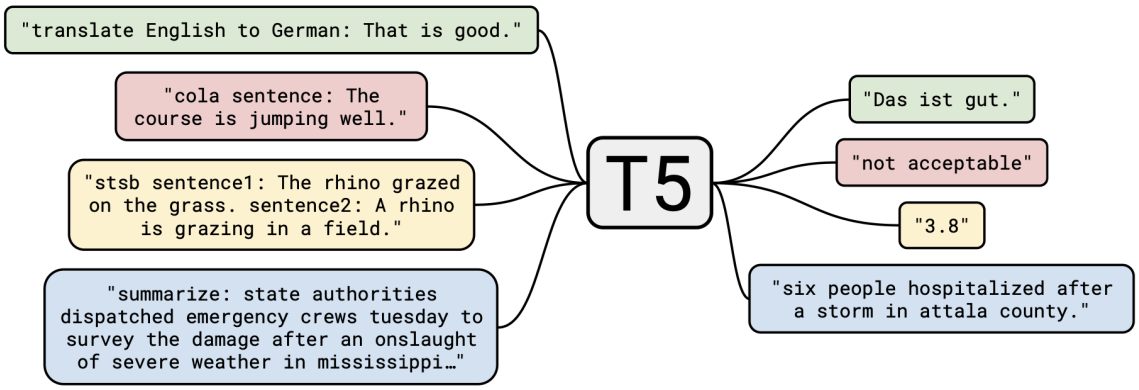

Figure 3.4.: Examples how the T5 text-to-text framework with task prefixes works. Figure obtained from Raffel et al. (2020).

As Roberts (2020) explains, the pre-training objective is to predict missing words in a corrupted text. These "blanks" can also appear at the end of the text, making this objective a generalization of a continuation task. They developed a new downstream task called "sized fill-in-the-blank", which asks the model to fill in a blank with a predetermined number of words. One could train the model to fill in the blank with roughly two words by giving it the input "I like to eat peanut butter and _2_ sandwiches." for instance. The model would then predict something like "I love peanut butter and jelly on my sandwiches.", filling the blank with three words.
The authors train five variants of T5: a small model, a base model, a large model, and models with 3 billion and 11 billion parameters. They perform well overall, with the exception of translation. The fact that they are able to perform nearly as well as humans on the SuperGLUE natural language understanding benchmark — which is made specifically to be challenging for machine learning models but simple for humans — is one of their more impressive results.

3.3.6. German Models

NLP has undergone a revolution thanks to the creation of LMs like the ones discussed, which have allowed researchers and practitioners to create more precise and effective algorithms for a variety of tasks. These models, however, are not applicable to other languages than the ones they were trained on (most of the time English), necessitating the development of language-specific models to guarantee the best results on tasks involving a given language. In order to deal with the data presented in this thesis, models that are able to deal with the German language are needed.
There are two approaches to this. Either train an architecture from scratch with German data or use a model that is capable of understanding multiple languages (including German). For ELMo, BERT and ELECTRA the first approach was chosen. May (2019) train a German ELMo on the German Wikipedia Text Corpus.
There are several German BERT models available that use the BERT architecture but have been trained on large German language corpora. For the following experiments, "bert-base-german-cased" is used. It was trained by Chan et al. (2019) on German news articles, German Wikipedia, and open legal data (German court decisions).
Instead of the English ELECTRA, this thesis uses the German "electra-base-german-uncased" (May and Reißel, 2020). Trained on the Cleaned Common Crawl Corpus 2019-09 German, German Wikipedia, Subtitles, and News, this model is the top-performing publicly available German NLP model on a variety of German evaluation metrics.
GPT-3 is multilingual right out-of-the-box and can produce text in a wide range of languages, including English, Spanish, French, German, Chinese, and many others, because it was trained on an extensive set of text from the internet. It is important to remember that the performance of the model will differ depending on the language and level of complexity of the text being generated. Additionally, mGPT (Shliazhko et al., 2022), a model that supports 60 languages, is available.
For T5, there is also a multilingual model. mT5 (Xue et al., 2020) is the multilingual version of T5 and was pre-trained on a corpus covering 101 languages, including German. Unlike T5, it has to be fine-tuned before it is usable on a downstream task.

3.4. Semantic Search

In order to find relevant information from a text (in this case, the party manifestos), one has to employ a search. When given a query sentence, the goal is to find the sentences in the text that are the most related to the query.

Semantic search is one of the techniques that could be used and refers to the process of finding information in a more human-like way by taking into account the meaning of the search query and the context in which it is being made. Traditional keyword-based search techniques (like BM25) only match the exact terms that the user enters, but semantic search goes beyond this. Semantic search seeks to comprehend the purpose of the query and the relationships between the words in order to provide more accurate results.
Word, sentence, or paragraph embeddings are the key component used in semantic search. Embeddings are numerical representations of words that capture the meaning and context of words in a continuous vector space. These representations enable word (or sentence or paragraph) comparison based on similarity and are learned using different algorithms (some of which will be presented here).
A central concept in semantic search is similarity. It measures how close two words or phrases are to each other in meaning. One way to determine the similarity between the phrase embeddings of the search query and the documents being searched is by calculating the dot products. The higher the product, the more likely it is that the sentence is relevant to the query.
In the following chapter, different approaches to semantic search (and BM25 as a baseline) are described. And later on (in Chapter 5), the results are compared, giving an estimate of how well they work together with the different models.

3.4.1. BM25

BM25 is a ranking function to determine the relevance of documents to a given query, often used by search engines. It was developed in the mid-1990s by Stephen E. Robertson, Karen Spärck Jones, and others (Robertson et al., 1995). It utilizes a bag-of-words retrieval function. A set of documents is ranked based on the number of query terms appearing in each of the documents. There are multiple implementations; here, BM25+, as introduced by Lv and Zhai (2011), is used. It is chosen over other adaptations since, first of all, it shows good performance in Trotman et al. (2014), and second, it is implemented in the python package *rank_bm25* by Brown (2020).
The retrieval status value rsv_q for a given query q is computed as follows:

$$rsv_q = \sum_{t \in q} log\left(\frac{N+1}{df_t}\right) \cdot \left(\frac{(k_1+1) \cdot tf_{td}}{k_1 \cdot \left((1-b) + b \cdot \left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}} + \delta\right), \tag{3.2}$$

where q is the query with individual terms t. N is the number of documents in the

collection, df_t is the document frequency (the number of documents containing the term t), tf_{td} is the number of times a term occurs in document d. L_d is the length of the document (in terms), and L_{avg} is the average document length. k_1, b and δ are parameters to be tuned.
BM25 (and all the different adaptations) is, in contrast to the other methods that follow, a simple keyword-based algorithm and has nothing to do with BERT or other LMs. It is chosen as a baseline to compare the much more complicated semantic search methods to.

3.4.2. SBERT

Earlier, BERT was introduced as a model for NLP. However, it is not suitable for semantic similarity search. To compute the similarity between a group of sentences, all pairs of sentences have to be fed into the network. That means that for a collection of 10 000 sentences, BERT has to do 49 995 000 pairwise computations (Reimers and Gurevych, 2019). This is often solved by obtaining fixed-size embeddings for the sentences, either by averaging the BERT output layer or by using the output of the [CLS] token (which appears at the beginning of every sentence). But as Reimers and Gurevych (2019) show, this practice doesn't produce good sentence embeddings. Instead, they offer a modification of BERT called SBERT (Sentence-BERT), which fine-tunes BERT. By using siamese and triplet networks, they update the weights in such a way that the produced sentence embeddings are semantically meaningful while still maintaining the accuracy of the original BERT. The result are fixed-size embedding vectors for input sentences that can be compared with each other using similarity measures like cosine-similarity or dot product.
The network structure that has to be trained depends on the data at hand. Reimers and Gurevych (2019) present three objective functions to deal with sentence embeddings obtained from BERT (they also experiment with ways to obtain the embeddings, for more on that, please see the paper):

- *Classification Objective Function* (visualized in Figure 3.5): The two sentence embeddings, u and v are concatenated with the element-wise difference $|u - v|$ and multiplied with a trainable weight W_t. Then a softmax is applied. The objective function is optimized using cross-entropy loss.

$$o = \text{softmax}(W_t(u, v, |u - v|)) \tag{3.3}$$

- *Regression Objective Function*: The cosine-similarity between u and v is computed,

and mean-squared-error loss is used as the objective function.

- *Triplet Objective Function*: Instead of pairs, there are now triplets of sentences, an anchor sentence with embedding a, a positive sentence with embedding p and a negative sentence with embedding n. The goal is to get a smaller distance between a and p than between a and n. This can be achieved by minimizing the following loss function:

$$max(||a - p|| - ||a - n|| + \epsilon, 0) \tag{3.4}$$

where $|| \cdot ||$ is Euclidean distance and margin $\epsilon = 1$.

