Bachelor Thesis

Application of neural topic models to twitter data from German politicians

Author

Anne Gritto

Supervisors

Matthias Aßenmacher
Prof. Dr. Christian Heumann
Department of Statistics

Prof. Dr. Paul Thurner
Department of Political Science

Department of Statistics

Ludwig-Maximilians-Universität München

Munich, 23 February 2022

Abstract

Topic modelling has become an important tool in the field of Natural Language Processing in recent years. Since we have arrived in a time, where large amounts of data are collected every day, we need tools that are able to understand, organize and label this mass of data. Topic models are capable of automatically extracting meaning from text data by identifying their topics. While the focus in the field of topic modelling has long been on Bayesian methods such as Latent Dirichlet Allocation, a new research area emerged that is known as neural topic modelling. In the scope of this thesis, the neural topic model BERTopic is applied to Twitter data written by German politicians. It leverages transformer as well as BERT embeddings in order to generate meaningful topics, although other embedding methods can also be applied. In this thesis, we used three different embedding models: Sentence BERT, German BERT and GottBERT. Our results show that the topics generated with Sentence BERT as embedding model were coherent and meaningful, whereas the other two models were not able to create cohesive topics.

Contents

List of Figures

List of Tables

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
biRNN	Bidirectional Recurrent Neural Network
BOW	Bag-of-words
CBOW	Continuous Bag-of-words
CNN	Convolutional Neural Network
c-TF-IDF	class-based Term Frequency - Inverse Document Frequency
DBOW	Distributed Bag-of-words
DM	Distributed Memory
FFNN	Feedforward Neural Network
GRU	Gated Recurrent Unit
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory
MLM	Masked Language Model
MMR	Maximal Marginal Relevance
NLP	Natural Language Processing
NSP	Next Sentence Prediction
NTM	Neural Topic Model
PV	Paragraph Vector
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT Pretraining Approach
SBERT	Sentence-BERT
SGD	Stochastic Gradient Descent
TF-IDF	Term Frequency - Inverse Document Frequency
UMAP	Uniform Manifold Approximation and Projection

1. Introduction

Humans have many different ways to communicate with each other, for example by using gesture, facial expressions and also language. With human language, one can ask questions, give instructions or express feelings and it is "one of the most complex tools used by humans" (Pilehvar and Camacho-Collados, 2020). Natural Language Processing (NLP) describes methods and techniques for machine processing and understanding of natural, human language. This allows speech-oriented interactions between individuals and computers.

NLP is a subfield of Artificial Intelligence as computers 'behave intelligent' by being able to process text or voice data and also to understand its meaning. There are many tasks that can be achieved using NLP, such as sentiment analysis, text classification, semantic relatedness or topic modelling. The latter is the subject of this thesis.

As the amount of data has grown rapidly in recent years, it is difficult to obtain relevant and structured information (Bansal, 2016). While humans would struggle to label thousands of documents or find specific documents given a query, a topic model is able to extract the most important information of texts by assigning documents to specific topics. It is an approach of unsupervised learning and is a widely used technique in document clustering or information retrieval from unstructured texts (Arun et al., 2010).

Although there exist various methods for topic modelling, the focus in this thesis will be on transformer-based models which consist amongst other things of neural networks. Since the introduction of Transformer models by Vaswani et al. (2017), they have shown amazing results in many NLP-related tasks. Especially pre-trained models have much power as they already hold accurate representations of words and sentences (Grootendorst, 2020b).

This thesis first gives an introduction to a topic model that does not rely on neural networks, but on Bayesian methods. Then, the basis for topic modelling with large pre-trained models is set by explaining neural networks in general, their application in NLP and their impact on the Transformer. It continues to describe a neural topic model by Grootendorst (2020a), called BERTopic, and its algorithm which will be applied in the course of this thesis to twitter data from German politicians. The results will be evaluated in Chapter 4 and the thesis concludes with some future outlook.

2. General methodological background

Topic modelling is a part of natural language processing which comprises many tasks. The goal is to extract meaningful topics from given text data. While these topics may be known prior to the modelling in some examples, this is not the case for every data. If one does not know the real labels, as is the case in this thesis, topic modelling is a task of unsupervised learning.

Until this day, various topic models have already been introduced with different approaches. One of the most famous ones is Latent Dirichlet Allocation (LDA) which will be explained in section 2.1. It is a generative, probabilistic model and is based on Bayesian methods. In contrast to that are topic models that use neural networks to accomplish their task. For this approach, words or texts need to be mapped to numerical vector spaces, which is known as word and text embedding, respectively. This enables machine learning models to process these embeddings, e.g., cluster embeddings of text to generate topics. There are several ways to embed words, sentences and documents. The most intuitive one is the Bag-of-words (BOW) approach. The basic idea is that a vocabulary is learned which consists of the words in each text. A text can be a sentence, one or more paragraphs, or documents. Therefore, one can think of this approach as a bag that contains all unique words which are used and so any information about grammar or the structure or order of words in the text is discarded. The model does not know where words are in the document, it only concerns whether known words appear in the document. While the bag-of-words approach can be used to generate text embeddings, it is moreover the basis for other models such as LDA.

This chapter first gives an overview of LDA in section 2.1, the topic modelling technique that is a part of probabilistic modelling. It assumes that each document exhibits multiple topics, therefore a document can be described as a distribution over topics.

Additionally, each topic is given by a distribution over words. While the documents are observed, the topic structure is unknown beforehand and needs to be inferred. As LDA is a probabilistic model, the hidden topic structure can be computed using its conditional distribution given the documents. This is also known as calculating the posterior distribution (Blei, 2012) in Bayesian statistics.

Following that part, a general introduction to neural networks and their role in embedding documents will be given in order to understand more complex tools such as Bidirectional Encoder Representations from Transformers (BERT). These transformer-based models have set a milestone in NLP and are the basis of topic models that are applied in this thesis.