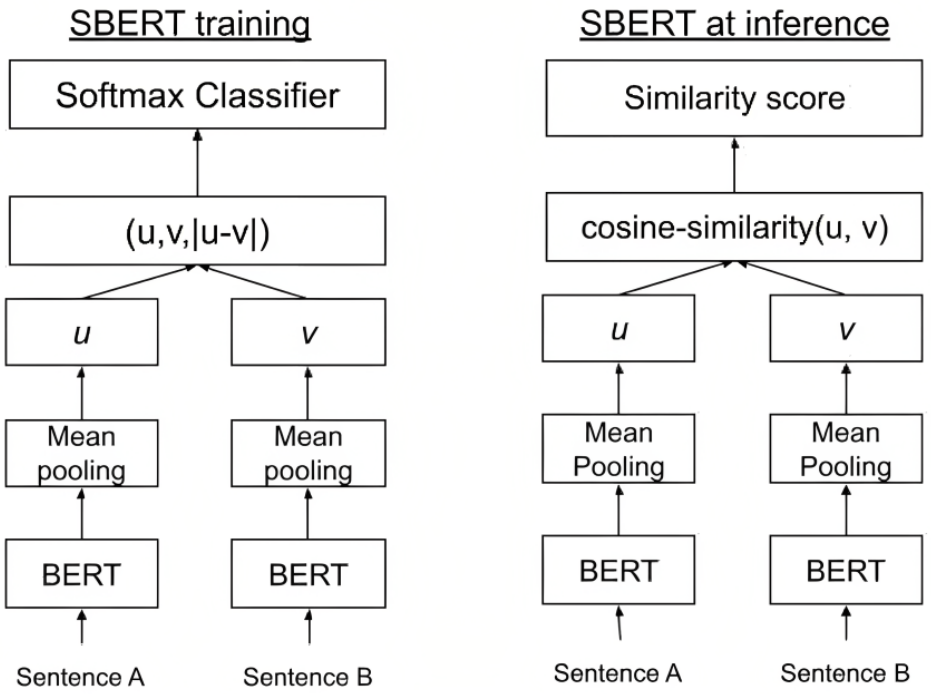

Figure 3.5.: Illustration of the Classification Objective Function of SBERT. Figure obtained from Kotamraju (2022).

It is important to remember that training a SBERT model requires labeled sentence pairs. There are, however, a variety of pre-trained models that can be used to produce high-quality embeddings for numerous downstream tasks without the need for those labeled pairs. Since the data for this thesis doesn't contain labeled sentence pairs, "paraphrase-multilingual-mpnet-base-v2" (Reimers and Gurevych, 2019), one of those pre-trained models that supports German, is used.

3.4.3. BERT-Flow

Li et al. (2020) follow a different approach. They attribute the poor performance of BERT sentence embeddings (obtained by mean pooling over the first and last layers) to not fully exploiting the semantic information that already exists in the BERT embeddings. BERT's sentences live in a non-smooth, anisotropic semantic space, which hinders semantic similarity tasks. That is why they propose an unsupervised approach to transform the embedding distribution. Through normalizing flows, the anisotropic, non-smooth distribution is changed to a smooth and isotropic Gaussian distribution, as is illustrated in Figure 3.6. By using an invertible mapping from the BERT embedding space to the Gaussian space, the information encoded in the embeddings does not change, but the embeddings are better at semantic similarity tasks. To achieve this Li et al. (2020) "learn a flow-based generative model by maximizing the likelihood of generating BERT sentence embeddings from a standard Gaussian latent variable".

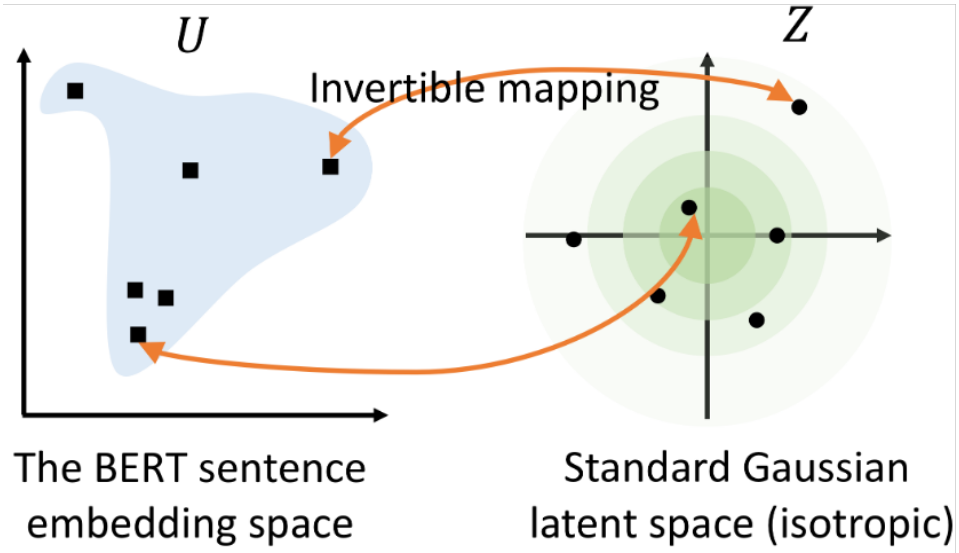

Figure 3.6.: Illustration of the goal of BERT-Flow. Figure obtained from Li et al. (2020).

They call the observed space (the BERT embedding space) U, and the latent space (the standard Gaussian latent space) Z. The goal is to learn $f : Z \rightarrow U$ with $u = f_\phi(z)$, which is an invertible transformation. $z \sim p_Z(z)$ is the prior distribution, a standard Gaussian distribution. The probabilistic density function for U is given as

$$p_U(u) = p_Z(f_\phi^{-1}(u))|det\frac{\partial f_\phi^{-1}(u)}{\partial u}|. \tag{3.5}$$

Therefore, in order to maximize the likelihood of U's marginal solve

$$max_{\phi} \quad \mathbb{E}_{u=BERT(sentence),sentence\sim D} \quad log(p_Z(f_{\phi}^{-1}(u))) + log(|det\frac{\partial f_{\phi}^{-1}(u)}{\partial u}|) \tag{3.6}$$

where D denotes the dataset.

Using the learned function f_ϕ and inverting it makes it possible to transform all BERT sentence embeddings u into latent standard Gaussian representations z. Li et al. (2020) follow an approach similar to Glow (Kingma and Dhariwal, 2018) by stacking multiple invertible transformations to parameterize f_ϕ as a neural network. It is important to note that training only affects the parameters of f_ϕ while the BERT parameters remain unchanged. Additionally, this implies that BERT-Flow can be used with any sentence embeddings. Both BERT (mean pooling over the first and last layers) and SBERT embeddings are used in this thesis.

3.4.4. Whitening

Whitening also attempts to solve the problem of non-smooth, anisotropic distribution. But instead of learning a network like with BERT-Flow, Huang et al. (2021) use a simple linear transformation. Whitening normalizes sentence embeddings with the goal of improving semantic similarity matching. It does this by transforming the covariance matrix to an identity matrix using a given mean and covariance matrix. The mean and covariance matrices are estimated from the sentence embeddings D (obtained by mean pooling the BERT embeddings). By applying

$$\hat{D} = (D - \mu)OC^{-\frac{1}{2}} \tag{3.7}$$

where μ is the mean vector of D, C is a diagonal matrix with the eigenvalues of the covariance matrix $Cov(D) = (D - \mu)^T(D - \mu)$ and O is the corresponding orthogonal matrix of eigenvectors (satisfying $Cov(D) = OCO^T$), one gets the whitened sentence embeddings $\hat{D}$. This approach is much simpler and easier to implement since it only requires estimating μ and C, while BERT-Flow calls for a whole NN to be trained. As the process of Whitening can be applied to any sentence embeddings, it is utilized in this thesis on BERT (mean pooling) and SBERT embeddings.

3.4.5. SBERT-WK

Wang and Kuo (2020) propose a method of dissecting BERT-based models through geometric analysis of the space spanned by the word representations. The goal is to

make use of all the different layers of BERT that capture different linguistic properties. SBERT-WK doesn't require further training; it uses existing word representations and adds them together in a new way to form a sentence embedding.

The first step is to look at each word's alignment and novelty properties, and then combine the word's representations across layers to construct a unified word representation. To then get the final sentence embedding vector, a weighted average of all unified word representations is computed based on the word importance measure.

To compute the unified word representation $\hat{v}_w$ of word w (for all N words) Wang and Kuo (2020) assign weight α_i to the words i-th layer representation v_w^i and take an average:

$$\hat{v}_w = \sum_{i=0}^{N} \alpha(v_w^i) v_w^i. \tag{3.8}$$

The weights α are a weighted average of two ways to measure the new information brought by a word interpretation at the i-th layer: inverse alignment and novelty. For a more detailed explanation of the two see the paper (Wang and Kuo, 2020). The idea behind the inverse alignment measure is that if neighboring layers word representations align well, they don't provide much additional information, so the weight should be small. Therefore, the inverse of the alignment similarity score is used. Novelty measures the new information a word representation brings with respect to the subspace spanned words in its neighboring window. Using a weighted average of those two measures finally leads to a unified word representation for each word.

In the next step, the unified word representations get added together to form a sentence embedding v_s

$$v_s = \sum_j \omega_j \hat{v}_{w(j)} \tag{3.9}$$

where $\hat{v}_{w(j)}$ is the unified word representation for the j-th word in a sentence.

They are weighted by their importance ω_j, which is defined as

$$\omega_j = \frac{|\sigma_j^2|}{\sum_k |\sigma_k^2|}, \tag{3.10}$$

where σ_j^2 is the variance of the offset-1 diagonal values of the cosine similarity matrix for the j-th word. That means that words that develop more quickly across layers get higher weights.

3.4.6. IS-BERT

IS-BERT is an extension on top of BERT that can learn sentence representations in an unsupervised manner, as proposed by Zhang et al. (2020). As mentioned before, sentence embeddings for BERT are often obtained by averaging the BERT output layer or by using the output of the [CLS] token. Instead, IS-BERT uses CNN (Convolutional Neural Network) layers with mean-over-time pooling, which are trained using a novel self-supervised objective that maximizes the mutual information between the global sentence embeddings and all of their local context embeddings. In order to utilize all the aspects of a sentence's local context information, the local contexts of other sentences are used as negative examples for contrastive learning.

In general, contrastive learning is a machine learning approach used to learn the general characteristics of a dataset without the need for labels by teaching the model to differentiate between similar and different data points. Instead of labels, the model needs pairs. A positive pair consists of two similar data points, and a negative pair consists of two contrasting data points. The model computes a similarity measure for the pairs and gets trained in such a way that the similarity for positive pairs should be high and that for negative pairs should be low.

But what are the local context representations? The process is illustrated in Figure 3.7. The first step is to get the length-l sequence of token embeddings $h_1, h_2, ..., h_l$ that can be obtained from BERT by encoding the input sentence. Multiple CNN layers with different kernel sizes are applied on top of these token embeddings. Different kernel sizes are used in order to be able to capture the different n-gram local contextual dependencies of the input sentence. The representations obtained with different window sizes are concatenated together to form one final local representation of a token ($\mathcal{F}_\theta^{(i)}(x)$, where x is the input sentence and i is the token index). To go from these token- to a sentence-representation, a mean-over-time pooling layer is used to create the global sentence representation $\mathcal{E}_\theta(x)$.

As mentioned before, the goal of contrastive learning is to maximize a similarity measure for positive pairs and minimize it for negative pairs. For IS-BERT, each sentence and its local context representations are treated as positive examples, and all the local context representations for other sentences are negative examples (as illustrated in Figure 3.8). In Zhang et al. (2020) the similarity measure to estimate the mutual information between the global sentence representation $\mathcal{E}_\theta(x)$ and each of the local token representations $\mathcal{F}_\theta^{(i)}(x)$ is the Jensen-Shannon estimator. Maximizing it encourages the global sentence representation to have high mutual information with its own local context representations. As a consequence, the model will be driven to preserve the distinctive information that

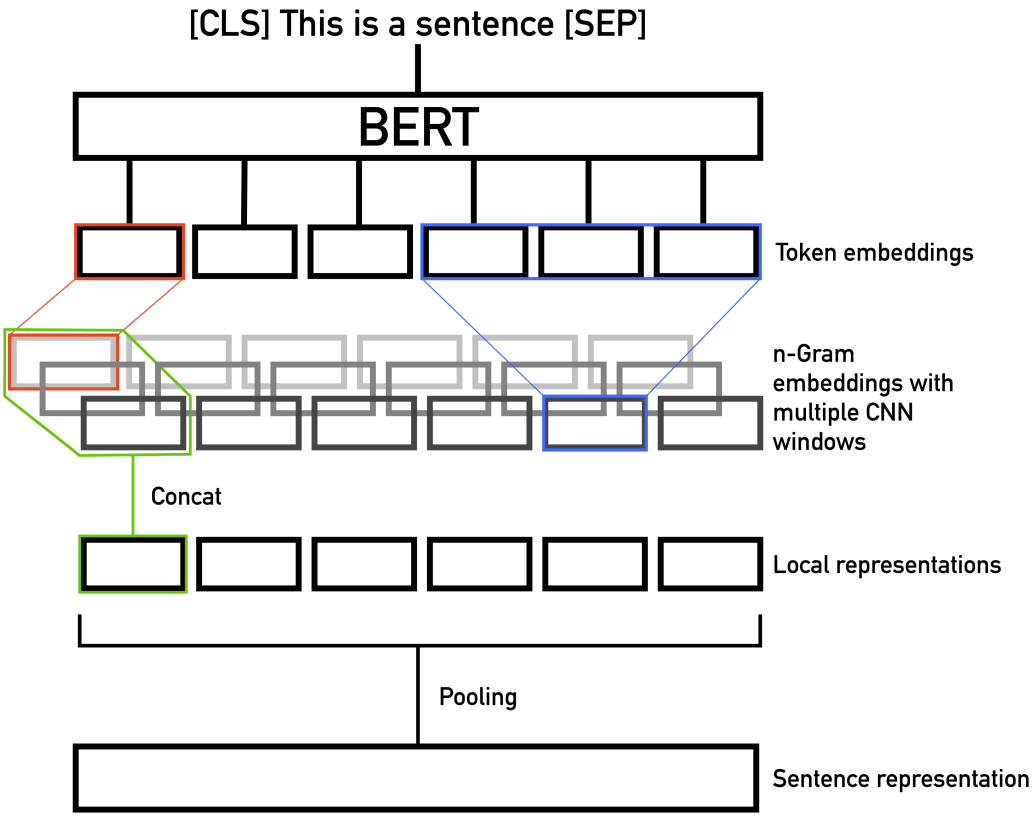

Figure 3.7.: Process describing how to get the local context representations for IS-BERT

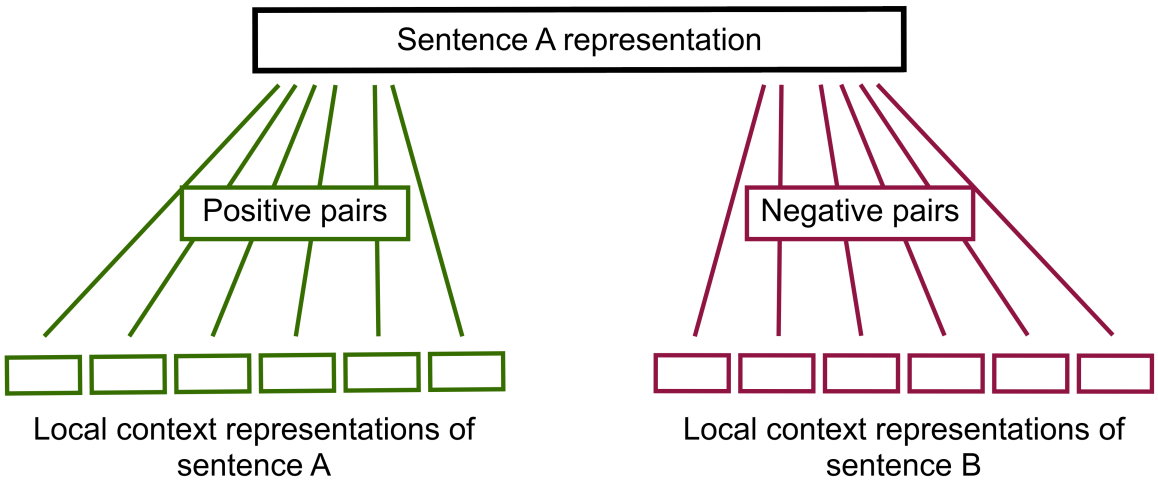

Figure 3.8.: Contrastive learning in IS-BERT

is shared by all local segments of the input sentence while remaining distinct from other sentences, resulting in expressive sentence representation (Zhang et al., 2020).

3.5. Input Pattern Design

A model needs input in order to produce predictions. What this input is depends on the task. For NLP models, the inputs are words, sentences, or paragraphs. And in the case of the inputs in this thesis, it gets even more complex. The input has to contain different pieces of information (the party, the user query, and the context) that somehow have to be put together to be fed into the model. There is not one way to do this; instead, this is another part where experimentation is needed. Following is a brief overview of prompting and prefix design in general before Chapter 4 goes over the different strategies utilized in this thesis.

3.5.1. Prompting

In recent years, there has been a great deal of interest and research in the area of prompting in NLP. Many of the current state-of-the-art LMs, such as GPT-3 (Brown et al., 2020) use prompting techniques to achieve their impressive performances. But what is prompting?

Prompting is the process of giving a stimulus or cue to a LM. A prompt is used to direct the model's generation process in a specific direction and can be in the form of a question, a statement, or a list of keywords. This can be done by reshaping the original task into a language modeling problem, a passage of text that has some unfilled slots. The LM then fills in the blanks, and that leads to the final output. For example: A model trained on general English data could be used to predict whether a review is positive or negative. To do this, the example sentence "I did not have a good time." could be transformed into a prompt such as "I did not have a good time. It was _____." The model then predicts something like "terrible", which indicates a negative review.

There are several benefits to using prompting in NLP. First and foremost, it can improve the performance of the model on a variety of tasks. By providing the model with a clear and relevant prompt, it is more likely to generate text that is accurate and useful. Prompting can also assist in lowering the amount of data needed to train a LM because, when given instructive prompts, the model can learn from fewer examples since it utilizes the general knowledge of language that a large pre-trained model has. A new task could be applied as few- or even zero-shot learning simply by using the right prompt. That makes fine-tuning much easier, faster, or even obsolete. Instead of trying to push the model to fit the data , the data is adjusted to suit the pre-trained model.

But that also means that the effectiveness of prompting relies on the model's ability

to understand the context of the prompt and generate relevant and coherent responses. Without a good LM as its base, any prompt, however good, will not lead to good results. Picking the right prompts is one of the biggest challenges. A poorly constructed prompt may lead to false or irrelevant responses, which could harm the model's overall performance. Additionally, using prompts can introduce biases into the model because they might contain presumptions or preferences that have an impact on the results.

3.5.2. Task Prefix

A versatile and effective method for fine-tuning big LMs is the use of task prefixes. That is what T5 (Raffel et al., 2020) is doing, for example. Models can be trained to carry out a variety of NLP tasks by specifying a distinct prefix for each one, without significantly altering the model architecture or training procedure. The model determines the task it must carry out by examining the prefix at the beginning of the input sequence, and then it produces the appropriate output. By using the prefix "sentiment analysis:", for instance, a model could be trained to perform sentiment analysis on movie reviews. Input sequences that start with this prefix could then be categorized as either positive or negative, depending on the language used in the review. The same model could also be used to translate texts by utilizing the prefix "translate English to German:" followed by an English text.
The ability to simultaneously train a model on multiple tasks is one of the key benefits of using task prefixes. Using the same model for a variety of tasks like summarization, translation, and classification without them interfering with one another leads to increased efficiency and cost savings. Moreover, the ability to switch between tasks seamlessly without the need for retraining makes task prefixes an effective and adaptable NLP solution.