2.1. A Bayesian approach: Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) which has been introduced by Blei et al. (2003) is a generative probabilistic model that is able to find latent, hidden topics in text corpora. While this is limited to text data here, LDA can also be applied to general collections of discrete data. The basic idea is that each document can be generated as a mixture of topics and that each topic is described by a distribution over words. Specifically, "documents are modelled via a hidden Dirichlet random variable that specifies a probability distribution on a latent, low-dimensional topic space" (Blei et al., 2003). This means that each document consists of multiple topics. In addition, the underlying topics are the same for every document in a text corpus, but each document possesses different proportions of these topics. Another assumption the LDA model makes is that both, topics and words, are exchangeable within a document.

Let a vocabulary of V words be given, where each word is represented by a unit-basis, sparse vector. A document consists of N words $d = \{w_1, ..., w_N\}$, a corpus is then the collection of M documents $D = \{d_1, ..., d_M\}$ and k is defined as the number of topics. The LDA model assumes the documents to arise by a generative process displayed in algorithm 1. In the first place, the number of words N per document needs to be determined with the help of a Poisson distribution. After that, the topic proportions θ for a document d, are sampled by using a Dirichlet distribution with the parameter α that defines the per-document topic distributions. For a small α value, the documents are likely to consist of only a few topics and as α gets bigger, the documents will have more of a topic mixture. The sum of the parameter θ equals 1 over all topics per document, $\sum_{i=1}^{k} \theta_i = 1$ with $\theta_i \geq 0$. The documents with N words are then generated iteratively: first, a topic z_n is picked that represents the topic for the n^{th} word of document d by using a Multinomial distribution with parameter θ. Finally, a word w_n is chosen given the selected topic and β, which is a distribution over the vocabulary for each topic. Therefore, β can also be represented by a $k \times V$ matrix with an entry being the probability of a certain word w_j occurring in a document given a topic z_i, $\beta_{ij} = p(w_j = 1 | z_i = 1)$. Furthermore, β is the parameter of a Dirichlet distribution and controls the distribution of words per topic. The topics will be represented by only a few words if β is small and, in turn, consist of many words if β is large. The resulting documents do not have correct grammar or syntax, but they contain representative words of the distributions of topics that were chosen to be in a document. Therefore, the LDA model is based on the bag-of-words assumption that ignores the order of words in a document (Blei et al., 2003; Blei, 2012).

Algorithm 1: Generative process of LDA for each document d in a corpus D, see Blei et al. (2003)

```
Choose number of words $N$ ~ Poisson($\xi$) in document;
Choose a topic mixture for the document $\theta$ ~ Dir($\alpha$);
for each of the $N$ words $w_n$ in the document do
    Pick a topic $z_n$ ~ Multinomial($\theta$);
    Choose a word $w_n$ based on $p(w_n|z_n, \beta)$, the topic‘s $z_n$ multinomial distribution;
end
```

This algorithm describes how documents can be generated. However, the documents are given to the algorithm as input in real life and the goal is to learn the topics as well as the word distribution of each topic. Blei (2012) describes this process as follows:

> “[...] the goal of topic modeling is to automatically discover the topics from a collection of documents. The documents themselves are observed, while the topic structure — the topics, per-document topic distributions, and the per-document per-word topic assignments — is *hidden structure*. The central computational problem for topic modeling is to use the observed documents to infer the hidden topic structure. This can be thought of as ‘reversing‘ the generative process — what is the hidden structure that likely generated the

observed collection¿‘

As mentioned before, LDA is a generative, probabilistic model which contains observed as well as hidden variables. The goal is to find the probabilities of these latent variables given the observed data by computing the posterior distribution of the hidden random variables given the observed documents. This is also called a conditional distribution (Blei et al., 2003) and is given by

$$p(\theta, z \mid d, \alpha, \beta) \;=\; \frac{p(\theta, z, d \mid \alpha, \beta)}{p(d \mid \alpha, \beta)}. \tag{2.1}$$

The numerator is the joint distribution of the topic proportions θ, a set of topics z and a document d given the parameters α and β and is according to Blei et al. (2003) defined as

$$p(\theta, z, d \mid \alpha, \beta) \;=\; p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta)\, p(w_n \mid z_n, \beta).$$

The denominator of equation 2.1, $p(d \mid \alpha, \beta)$, is the probability that the real document can be recreated, but under any topic model. This probability could be computed theoretically “by summing the joint distribution over every possible instantiation of the hidden topic structure“ (Blei, 2012). However, this is intractable to compute and thus, Blei et al. (2003) propose to use approximate inference algorithms such as Laplace approximation, variational approximation and Markov chain Monte Carlo (Blei et al., 2003; Jordan et al., 1999).

Since its introduction, LDA has been an effective tool for topic modelling and has shown good results in many fields (Blei, 2012). Besides, it is easy to apply as there are implementations in Python, e.g. in the package `scikit-learn` by Pedregosa et al. (2011). Another advantage of LDA is that it can also contribute to more complicated goals by extending and adapting the model in different ways. However, LDA suffers from some drawbacks as well, such as determining the number of topics K in advance. While the bag-of-words assumption can be accepted when doing topic modelling, it still loses the order of words and therefore also its context and could not be used, for example, for language generation (Blei, 2012). Apart from LDA, which is a Bayesian approach, neural networks have gained a lot of attention in recent years to perform topic modelling, as those are able to capture the semantics and context of words in documents.

2.2. Feedforward neural network

The beginning of neural networks can be traced back to the 1940s, when neurophysiologist, Warren McCulloch, and mathematician, Walter Pitts, described how neurons in the brain work. These neurons are attached and operate as information messengers. A neuron can initiate an electrical impulse when some threshold of excitation is exceeded (Pitts and McCulloch, 1943). This enables neurons "to transmit information between different areas of the brain, and between the brain and the rest of the nervous system" (National Institute of Neurological Disorders and Stroke, 2002). Neural networks are inspired by this computation mechanism of the brain.