3.5.3. Prompt Tuning

So far prompts were text chosen by humans depending on what they think might work best. Lester et al. (2021) propose prompt tuning using so-called soft prompts. The LM is frozen, and instead the prompts are learned. The soft prompts are attached to the beginning of each embedded input and initialized at a fixed length. After each model iteration, the gradients calculated through back-propagation are only applied to these soft prompt vectors, keeping the core model the same. While that means that the prompts are not interpretable for humans, they still act in the same capacity as a manually written

text prompt, but without being limited by discrete language.
The authors claim that their approach outperforms prompt design (tested on GPT-3 (Brown et al., 2020) and T5 (Raffel et al., 2020)). Additionally, prompt tuning catches up to model tuning in terms of performance as model size grows. Lester et al. (2021) also show that soft prompting offers better transferability than full model fine-tuning in addition to parameter efficiency.

4. Experiments

This study aims to investigate how different semantic search techniques, input patterns, and models can be used to predict political party stances.
A series of experiments is conducted to compare the effectiveness of six semantic search techniques (plus two variations), examine the impact of five different input patterns, and explore the effectiveness of two language models.

4.1. Semantic Search

The various methods for semantic search presented in chapter 3.4 are applied to all sentences in the 2021 party manifestos. These sentences are shuffled, since this significantly affects the outcomes according to Li et al. (2020). The semantic search models are then applied to this shuffled dataset. There are a total of eight different methods used:

- **BM25**: use the BM25+ algorithm as described in Lv and Zhai (2011), it is implemented in the python package *rank_bm25* by Brown (2020). This is the only algorithm from the list that has nothing to do with BERT (Devlin et al., 2018). It was chosen as a baseline to compare the more sophisticated methods to.

- **SBERT**: the sentence embeddings are obtained by using the pre-trained "paraphrase-multilingual-mpnet-base-v2" model (Reimers and Gurevych, 2020) from the python package *sbert* by Reimers and Gurevych (2019). "Paraphrase-multilingual-mpnet-base-v2" is chosen since it is the highest-performing multilingual model according to Reimers (2022).

- **BERT-Flow**: two BERT-Flow models get trained following Li et al. (2020), one uses "bert-base-german-cased" (Chan et al., 2019) to obtain the base embeddings (from now on called "BERT-Flow") and the other uses the SBERT model "paraphrase-multilingual-mpnet-base-v2" (Reimers and Gurevych, 2020) (from now on called "SBERT-BERT-Flow"). The code used has been altered from Wang (2021).

- **Whitening**: the Whitening transformation from Huang et al. (2021) is also applied to two models. "Bert-base-german-cased" (Chan et al., 2019) (from now on called "Whitening") and "paraphrase-multilingual-mpnet-base-v2" (Reimers and Gurevych, 2020) (from now on called "SBERT-Whitening").

- **SBERT-WK**: the code from "SBERT-WK-Sentence-Embeddings" (Wang, 2020) (Github repository following the paper from Wang and Kuo (2020)) is used to train a model that uses "bert-base-german-cased" (Chan et al., 2019) as its base model.

- **IS-BERT**: using "bert-base-german-cased" (Chan et al., 2019) as the base model, a model as described by Zhang et al. (2020) is trained. The code is from the "IS-BERT" repository (Zhang, 2020).

These eight methods are applied to the queries to get the five most semantically similar sentences from each of the parties manifestos. For all methods except BM-25, that means computing the dot score between the embeddings of the query and all sentences in the manifesto and choosing the five sentences with the highest dot score.

4.2. Input Pattern Design

The input that gets fed into the model can be built from three "building blocks":

- **Party**: the name of the party the model should predict agreement for: *spd*, *cdu* (stands for both CDU and CSU), *grüne*, *fdp*, *afd* or *linke*.

- **Query**: the user-generated input sentence (for example: "BAföG für alle Studierenden").

- **Context**:
  - Up to five sentences from the party manifesto that relate to the user query which are found using the different semantic search techniques.
  - They are added as a list with square brackets [...] to indicate the beginning and the end.
  - If the input is longer than 512 tokens (the maximum input length for the models) for a certain semantic search method, only four (or, if necessary, three) sentences are used as context. This is the case for BERT-Flow (four sentences), SBERT-BERT-Flow (three sentences), and IS-BERT (three sentences).

From these building blocks, five different patterns (plus two variations) are built.

Baseline:

<party>: <query>

for example: linke: 'BAföG für alle Studierenden'
This is the same input that Witte et al. (2022) use to train their model. The input doesn't contain any context and therefore serves as the baseline. By comparing the other input patterns with it, it is possible to determine if adding context improves the performance.

No additional text (pattern 1-1):

<party>: <context>, <query>

for example: linke: ['Wir setzen uns für ein rückzahlungsfreies, elternunabhängiges und bedarfsgerechtes BAföG ein, das alle erreicht, die es brauchen.', 'Das führt zu Stress bei Studierenden und Beschäftigten.', 'Ebenso muss die Kopplung des BAföG an Leistungsüberprüfungen abgeschafft werden.', 'Nur noch 11 Prozent der Studierenden erhalten überhaupt BAföG, nur 8 Prozent den Höchstsatz.', 'Das BAföG muss an die Lebenswirklichkeit angepasst werden und die Ausbildung umfassend finanzieren.'], 'BAföG für alle Studierenden'
a variation of this is pattern 1-2:

<party>: <query>, <context>

Simply concatenating the different building blocks results in these two patterns. But the model is not given any more information on what exactly it is that it receives. It is also of interest if the order of the building blocks matters.

Minimal text (pattern 2-1):

<party>: Satz: <query>, Kontext: <context>

for example: linke: Satz: 'BAföG für alle Studierenden', Kontext: ['Wir setzen uns für ein rückzahlungsfreies, elternunabhängiges und bedarfsgerechtes BAföG ein, das alle erreicht, die es brauchen.', ..., 'Das BAföG muss an die Lebenswirklichkeit angepasst werden und die Ausbildung umfassend finanzieren.']
and as a variation pattern 2-2:

<party>: Kontext: <context>, Satz: <query>

Here, the model gets added information. It now knows that the query is the "Satz" (sentence) and the context is the "Kontext" (context). Does this improve the performance, or is this extra information not needed?

Human-speech-like (pattern 3):

Passt <query> zur <party> und <context>?

for example: Passt 'BAföG für alle Studierenden' zur linke und ['Wir setzen uns für ein rückzahlungsfreies, elternunabhängiges und bedarfsgerechtes BAföG ein, das alle erreicht, die es brauchen.', ..., 'Das BAföG muss an die Lebenswirklichkeit angepasst werden und die Ausbildung umfassend finanzieren.']?

Human-speech-like (pattern 4):

<party> sagt: <context>, passt <query> dazu?

for example: linke sagt: ['Wir setzen uns für ein rückzahlungsfreies, elternunabhängiges und bedarfsgerechtes BAföG ein, das alle erreicht, die es brauchen.', ..., 'Das BAföG muss an die Lebenswirklichkeit angepasst werden und die Ausbildung umfassend finanzieren.'], passt 'BAföG für alle Studierenden' dazu?

Patterns 3 and 4 are similar to how a human would ask a question. This is the style of input a language model like ChatGPT (OpenAI, 2022) would like to receive. It is of interest to see if this also works for the models used here.

That means that, together with the eight different semantic search methods, there are now (8*6)+1 = 49 different inputs.

4.3. Models

The various inputs are fed into two different models. The two models that are used are:

- **ELECTRA**: "electra-base-german-uncased" (May and Reißel, 2020) is used and fine-tuned for 13 epochs with a learning rate of 2.21e-5 (since those were the optimal hyper-parameters in the experiments of Witte et al. (2022))

- **BERT**: "bert-base-german-cased" (Chan et al., 2019) is used and trained for 11 epochs with a learning rate of 1.95e-5 (hyper-parameters also taken from Witte et al. (2022))

The decision to use the ELECTRA and BERT models was based on a number of factors. First of all, these models were selected because Witte et al. (2022), on which this project is based, tested them. Second, the models' size had to be taken into account in order for them to fit within the computing resources available and still provide good performance. Both ELECTRA and BERT were found to be sufficiently compact for the needs of the project and to deliver results that were competitive. Both models are also well-known in the NLP community, and a substantial body of research backs their efficacy. These factors made the ELECTRA and BERT models an ideal choice for the project's needs.

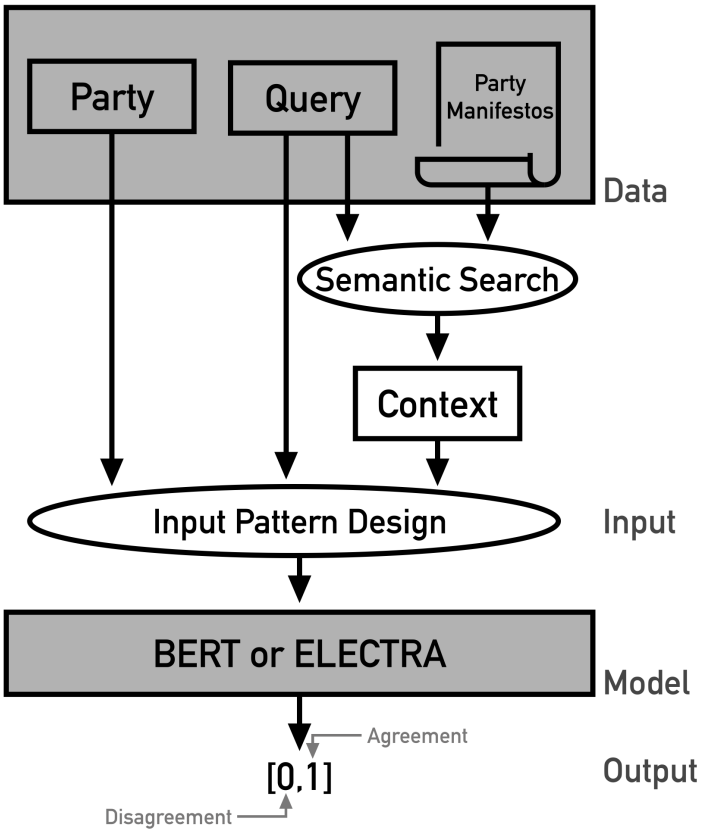

Figure 4.1.: An illustration of the experiments' approach.

Figure 4.1 illustrates how the experiment is set up. Having two models means that there are now 49*2= 98 different predictions for every user query. The goal is to find the combination that works best and see if there are any semantic search techniques or input

pattern designs that work better than others. It is also of interest if adding sentences from the manifestos as context even had any effect on the performance of the models.

4.4. Evaluation

To evaluate the performance, about 30% of the data is held back as testing data and is not used for training. By calculating the accuracy and F1 score using the test data, the models' performance is assessed.
The percentage of instances that are correctly classified out of all instances is measured by the accuracy metric. It is given by the formula:

$$\text{Accuracy} = \frac{\text{Number of Correctly Classified Instances}}{\text{Total Number of Instances}} \tag{4.1}$$

For instance, the model's accuracy is 90% if it correctly predicts 90 out of 100 party stances. Since accuracy is simple to comprehend and interpret, it is frequently used to evaluate model performance. However, it can be misleading in some cases, especially when the classes are imbalanced.
F1 score is a better metric than accuracy when the classes are imbalanced, because it takes both precision and recall into account. Precision measures the percentage of true agreements out of all predicted agreements, while recall measures the percentage of true agreements out of all actual agreements. F1 score is given by the formula:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.2}$$

F1 is also useful when the cost of false agreement and false disagreement predictions are different. For example a false disagreement (a party not agreeing being classified as agreeing to a stance) could be more costly than a false agreement (an agreeing party being classified as disagreeing). However, it appears that this is not the case in this context. The classes are also not imbalanced, the amount of agreement and disagreement is approximately the same both for the training and the testing data. For this reason, the F1 score results are included in the appendix A.2.

4.5. Further Experiments

As of now, the context consists solely of statements taken directly from the party manifestos. Another option is to condense these five (or, in some cases, four or three

sentences) into a single sentence that, ideally, includes all of the relevant information. As the summarized context should be much less complex and contains less unnecessary noise, this might improve the performance of the model. Additionally, it is shorter, which might make it simpler for the model to "remember".

IGEL (Schmid, 2023), which stands for **I**nstruction-tuned **Ge**rman large **L**anguage Model, is a family of LLMs developed specifically for the German language. The BigScience BLOOM architecture (BigScience, 2022), notably the BLOOM-CLP German version by Ostendorff (2022), is the foundation of IGEL. A dataset of instructions that have been automatically translated from English to German is used to train the model. Hallucination, toxicity, and stereotypes—common flaws in language models—are some of IGEL's acknowledged drawbacks. It is utilized in this instance to obtain one sentence summaries.

In order to do this IGEL is given the command: "Als eins Satz zusammenfassen: <context>". This roughly translates to "Summarize in one sentence: <context>", but the number "eins" is used in place of the correct word "ein". This is done because testing revealed that doing so improved the model's comprehension of the fact that it only needed to produce one sentence.

However, this still frequently fails, as evidenced by the summary results for the example context: ['Wir setzen uns für ein rückzahlungsfreies, elternunabhängiges und bedarfsgerechtes BAföG ein, das alle erreicht, die es brauchen.', 'Das führt zu Stress bei Studierenden und Beschäftigten.', 'Ebenso muss die Kopplung des BAföG an Leistungsüberprüfungen abgeschafft werden.', 'Nur noch 11 Prozent der Studierenden erhalten überhaupt BAföG, nur 8 Prozent den Höchstsatz.', 'Das BAföG muss an die Lebenswirklichkeit angepasst werden und die Ausbildung umfassend finanzieren.'].

This context is summarized by IGEL as four rather than one sentence: "Wir fordern eine elternunabhängige, bedarfsgerechte und rückzahlungsfreie Ausbildung, die alle erreicht. Das führt zu Stress bei Studierenden und Beschäftigten, und der Studienerfolg ist unzureichend. Daher fordern wir, die Verknüpfung des BAföG mit Leistungsüberprüfungen abzuschaffen. Wir fordern auch, dass die Ausbildung mit dem realen Leben Schritt hält und die Ausbildung umfassend finanziert wird."

The structure of the experiments, including the summary experiment, is depicted in Figure 4.2. Because IS-BERT has proven to be the most effective semantic search technique (see Chapter 5), only contexts obtained using this method will be used in the experiment. Once the summary is generated, the resulting summary will be used as the context for the model, employing the different input patterns described in Chapter 4.2.

Only ELECTRA will be used out of the two models because it has been found to produce the best results (again see Chapter 5). The results will be discussed by examining the accuracy, with the F1 score results located in the appendix, just like for the other models.

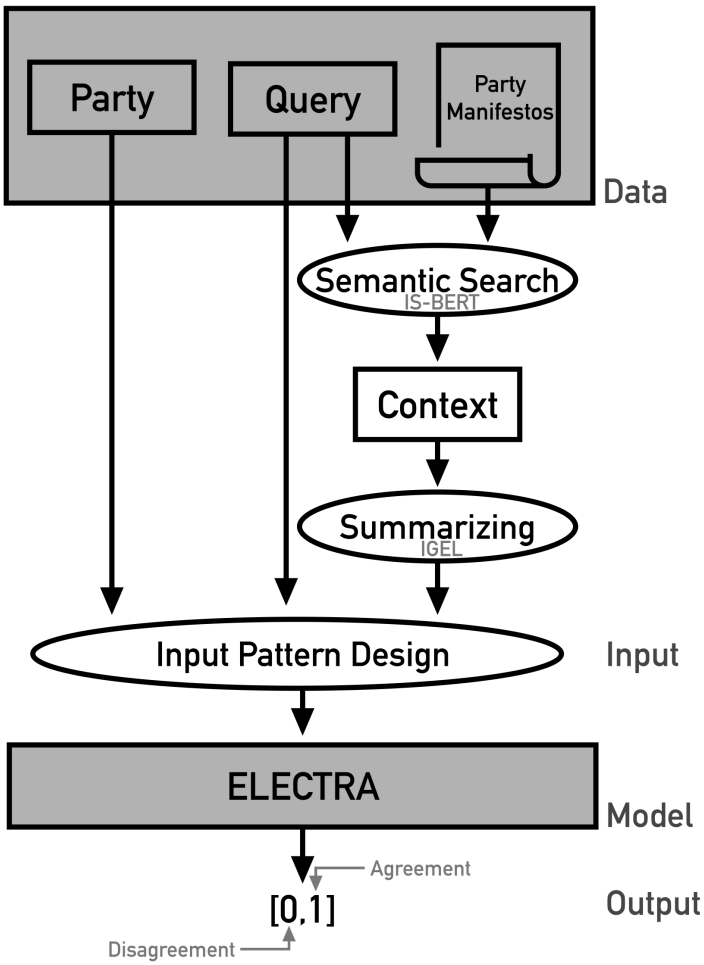

Figure 4.2.: An illustration of the experiments' approach with summarization.

5. Results

The goal of this thesis is to determine whether using data from political party manifestos can enhance party stance prediction. To do this, a model that examines user-generated statements and predicts if the parties will agree with them is built. To add more context, the model is also fed relevant passages from party manifestos.
The thesis discusses several semantic search techniques for locating relevant passages in the manifestos and evaluates various input patterns to determine the most effective means of conveying this additional information to the model. Additionally, it compares how ELECTRA (Clark et al., 2020) and BERT (Devlin et al., 2018) perform when given these various inputs.
For each model, 49 inputs were evaluated, including one without any added context, which served as a baseline with no semantic search. The models' performance was evaluated by calculating the accuracy on the test data.