A Feedforward Neural Network (FFNN) takes an input, gives that through a number of hidden layers which are non-linear functions to pass it to its output. Figure 2.1 shows a typical neural network with two hidden layers. Every circle can be thought of as neurons that are arranged in layers and each neuron of a layer is connected to those of the adjoined layer. Therefore, this FFNN is fully connected. Each neuron is connected by arrows which all carry a weight that reflect its importance. Furthermore, "the sigmoid shape inside the neurons in the middle layer represents a non-linear function [...] that is applied to the neuron's value before passing it to the output" (Goldberg, 2015). The layers can be represented as vectors which may have different dimensions. In figure 2.1, e.g., the input vector $x = (x_1, ..., x_4)$ has 4 dimensions, while the first and second hidden layer consist of 6 and 5 layers respectively, and the output layer $y = (y_1, y_2, y_3)$ then only has 3 dimensions. One can think of the first hidden layer as a linear transformation that starts with 4 dimensions and results in 6 dimensions.

The hidden layers are represented as functions which need to be approximated. Specifically, a good mapping $\hat{y} = f(x; \theta)$ needs to be found where θ is learned by the neural network. As the neural network in figure 2.1 contains two hidden layers, it has two functions $f^{(1)}$ and $f^{(2)}$ which are attached in a chain to form $f(x) = f^{(2)}(f^{(1)}(x))$. The depth of the model is given by the length of this chain (Goodfellow et al., 2016, Chapter 6).

The simplest FFNN is the perceptron which can be seen as a linear function $f(x; \theta)$ of the input:

$$f(x; w, b) = x^T w + b$$

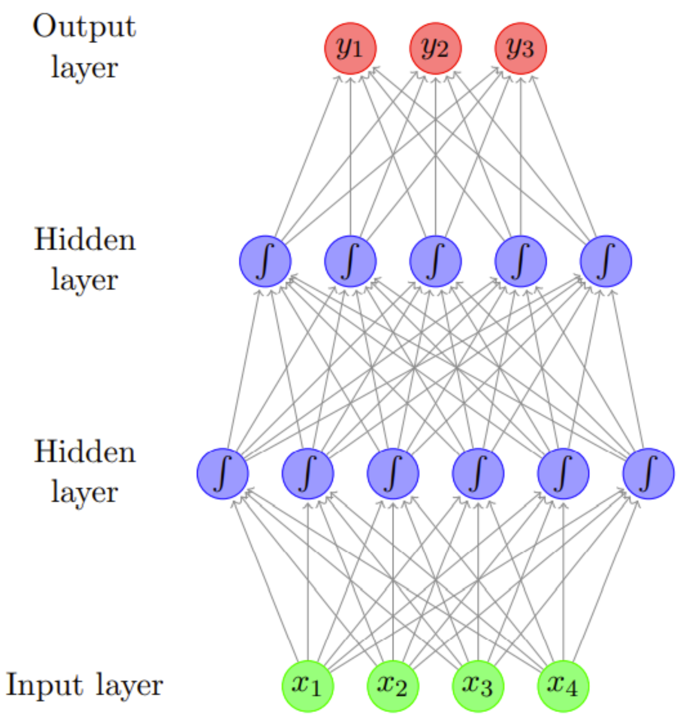


Figure 2.1.: Feedforward neural network with an input layer, two hidden layers and an output layer by Goldberg (2015)

where w is the weight matrix and b a bias term (Goodfellow et al., 2016, Chapter 6). To describe the features adequately, a non-linear function must be used which is called an activation function ϕ. The activation function for a vector of hidden units is defined as

$$h = \phi(W^T x + c)$$

with W being the weights of a linear transformation and c the biases, therefore a vector of biases b. While W and c are learned during training, the function ϕ is given and can have different forms, e.g., sigmoid, tanh or rectified linear unit (ReLU). The latter is simple to work with, computationally cheap and also produces excellent results. It is defined as

$$\phi(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x}) = \max(0, \mathbf{x}) = \begin{cases} 0 & x < 0 \\ x & otherwise. \end{cases} \tag{2.2}$$

A FFNN is called *feedforward* because the information flows only in one direction and there exist thus no feedback connections. If a neural network has feedback connections, it is called a recurrent neural network (RNN).

Training a neural network with stochastic gradient descent

Training a FFNN is similar to training a typical machine learning model with gradient descent (Nielsen, 2015). Therefore, a cost function

$$J(\theta) = \mathbb{E}_{(x,y)}\, \mathcal{L}(\hat{y}_i, y_i) = \mathbb{E}_{(x,y)}\, \mathcal{L}(f(x_i|\theta), y_i)$$

between predicted $\hat{y}_i$ and actual values y_i needs to be minimized, where $\mathcal{L}$ is the loss function. The goal is to find those parameters θ of the neural network that minimize the cost function $J(\theta)$ (Nielsen, 2015). The loss function that is used to train neural networks relies typically on maximum likelihood estimation and is the negative log-likelihood $\mathcal{L}(f(x_i|\theta), y_i) = -\log p(y_i|x_i;\theta)$.

To minimize $J(\theta)$ with respect to its parameters θ, an iterative algorithm called Stochastic Gradient Descent (SGD) can be used. Algorithm 2 gives an overview of SGD. First, a random data point is sampled from the training data set. Second, the loss function needs to be computed and therefore also the prediction $\hat{y} = f(x|\theta_t)$ based on the initial parameter θ_t. The gradient is then obtained via backpropagation (Rumelhart et al., 1986) which calculates the derivatives of the loss function by using the chain rule. Lastly, the parameters θ_t are updated in the direction of the gradient and scaled with a learning rate η (Goldberg, 2015).

Algorithm 2: Training a neural network with stochastic gradient descent and mini-batch size of 1. Algorithm is adopted from Goldberg (2015).

```
while stopping criteria have not met do
    1. Randomly sample a data point $(x, y)$;
    2. Compute loss function based on parameter estimation $\theta_t$
       $\mathcal{L}(\hat{y}, y) = \mathcal{L}(f(x|\theta_t), y) = \log p(y_t|x_t;\theta)$;
    3. Get the gradient of the loss function w.r.t $\theta$
       $g_{t+1} = \nabla\mathcal{L}(\hat{y}, y) = \left(\frac{\partial}{\partial\theta_j}\mathcal{L}(\hat{y}, y)\right)_{j=1,\dots,r}$;
    4. Update the parameter with the gradient $g_{t+1}$ and a learning rate $\eta$
       $\theta_{t+1} \leftarrow \theta_t - \eta\, g_{t+1}$;
end
```

The following chapter now presents ways to embed words and texts in numerical vectors that are based on neural networks.

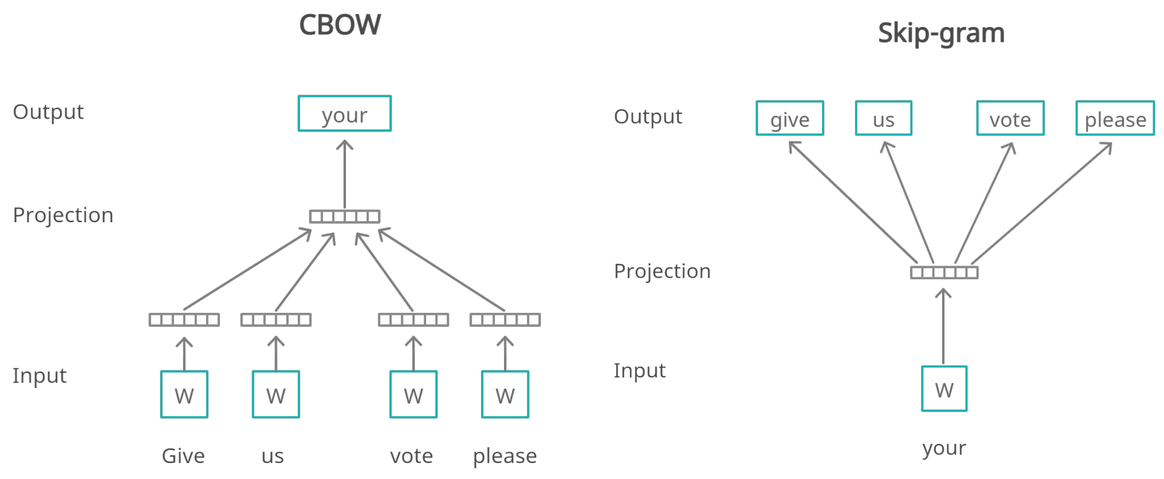
2.3. Distributed representations of words, sentences and documents

There exist several ways to transform text data into numerical vectors in order to feed them to other machine learning algorithms. The most simple one is the bag-of-words approach which was shortly described at the beginning of this chapter. Although it is intuitive to understand and easy to implement, it loses the order of words and therefore, the context of a document. Methods that are based on neural networks are more complex on the one hand. But on the other hand, they are able to capture the semantics and contexts of words. One important component in neural networks for language is the usage of an embedding layer in which words, sentences or texts are mapped to continuous vectors in a relatively low-dimensional space. Machine learning algorithms can then operate with these embeddings because they are numerical representations of symbols. This representation of phrases is learned by the neural network while training (Goldberg and Hirst, 2017).

Rumelhart et al. (1986) first proposed distributed representations for words which was the beginning of various other methods. One successful technique to embed words was then introduced by Mikolov et al. (2013a) known as Word2Vec. One advantage of distributed representations of words is that words with similar meanings are close to each other in the vector space. For example, the representations of *Germany* and *France* are close to each other, whereas *Germany* and *apple* are more distant. After that milestone in NLP was set, other forms of embedding methods were researched, such as sentence and document embeddings.

2.3.1. Word2Vec

Word2Vec learns vector representations for words by using a simple feedforward neural architecture that is trained with language modelling objective (Pilehvar and Camacho-Collados, 2020; Mikolov et al., 2013b). Every word is mapped to a unique vector and is represented by a column in the weight matrix W. Word2Vec is also known as a predictive model as the task is to predict a word given the other words in a context (Le and Mikolov, 2014). Unlike bag-of-words, Word2Vec captures the context and semantics of a word, e.g., the representations of *vector(King)* - *vector(Man)* + *vector(Woman)* result in a vector that is very close to *vector(Queen)* (Mikolov et al., 2013c).



(a) The framework of CBOW combines the vector representations of the surrounding words to predict the middle word.

(b) The framwork of Skip-gram uses the distributed representation of an input word to predict its context.

Figure 2.2.: Example of the two word2vec architectures, CBOW and Skip-gram, proposed by Mikolov et al. (2013a).

Two different Word2Vec models were proposed: Continuous Bag-of-words (CBOW) and Skip-gram which are depicted in figure 2.2. The training objective of the CBOW model is to join the representations of surrounding words, the context, to predict the current middle word. This can be achieved according to Le and Mikolov (2014) by maximizing the average log probability, given a sequence of words $w_1, w_2, ..., w_T$, as

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, ..., w_{t+k})$$

with w_t being the target word and

$$p(w_t | w_{t-k}, ..., w_{t+k}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)} .$$

The Skip-gram model is the counterpart of Continuous Bag-of-words. Its aim is to predict the context given the distributed representation of the input word. The objective of the the Skip-gram model is according to Mikolov et al. (2013b) also to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,\, j\neq 0}\log p(w_{t+j}|w_t)$$

where c denotes the size of the context and $p(w_{t+j}|w_t)$ is defined by using the softmax function

$$p(w_O|w_I) \;=\; \frac{\exp\left({v'_{w_O}}^T v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_{w_O}}^T v_{w_I}\right)}$$

with v_w and v'_w being the input and output vector representation of words and W being the number of words in the vocabulary. The architectures of these models are shown in figure 2.2. Both, CBOW and Skip-gram, are trained by using stochastic gradient descent and the gradient is then obtained via backpropagation (Rumelhart et al., 1986).