5.1. ELECTRA

ELECTRA: Accuracy by input and model

Input \ Model	BM25	SBERT	BERT-Flow	SBERT-BERT-Flow	Whitening	SBERT-Whitening	SBERT-WK	IS-BERT	Mean
Pattern 1-1	0.798	0.598	0.800	0.765	0.800	0.623	0.796	0.813	0.749
Pattern 1-2	0.794	0.498	0.794	0.790	0.801	0.795	0.809	0.802	0.760
Pattern 2-1	0.498	0.794	0.498	0.784	0.785	0.805	0.813	0.789	0.721
Pattern 2-2	0.498	0.804	0.819	0.794	0.816	0.755	0.803	0.820	0.764
Pattern 3	0.498	0.782	0.795	0.783	0.745	0.779	0.786	0.790	0.745
Pattern 4	0.758	0.498	0.805	0.792	0.794	0.817	0.773	0.797	0.754
Mean	0.641	0.662	0.752	0.785	0.790	0.762	0.797	0.802	0.749

Accuracy without semantic search: 0.806

Figure 5.1.: The accuracy calculated on the test data for all models with ELECTRA as the base model

The performance of ELECTRA was evaluated using various combinations of semantic search techniques and input patterns, and the results are presented in Figure 5.1. It displays the accuracy scores of the ELECTRA models with and without semantic search. The accuracy without semantic search is 0.806 (displayed below the table), which serves as a baseline for comparing the performance of models that utilize semantic search techniques to provide additional context. Interestingly, only seven out of the 48 context-based models outperformed the baseline model without semantic search. Six models only ever predict agreement (accuracy of 0.498) regardless of the query. This issue was mostly observed in the BM25 models, with three out of the six BM25 models suffering from this problem. The average accuracy for BM25 was the lowest (0.641), while SBERT models performed slightly better, followed by BERT-Flow, SBERT-Whitening, SBERT-BERT-Flow, Whitening, and SBERT-WK. On the other hand, IS-BERT was the most effective semantic search technique overall. The two Whitening models (SBERT-Whitening and Whitening) and the two BERT-Flow models (SBERT-BERT-Flow and BERT-Flow) don't seem to behave similarly to each other.
Notably, the various input patterns produced similar accuracy scores, with Pattern 2-2 performing the best and Pattern 3 performing the worst. Interestingly, the performance of the input patterns does not follow a discernible pattern. Patterns sharing similar properties do not show a consistent performance trend, Patterns 1-1 and 1-2 or 2-1 and 2-2 do not exhibit similar behaviors.

5.2. BERT

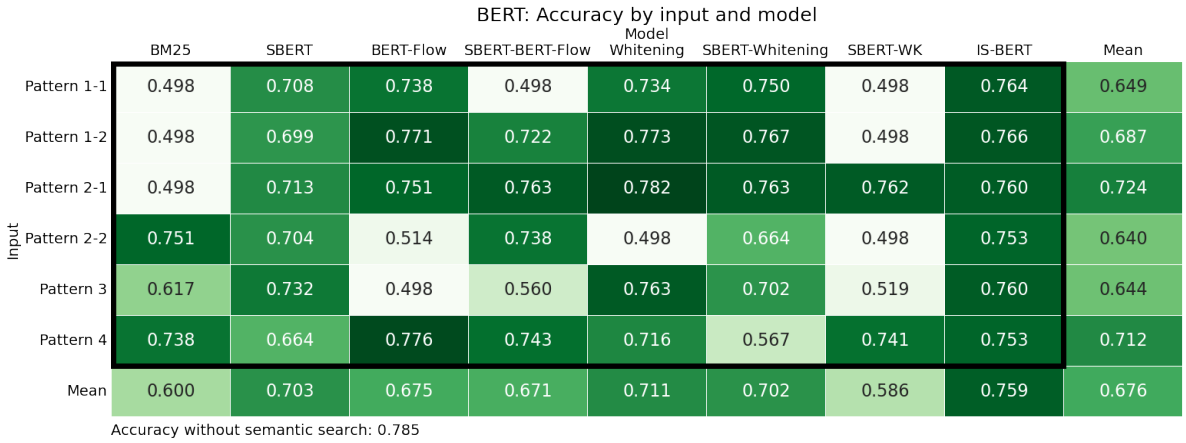

Figure 5.2.: The accuracy calculated on the test data for all models with BERT as the base model

Figure 5.2 presents the results of all models that used BERT as the base model. Overall, the models that incorporated semantic search performed worse than the baseline without semantic search (0.786), with a notable number of models that always predict agreement regardless of the query. Among the semantic search techniques, IS-BERT outperforms the others with a mean accuracy of 0.759, followed by Whitening, SBERT, and SBERT-Whitening. While BERT-Flow and SBERT-BERT-Flow perform better than the baseline BM25, SBERT-WK performs even worse. Again, there seems to be no correlation between how the two Whitening models and the two BERT-Flow models behave.
The two best-performing input patterns are Pattern 2-1 and Pattern 4, while Pattern 3 and Pattern 2-2 perform the worst in terms of accuracy. Once more, there is no discernible pattern in how the input patterns perform.

5.3. Main Results

The results of both the ELECTRA and BERT experiments indicate that incorporating additional context through semantic search techniques does not necessarily improve the performance of the models. In fact, the accuracy of most models that used semantic search were worse than the baseline model without it. However, IS-BERT showed promising results and performed consistently well across all patterns, indicating its effectiveness in this task. On the other hand, models that used BM25 as the semantic search technique performed poorly. The different input patterns did not show a big difference in performance, with only slight variations in accuracy. A considerable number of models showed poor performance by only predicting agreement. Additionally, the best-performing model only had a marginal improvement over the baseline. The experiments revealed that, on average, the ELECTRA models outperformed the BERT models, supporting the findings of Witte et al. (2022).

5.4. Further Experiments

The findings of the summarization experiment discussed in Chapter 4.5 are shown in Figure 5.3. As can be seen, none of the models that make use of the summarized context perform better than the baseline of no context. The worst IS-BERT model without summarization performs equally well as the best result with summarization (accuracy of 0.789). All models perform better without summarization, with the exception of pattern 2-1 (which performs the same).

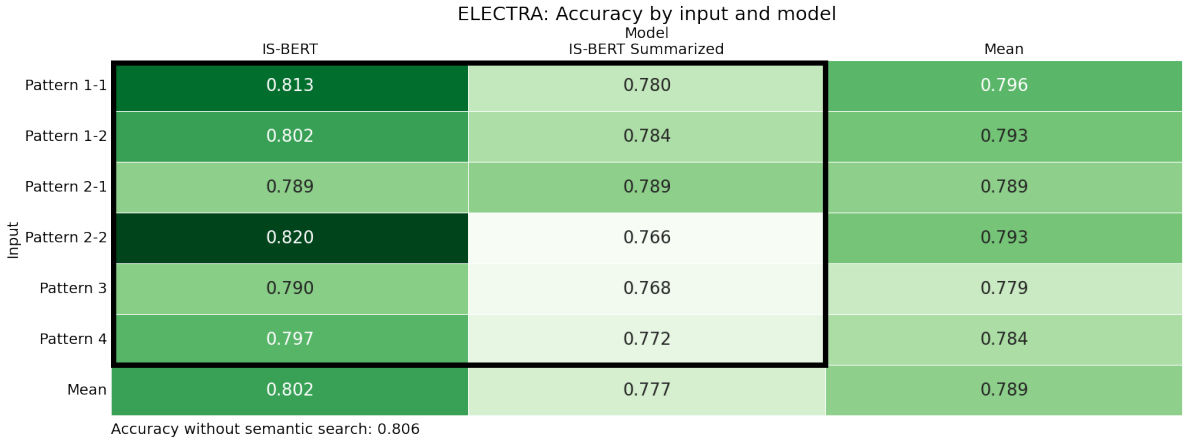

Figure 5.3.: The accuracy calculated on the test data for the IS-BERT and IS-BERT Summarized models (with ELECTRA as the base model)

6. Discussion, Limitations and Outlook

6.1. Discussion

The study examined the effectiveness of incorporating information from political party manifestos to enhance the accuracy of models that predict party stances. The models proposed in this study employ user-generated statements and relevant passages from party manifestos as inputs to predict the political party most likely to agree with the statement.

The study's findings indicate that while incorporating data from political party manifestos could in some cases improve the model's ability to predict party stances, the improvement was not substantial. The performance of various semantic search techniques differed, with some techniques (IS-BERT) providing better results than others, but no technique showed great improvement. The performance of the model is not seriously affected by the various input patterns. In addition, the study compared the performance of two NLP models, ELECTRA (Clark et al., 2020) and BERT (Devlin et al., 2018), and found that ELECTRA was better suited for this task. Overall, it can be said that providing context in the form of party manifesto passages doesn't make the model much better.

In an effort to make the context more understandable, it was summarized. But this resulted in a worse performance when compared to the models without summarization. The context itself may be too confusing or complex to be adequately summarized, or the IGEL model (Schmid, 2023) used for summarization may simply not be the best model for this task.

The lack of noticeable improvement of a model with vs without context could have a number of causes. Firstly, it is possible that the models are unable to effectively handle the increased complexity introduced by the addition of information from party manifestos, leading to confusion and reduced accuracy. Second, party manifestos could include an excessive amount of ambiguous or meaningless language that doesn't add any useful

information to the model and could even hinder its performance by adding extra noise. Finally, it is plausible that using semantic search methods to extract relevant sentences from party manifestos yields subpar results, resulting in less-than-ideal inputs that further perplex the model. These potential explanations warrant further investigation that could help shed light on the limited effectiveness.

6.2. Limitations

There are some general restrictions in addition to these three possible causes for the poor improvement of the model through context adding. This study's language focus is one major drawback. It only looked at how well German-language political party manifestos could be used to integrate context. The results might not apply to other languages as a result. The model may be impacted by the distinctive structural features, idiomatic expressions, and cultural contexts of the language. Therefore, more research is required to determine whether results differ between languages.

Secondly, the study's scope was restricted to a single setting, namely political stance prediction. Although this is a relevant and important application of NLP, the effectiveness of the suggested idea may differ depending on the task or application. Additional context may be very beneficial to a model and improve performance in a different setting than this one.

This study's third drawback is that it only used party manifestos as its context. Manifestos are a great source of information about political parties' ideologies, but they might not include all the necessary details and may be (intentionally) vague and general. The predictive power of the model may be improved by using data from additional sources, such as speeches, interviews, and news articles. Future studies might examine how well various context types can be incorporated into the model. By including these extra sources of information in the models, one may be able to create more reliable and beneficial models that can better assist people in making informed decisions. It is important to keep in mind that incorporating these additional sources will also present new challenges, including the need to eliminate noise and irrelevant data as well as different input patterns, and that additional research will be required to help solve these problems.

6.3. Outlook

The semantic search techniques and patterns employed were limited. Although the study used a variety of semantic search strategies and input patterns, there are still a wide range of strategies and patterns that have the potential to enhance the performance of the model. Additionally, it would seem that better semantic search methods are needed, particularly for German. Investigating the use of other semantic search models, such as a Cohere model (Reimers and Kayid, 2022) created specifically for German semantic search, may be worthwhile.

It might also be interesting to look into how the number of similar sentences used as context affects the results and whether or not using only sentences that are more similar than a given threshold could improve performance.

Furthermore, it might be advantageous to further investigate summarization in order to give the model information that is more condensed and understandable. While one experiment was conducted and did not yield promising results, there are many more models that can be tested. Trying different commands on those models is certainly useful as well, since they greatly impact the outcomes. The small-scale experiment conducted here shouldn't discourage anyone from pursuing this avenue further.

Another approach to the problem of including party manifestos in a predictive model is to change the emphasis from sentences to paragraphs. It's possible that the relevant info is spread out over several sentences, forming a coherent paragraph, rather than being contained in a single sentence. The most relevant paragraph can be found using a semantic search, which can then be summarized to produce a concise sentence that conveys the main idea. This approach could potentially address the issue of party manifestos being overly complex and lacking in clarity. Further research is needed to determine the effectiveness of this approach and how it compares to sentence-level analysis.

In order to identify input patterns that are more effective at predicting party stances, it may also be useful to experiment with even more of them. Small adjustments to the current patterns in addition to completely different patterns may have an impact on the performance. For instance, it might be beneficial to remove or swap out the square brackets around the context for something else.

The same is true for models. Although the study used the well-known NLP models ELECTRA (Clark et al., 2020) and BERT (Devlin et al., 2018), there are more recent and superior models available, such as, for example, T5 (Xue et al., 2020). It would be beneficial to test out various models and hyperparameters to see which ones work best

for this task, with a focus on finding models that are suited for the German language. However, these models may be computationally expensive, not easily fine-tunable, or not available open source. These are limitations that have to be considered.
Generally speaking, before a model used for predicting political stances can be put to use, its performance must first improve from the results of these experiments. It is crucial to consider the potential implications of using the proposed model in political decision-making. Given the sensitivity of politics, it is essential that the model is capable of accurately predicting political stances and is reliable enough to handle unforeseen scenarios without generating inaccurate results. If the model were to be utilized as a decision aid for voting, the consequences of its errors could be severe. By predicting the wrong parties, the algorithm would be able to change votes. Therefore, it is also imperative to investigate any potential biases in the model to ensure its reliability and accuracy (as done with ChatGPT in Hartmann et al. (2023) for example).
It's also important to keep in mind that party politics can change over time and can differ in different states or regions. The suggested models face a challenge because, in order to remain accurate, they must be able to adjust to shifting political dynamics. It might be necessary to continually update the model with the most recent political data and train it in various political contexts in order to address this challenge. Incorporating feedback mechanisms that enable users to report any errors or discrepancies in the model's predictions may also be beneficial. Doing so could help the model perform better over time.
To improve model performance, the thesis proposes adding context. However, since new models are constantly being released, things change. Models like ChatGPT (OpenAI, 2022) might not need additional context. Therefore, it may be worthwhile to consider the possibility of completely discarding the notion of semantic search and input patterns and instead looking into whether models like ChatGPT (OpenAI, 2022) outperform fine-tuned models on tasks like political stance prediction. This would require a shift in research focus towards understanding how to obtain the best results from such models and ensuring that these results are reliable and unbiased.

7. Conclusion

In order to increase the accuracy of large-scale language models for predicting party stance positions, this thesis looked into the possibility of using relevant information from political party manifestos. The use of NLP models in the field of politics has gained significant interest, and this work contributes to the development of such models.
In order to predict whether political parties would agree with a statement, the models proposed in this study use user-generated statements and relevant sentences from party manifestos as inputs. Sentences from the 2021 party manifestos were chosen using eight different semantic search techniques, namely BM25, SBERT, BERT-Flow (and SBERT-BERT-Flow), Whitening (and SBERT-Whitening), SBERT-WK, and IS-BERT. The effectiveness of seven different input patterns and two language models, ELECTRA (Clark et al., 2020) and BERT (Devlin et al., 2018), was investigated.
The results show that although including information from political party manifestos could occasionally increase the model's capacity to predict party stances, the improvement was not substantial. The performance of different semantic search techniques varied, with IS-BERT providing better results than other techniques, but none showed substantial improvement. The performance of the model is not seriously affected by the various input patterns. The study also discovered that ELECTRA performed better on this task than BERT.
One possible explanation for the lack of improvement after adding context using party manifestos is the models' inability to handle the added complexity or the presence of ambiguous language in the manifestos. The study only examined political party manifestos written in German, so it's important to keep in mind that the findings might not apply to other languages or contexts. Additionally, manifestos might not always include all the necessary information, so expanding the model's data set to include information from other sources like speeches and news articles could increase its accuracy. Better semantic search techniques are also required, especially for German, and testing out various models and hyperparameters may help determine a more effective strategy.

A. Appendix

A.1. Electronic Appendix

To access all material relevant to this thesis, please refer to the respective GitLab repository which is available under: `https://gitlab.lrz.de/statistics/winter22/ma_schulzvanheyden`
It contains the following:

- All of the data used.
- The embeddings from the semantic search methods.
- The code that is necessary to reproduce the analysis of this thesis with additional instructional clues on how to utilize it.
- A folder with the LaTeX project.

A.2. F1 Score Results

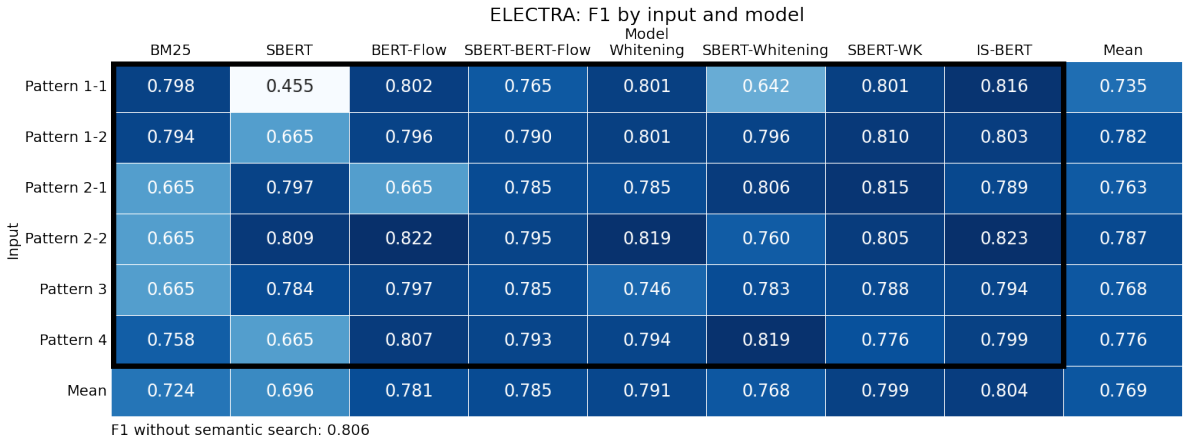

Figure A.1.: The F1 score calculated on the test data for all models with ELECTRA as the base model

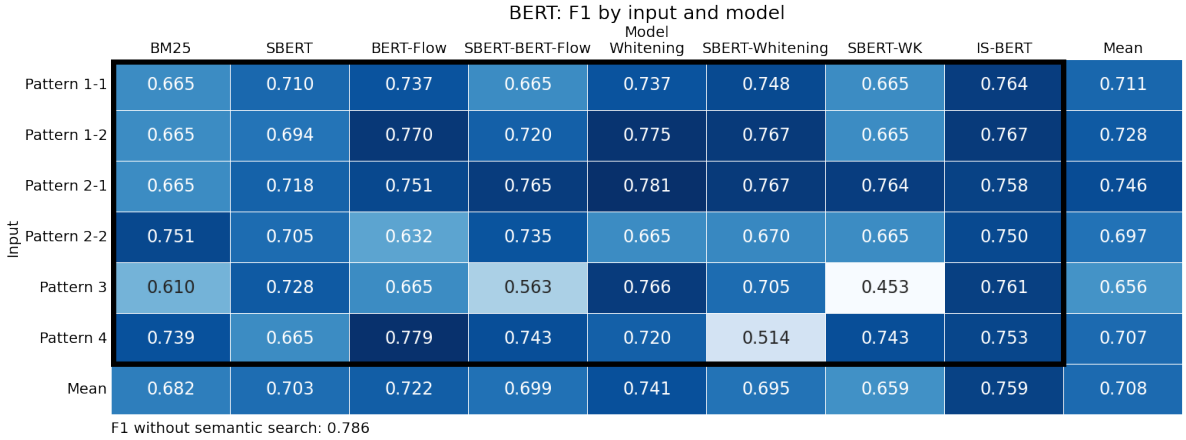

Figure A.2.: The F1 score calculated on the test data for all models with BERT as the base model

The results for ELECTRA using the F1 score shown in Figure A.1 mirrored those of the accuracy scores seen in Chapter 5, with IS-BERT achieving the highest average F1 score of 0.804, while SBERT models gave the worst results. The most effective pattern was Pattern 2-2, whereas the worst-performing pattern was Pattern 1-1.