2.3.2. Paragraph Vector

Le and Mikolov (2014) introduced Paragraph Vector (PV), which is also known as Doc2Vec, to predict words in a given paragraph. It is an unsupervised algorithm that learns continuous distributed vector representations of texts. These texts may differ in length and can therefore be sentences, paragraphs or large documents as well. Doc2Vec is inspired by the method for learning word vectors but adds a unique vector for each paragraph. There are two different approaches of Paragraph Vector: Distributed Memory and Distributed Bag-of-words.

In the Distributed Memory (DM) model of Paragraph Vectors (PV-DM), paragraph vectors and word vectors both contribute to predicting the next word given several contexts which are sampled from the document. This framework is depicted in figure 2.3. Every paragraph gets a vector, which is represented by a column in matrix D. The paragraph vectors are unique among paragraphs, while the word vectors, which are vectors represented by a column in matrix W, are shared. The paragraph vectors and word vectors are either concatenated, averaged or summed to predict the next word in a context.

This approach is called Distributed Memory as the "paragraph vector represents the

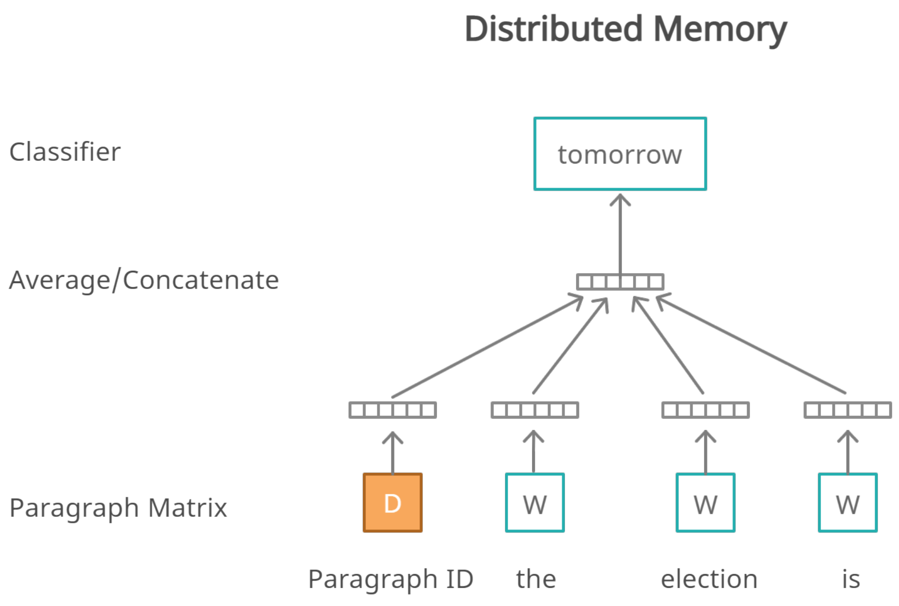


Figure 2.3.: The framework of PV-DM adopted from Le and Mikolov (2014). The concatenation or average of the paragraph vector with the word vectors, which are the context words, is used to predict the forth word.

missing information from the current context and can act as a memory of the topic of the paragraph“ (Le and Mikolov, 2014). The training of word and paragraph vectors is done by using stochastic gradient descent and backpropagation (Rumelhart et al., 1986). At every step of gradient descent, a fixed-length context is sampled from a random paragraph. Then an error gradient is computed which is used to update the parameters in the model. If a paragraph vector needs to be computed for a new paragraph, an inference step is being used, which is also attained by gradient descent. When the training has finished, one can use the paragraph vectors as features to feed to machine learning techniques such as logistic regression (Le and Mikolov, 2014). PV-DM is based on the idea of Continuous bag-of-words which was described in part 2.3.1.

The Distributed Bag-of-words (DBOW) model of Paragraph Vectors (PV-DBOW) loses the order of words as the context words are being ignored. Here, only the paragraph vectors are used to predict words in a small window. Figure 2.4 shows the framework of Distributed bag-of-words that is inspired by Le and Mikolov (2014). It illustrates that only the paragraph vectors of matrix D are used to predict the output. Paragraph vectors are also trained in this approach by gradient descent and backpropagation. At

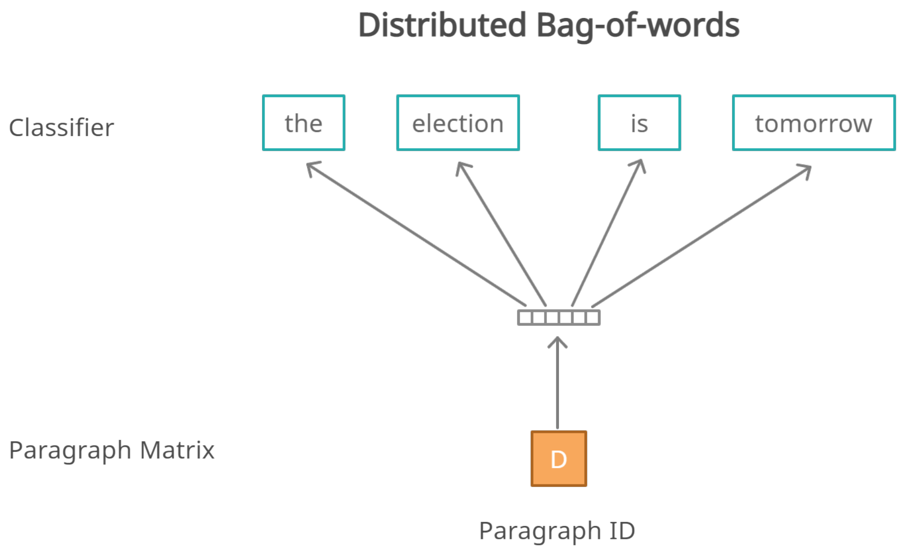


Figure 2.4.: The framework of PV-DBOW adopted from Le and Mikolov (2014). The paragraph vector is trained to predict the words in a context window.

every step of stochastic gradient descent, a text window and a random word from that specific text window are sampled to form a classification task given the paragraph vector. PV-DBOW is similar to the Skip-gram model of Word2Vec. This model is conceptually simple. Furthermore, it does not need to store as much data as Distributed Memory has to because it does not have to learn vector representations of each word. In Le and Mikolov (2014), the authors suggest combining PV-DM and PV-DBOW because its combination is more consistent across many tasks.

2.4. Specialized architectures

Apart from feedforward neural networks that are used, e.g., in Word2Vec or Paragraph Vector presented in section 2.3, other architectures such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) find application in the machine learning context. CNNs, for example, are prevalent in image processing tasks or computer vision. This network uses filters to extract relevant features from the input data. If the input data are images, CNNs capture their spatial features which help to identify objects (Pai, 2020). A feedforward neural network becomes a CNN as soon as one hidden layer is a convolutional layer and has several assumptions. Firstly, it assumes that the same object can be in different areas, therefore the features of an object are not determined by its position in the input. Secondly, a CNN expects meaningful objects to occur in a coherent area. Lastly, with a higher number of convolutional layers, the features become more complex and can capture bigger objects (Goodfellow et al., 2016, Chapter 9). In the following, recurrent neural networks will be explained in more depth as those are often applied in natural language processing tasks.

Recurrent neural networks

While CNNs use convolutional layers to deal with a grid of values, a RNN (Elman, 1990) processes sequences of input data. As text is written and read sequentially, RNNs are particularly important in natural language processing. Especially RNNs with gated architectures such as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) are "powerful at capturing statistical regularities in sequential inputs" (Goldberg and Hirst, 2017). A RNN consists of an input and output layer, as well as of a dynamic number of hidden layers, just like a FFNN. However, a recurrent neural network shares its parameters across different time steps. Furthermore, it accepts a new input at each time step which acts as a memory of the contents of the previous sequences (Ruder, 2019).

Let $x = \{x_1, ..., x_J\}$ be an ordered sequence of inputs, e.g. a sentence with embedded words. The hidden units of the network are updated recursively and the current hidden state h is represented at each time step t as

$$\begin{aligned} h_t &= \tanh(Wx_t + Uh_{t-1} + b) \\ y_t &= \phi_y(Vh_t + c) \end{aligned}$$

where h_0 is initialized with 0 at the beginning, W, U and V are weight matrices and b as well as c denote the biases. One can see here that the previous hidden state h_{t-1} is considered when computing the new hidden state h_t. Furthermore, the RNN can produce an output y_t at each time step, where ϕ_t is an activation function (Ruder, 2019; Goodfellow et al., 2016, Chapter 10).

Recurrent neural networks can be trained via backpropagating through time (Werbos, 1988). However, when training an RNN, one may encounter the problem of vanishing or exploding gradients. During backpropagation, the gradients are multiplied repeatedly with the same values. Gradients that are smaller or greater than 1, will either vanish or explode at some point when going back through the network (Aßenmacher, 2021). Due to the vanishing gradients problem, a model is unable to learn long-range dependencies across time steps which is also known as a short-term memory (Phi, 2018). As RNNs suffer from short term memory, LSTMs and GRUs can be applied to combat that weakness. These two are special recurrent neural networks that are able to learn long-term dependencies by using gates. The gates are tensor operations that learn which information to add to or remove from the hidden state (Phi, 2018; Hochreiter and Schmidhuber, 1997; Cho et al., 2014).

While an RNN is able to model sequential data, it processes this data in a left-to-right manner and therefore only considers the previous words $w_1, ..., w_{t-1}$ of a word w_t. However, future input information $w_{t+1}, ..., w_n$ is also useful for prediction. To overcome this limitation, Schuster and Paliwal (1997) proposed a bidirectional recurrent neural network (biRNN) that adds a hidden layer to the network to process information in a backward direction. Hence, two different representations of the forward and backward RNN can be produced with a biRNN which are then concatenated and form a representation of each token that considers its left and right context (Aßenmacher, 2021; Zhang et al., 2021, Chapter 9).

Encoder-decoder architecture

One task when processing text data is machine translation which can be understood as automatically translating a sequence, e.g., a sentence to another language. As sentences are built differently in languages, the length of the output sequence may have a different length than the input sequence. Encoder-decoder architectures take this requirement into account. The encoder transforms the input of variable sequence length into a state which has a fixed shape. Then, the decoder uses that state to map it to a variable-length output sequence. Sutskever et al. (2014) introduced this architecture, which is also known as sequence-to-sequence modelling, and both, encoder and decoder are designed by using RNNs (Zhang et al., 2021, Chapter 9).