When looking at the BERT based models in Figure A.2, the results are also consistent with the accuracy findings. IS-BERT models exhibit the best performance with a mean F1 score of 0.759, while SBERT-WK models perform worse than BM25. Among the input

patterns, Pattern 2-1 maintains its position as the best-performing pattern, followed by Pattern 1-2 and Pattern 1-1.

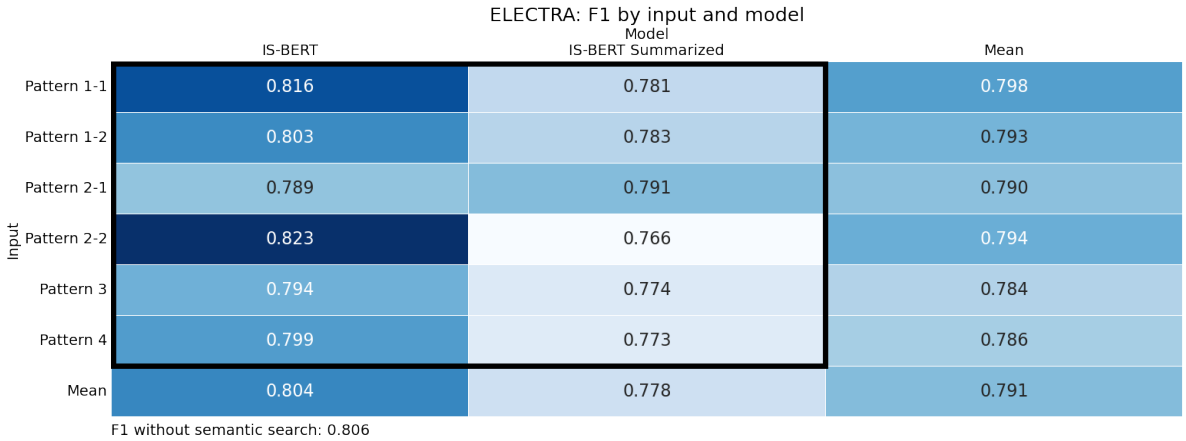

Figure A.3.: The F1 score calculated on the test data for the IS-BERT and IS-BERT Summarized models (with ELECTRA as the base model)

The results of Figure A.3's F1 score analysis for the summarization experiment support the accuracy's findings. On average, IS-BERT models with summarization perform worse than IS-BERT models without summarization. With an F1 score of 0.791 versus 0.789, the best model (Pattern 2-1) only marginally outperforms the worst IS-BERT model. Other than that, all models perform worse.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., et al. (2022). Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074.

BBC (2020). Germany's afd: How right-wing is nationalist alternative for germany? URL: `https://www.bbc.com/news/world-europe-37274201`.

BigScience (2022). Introducing the world's largest open multilingual language model: Bloom. URL: `https://bigscience.huggingface.co/blog/bloom`.

Brown, D. (2020). Rank-BM25: A Collection of BM25 Algorithms in Python.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bundeszentrale für politische Bildung (2021). Wahl-o-mat. URL: `https://www.bpb.de/themen/wahl-o-mat/`.

Chan, B., Möller, T., Pietsch, M., and Soni, T. (2019). bert-base-german-cased · hugging face. URL: `https://huggingface.co/bert-base-german-cased`.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Conradt, D. P. (2013). Christian social union. URL: `https://www.britannica.com/topic/Christian-Social-Union`.

Conradt, D. P. (2021). Christian democratic union. URL: `https://www.britannica.com/topic/Christian-Democratic-Union-political-party-Germany`.

Decker, F. (2021). Kurz und bündig: Die grünen: Parteien in deutschland: Bpb. URL: `https://www.bpb.de/themen/parteien/parteien-in-deutschland/gruene/42149/kurz-und-buendig-die-gruenen/`.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

dontloo (2019). What exactly are keys, queries, and values in attention mechanisms? Cross Validated. URL: https://stats.stackexchange.com/q/424127 (version: 2021-12-31).

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Gupta, P., Tsvetkov, Y., and Bigham, J. P. (2021). Synthesizing adversarial negative responses for robust response ranking and evaluation. *arXiv preprint arXiv:2106.05894*.

Hartmann, J., Schwenzow, J., and Witte, M. (2023). The political ideology of conversational ai: Converging evidence on chatgpt's pro-environmental, left-libertarian orientation. *arXiv preprint arXiv:2301.01768*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, J., Tang, D., Zhong, W., Lu, S., Shou, L., Gong, M., Jiang, D., and Duan, N. (2021). Whiteningbert: An easy unsupervised sentence embedding approach. *arXiv preprint arXiv:2104.01767*.

Iyyer, M., Enns, P., Boyd-Graber, J., and Resnik, P. (2014). Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1113–1122.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31.

Kotamraju, S. (2022). An intuitive explanation of sentence-bert. URL: `https://towardsdatascience.com/an-intuitive-explanation-of-sentenc e-bert-1984d144a868`.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., and Li, L. (2020). On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*.

Lv, Y. and Zhai, C. (2011). Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 7–16.

Marschall, S. (2022). Universität düsseldorf: Facts about the wahl-o-mat. URL: `https://www.sozwiss.hhu.de/en/institut/abteilungen/politikwissenschaft /politik-ii/prof-dr-stefan-marschall/forschungsprojekte/wahl-o-mat-res earch/facts-about-the-wahl-o-mat`.

May, P. (2019). German ELMo Model. URL: `https://github.com/t-systems-on-s ite-services-gmbh/german-elmo-model`.

May, P. and Reißel, P. (2020). bert-base-german-cased · hugging face. URL: `https: //huggingface.co/german-nlp-group/electra-base-german-uncased`.

Murphy, M. (2016). Mit built a donald trump ai twitter bot that sounds scarily like him. URL: `https://qz.com/631497/mit-built-a-donald-trump-ai-twitter-bot-tha t-sounds-scarily-like-him`.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

OpenAI (2022). Chatgpt: Optimizing language models for dialogue. URL: https://openai.com/blog/chatgpt/, Accessed: 2023-01-10.

Ostendorff, M. (2022). Bloom-clp german (6.4b parameters). URL: https://huggingface.co/malteos/bloom-6b4-clp-german.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *CoRR*, abs/1802.05365.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Reimers, N. (2022). Pretrained models. URL: https://sbert.net/docs/pretrained_models.html.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Reimers, N. and Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Reimers, N. and Kayid, A. (2022). Bonjour. مرحبا. guten tag. hola. cohere's multilingual text understanding model is now available. URL: https://txt.cohere.ai/multilingual/, Accessed: 2023-03-30.

Roberts, A. (2020). Exploring transfer learning with t5: The text-to-text transfer transformer. URL: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995). Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Rohmann, S. (2009). Was ist der wahl-o-mat? URL: https://www.uni-siegen.de/wahlomat/was_ist.html.

Schmid, P. (2023). Igel: Instruction-tuned german large language model for text. URL: https://huggingface.co/philschmid/instruct-igel-001.

Schubert, K. and Klein, M. (2016). Das politiklexikon. 6., aktual. u. erw. *Aufl., Bonn: Dietz*.

Sharma, P. and Moh, T.-S. (2016). Prediction of indian election using sentiment analysis on hindi twitter. In *2016 IEEE international conference on big data (big data)*, pages 1966–1971. IEEE.

Shliazhko, O., Fenogenova, A., Tikhonova, M., Mikhailov, V., Kozlova, A., and Shavrina, T. (2022). mgpt: Few-shot learners go multilingual. *arXiv preprint arXiv:2204.07580*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Tech Advisor (2019). How tech giants are investing in artificial intelligence. URL: https://www.techadvisor.com/article/724782/how-tech-giants-are-investing-in-artificial-intelligence.html, Accessed: 2023-04-03.

Trotman, A., Puurula, A., and Burgess, B. (2014). Improvements to bm25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*, pages 58–65.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, B. (2020). Sbert-wk: A sentence embedding method by dissecting bert-based word models. URL: https://github.com/BinWang28/SBERT-WK-Sentence-Embedding.

Wang, B. and Kuo, C.-C. J. (2020). Sbert-wk: A sentence embedding method by dissecting bert-based word models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2146–2157.

Wang, K. (2021). Pytorch - bertflow. URL: https://github.com/UKPLab/pytorch-bertflow.

Wikipedia (2022a). Freie demokratische partei. URL: https://de.wikipedia.org/wiki/Freie_Demokratische_Partei.

Wikipedia (2022b). The left (germany). URL: https://en.wikipedia.org/wiki/The_Left_(Germany).

Wikipedia (2022c). Liste der politischen parteien in deutschland. URL: https://de.wikipedia.org/wiki/Liste_der_politischen_Parteien_in_Deutschland.

Witte, M., Schwenzow, J., Heitmann, M., and Assenmacher, M. (2022). Wahl-o-bot: Der ki wahlcheck. URL: https://wahl-o-bot.de/Methodik.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2020). mt5: A massively multilingual pre-trained text-to-text transformer. *CoRR*, abs/2010.11934.

Zhang, Y. (2020). Is-bert. URL: https://github.com/yanzhangnlp/IS-BERT.

Zhang, Y., He, R., Liu, Z., Lim, K. H., and Bing, L. (2020). An unsupervised sentence embedding method by mutual information maximization. *arXiv preprint arXiv:2009.12061*.

Declaration of Authenticity

The work contained in this thesis is original and has not been previously submitted for examination which has led to the award of a degree.

To the best of my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made.

Lea Schulz-Vanheyden