Attention mechanism

Encoder-decoder architectures are neural models that have set an important milestone in NLP and specifically machine translation. Nevertheless, the encoder transforms the input sequence into a fixed-length hidden state which results in a bottleneck when trying to improve the performance of this architecture. For that reason, Bahdanau et al. (2014) proposed the attention mechanism, to allow a model to automatically search for those parts of an input sentence that contain the relevant information to predict a target word. Just as neurons in neural networks, attention-based models are inspired by the brain. For example, although a human receives more visual information every second than the brain can process, one is able to filter and direct one‘s attention to objects of interest. Attention models try to adopt this ability by weighting representations differently, therefore aligning only to those parts that are relevant for prediction (Zhang et al., 2021, Chapter 10). Formally, weights can be computed for every encoded representation in the hidden layers $(h_1, ..., h_{T_{in}})$, also called annotations, where T_{in} denotes the length of the input sequence. The weight α_{ij} for each annotation h_j is given according to Bahdanau et al. (2014) by

$$\alpha_{ij} \;=\; \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_{in}} \exp\left(e_{ik}\right)}, \tag{2.3}$$

where e_{ij} is the alignment calculated by an alignment model $a(.)$ with inputs being the previous hidden state s_{i-1} and the annotation h_j:

$$e_{ij} \;=\; a(s_{i-1}, h_j). \tag{2.4}$$

After that, a context vector c_i can be computed as the weighted sum of the annotations of the input sequence:

$$c_i \;=\; \sum_{j=1}^{T_{in}} \alpha_{ij} h_j. \tag{2.5}$$

This process is described by Bahdanau et al. (2014) as:

> “The probability α_{ij} , or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in

the source sentence into a fixed-length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly.“

An important advantage of the attention mechanism when being compared to RNNs is its remarkable long-term memory. In theory, it has an infinite context window which the model can resort to while generating text. Transformer models, which will be explained in section 2.5, share this advantage as they consist of an attention-based encoder-decoder architecture (Phi, 2020).

2.5. Transformer-based models

Since their introduction by Vaswani et al. (2017), transformer models have been applied in many tasks in the field of NLP such as machine translation or conversational chatbots because they show state-of-the-art results. Transformers have an encoder-decoder architecture (Section 2.4) where both networks, encoder as well as decoder, rely on self-attention and multi-head attention. The principle of attention mechanism was described in the previous section. To achieve self-attention, the embeddings of an input sequence are fed through three distinct linear layers to create a query, key and value vector, denoted as q_i, k_i and v_i respectively of dimensions d_q, d_k and d_v. The attention function is calculated on a set of queries simultaneously and therefore the vectors of query, key and value result in the matrices $Q \in \mathbb{R}^{n_{in} \times d_k}, K \in \mathbb{R}^{n_{in} \times d_k}$ and $V \in \mathbb{R}^{n_{in} \times d_v}$. While the computation of the attention weights α_{ij} and the context vector c_i is equivalent to formulas 2.3 and 2.5, the alignment score is calculated as

$$a(Q, K) = \frac{QK^T}{\sqrt{d_k}} \tag{2.6}$$

Note that the dimensions of the queries and keys, d_q and d_k, must be equal. This attention mechanism is also known as the *scaled dot-product attention* and when putting the pieces of formulas 2.3, 2.5 and 2.6 together, the Attention function is given in matrix notation according to Vaswani et al. (2017) by

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

Generally speaking, the self-attention mechanism empowers the model to associate each individual word in the input to other words in the input and is able to predict which words are likely to occur together (Phi, 2020).

The multi-head attention mechanism is then displayed in figure 2.5. Vaswani et al. (2017) found it beneficial to execute this mechanism of self-attention h times in parallel, rather than conducting a single attention function. This means that the model learns h different projection matrices for values, keys and queries, respectively, which “project the input embeddings into different sub-spaces of the original embedding space“ (Aßenmacher, 2021). The scaled dot-product attention is then computed in parallel and its results are concatenated. Lastly, the concatenated vector is being linearly projected which results in the final values. Both, encoder and decoder, contain self-attention layers.

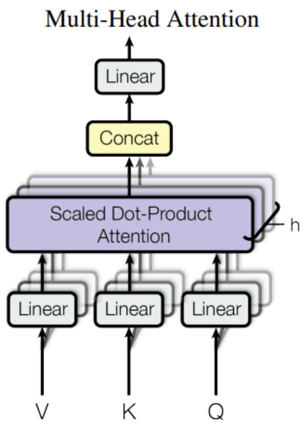


Figure 2.5.: Multi-Head Attention mechanism used in Transformer model of Vaswani et al. (2017).

The architecture of the Transformer model is depicted in figure 2.6 where the encoder is shown on the left half and the decoder on the right. In general, the encoder produces distributed representations $z = (z_1, ..., z_{T_{in}})$ of an input sequence of tokens $(x_1, ..., x_{T_{in}})$. The decoder then uses z to generate the output sequence $(y_1, ..., y_{T_{out}})$, while the previously produced tokens are being fed at each time step as an additional input (Vaswani et al., 2017). In the following, the encoder and afterwards the decoder will be explained in more detail.

The first step of the encoder is to embed an input sequence into vector representations of each token. Due to the lack of the model‘s recurrence or convolution, it would lose information about the order of the sequence. To counteract this, Vaswani et al. (2017) added positional encodings such as

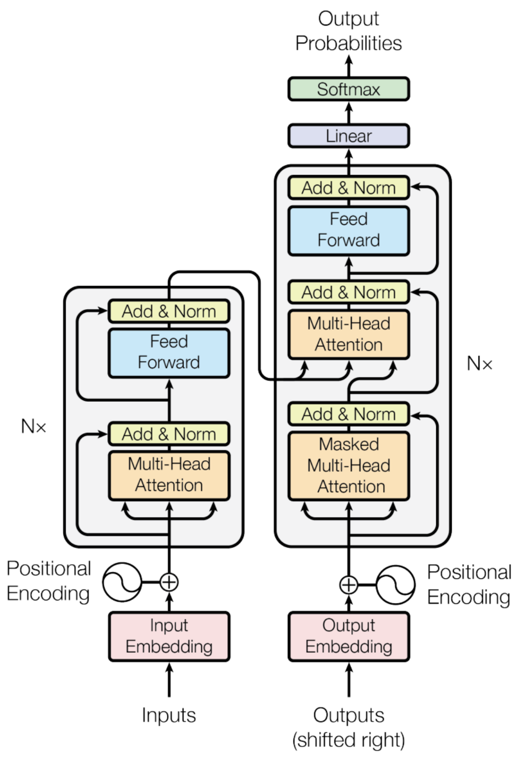


Figure 2.6.: Architecture of the Transformer model which was proposed by Vaswani et al. (2017).

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \end{aligned} \tag{2.7}$$

with pos being the position, i the dimension and d_{model} denotes the output dimension of the encoder. It produces output vectors for even and odd indices of the input vector, respectively, which are then added to their corresponding input embedding (Phi, 2020). The embeddings with their positional encoding are subsequently passed in the encoder layer which is constructed of a stack of six identical layers, where each layer consists of a multi-head self-attention mechanism and a feedfoward neural network. Residual connections (He et al., 2015) are employed around the multi-head attention module as well as the FFNN, which means that the output of each module is added to their corresponding input. Furthermore, the outputs of both residual connections go through a layer normalization (Ba et al., 2016). The outputs of the embedding layers and the modules, multi-head attention mechanism and FFNN, in the encoder are of dimensions $d_{model} = 512$.

The decoder layer also consists of a stack of six identical layers and its construction is similar to the one of the encoder. Each layer consists of two multi-headed self-attention mechanisms and a FFNN, where all modules also have residual connections and a layer normalization. The decoder takes two inputs: the true outputs and the continuous representations generated by the encoder which contains the inputs‘ information (Phi, 2020). The true outputs are first fed into an embedding and positional encoding layer as in formulas 2.7. The resulting embeddings then go through the first multi-head attention module which returns the attention scores for the continuous representations of the encoder. This self-attention mechanism behaves a little different than the other two. As the decoder generates a sequence token by token and is, moreover, autoregressive, it should not consider future tokens. Therefore, a mask is applied on future tokens which prevents them from contributing to the predictions. This first module outputs a masked vector that holds information on how the model should attend to the input. Furthermore, it acts as an input, specifically as the values vector v_i, for the second multi-head attention mechanism. The continuous representations produced by the encoder are the queries and keys in this layer and the decoder can then decide on which inputs to focus (Phi, 2020). The third module is then again a feedforward neural network and the outputs are passed through a linear layer which can be thought of as a classifier. Lastly, that produced output is fed into a softmax layer to produce probability scores.

This concludes the architecture of the Transformer model, a sequence transduction model that is based on attention mechanisms. Originally, the focus of the research of Transformer models was laid on translation tasks, but after that, other influential models based on the Transformer architecture, such as BERT (Bidirectional Encoder Representations from Transformers), were introduced (Carrigan et al., nd).

2.5.1. Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT), proposed by Devlin et al. (2018), relies on a Transformer-based architecture and is able to capture bidirectional context from text data. It has been pre-trained as a language model on large amounts of unlabelled text in a self-supervised manner (Carrigan et al., nd). To exploit its full potential, the model can be fine-tuned on a given downstream task in a supervised way. This process is also called *transfer learning* as the knowledge of the pre-trained model is shifted to a more task-specific knowledge.

A BERT model can have different sizes. In the original paper, Devlin et al. (2018) put their focus on a base and a large model. The base model of BERT consists of 12 Transformer blocks L, a hidden size of $H = 768$ and $A = 12$ self-attention heads, while the large model is trained with $L = 24$, $H = 1024$ and $A = 16$. The vocabulary of BERT is defined by the WordPiece embeddings (Wu et al., 2016) which contains 30000 tokens.

Pre-training BERT

The goal of pre-training is to give the model an understanding of the language and its context. Therefore, BERT is trained in this phase on two unsupervised tasks simultaneously: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). The training objective of MLM is to predict masked tokens given a sentence. The masked tokens all represent a word because a certain percentage of input words are being randomly masked out to train a bidirectional representation (Devlin et al., 2018). When training with NSP, BERT considers two sentences s_1 and s_2 and determines whether s_2 actually follows s_1. This can be understood as a binary classification problem in order to capture the relationship between two sentences. Specifically, 50% of the time the second sentence is the actual next sentence and 50% of the time it is a randomly sampled sentence (Devlin et al., 2018). BERT has been pre-trained as a language model on large amounts of raw text, specifically on the BooksCorpus (Zhu et al., 2015) and English Wikipedia which consist of 800 million and 2500 million words, respectively. The base BERT model results after pre-training in 110 million parameters and the large model in 340 million total parameters (Devlin et al., 2018).

Fine-tuning BERT

Once the BERT model is pre-trained, which needs huge computational cost, it can be fine-tuned on specific tasks which requires less data to get good results. Furthermore, the amount of time and resources that are needed are much lower (Carrigan et al., nd). Fine-tuning is done by initializing the BERT model with the parameters learned from pre-training which are then fine-tuned while training the model on the downstream task (Devlin et al., 2018).

2.5.2. Robustly Optimized BERT Approach

Robustly optimized BERT approach (RoBERTa), proposed by Liu et al. (2019), relies heavily on the architecture of a BERT model, but some adjustments have been made. Firstly, the authors decided to use a larger text corpora to pre-train the model. While BERT is trained on approximately 16GB of uncompressed text, RoBERTa considers over 160GB of text data. Secondly, instead of masking the tokens once before pre-training, Liu et al. (2019) chose a dynamic masking which means that the masking pattern is updated each time, a new sequence is fed to the model. Thirdly, RoBERTa discards the NSP objective in the pre-training phase as Liu et al. (2019) found that it would slightly improve the performance of downstream tasks. The RoBERTa model is furthermore trained on a larger batch size than BERT (8000 vs. 256) as the perplexity score improves for the MLM objective as well as the accuracy of the downstream task. Lastly, RoBERTa is trained with a vocabulary that consists of 50000 tokens, it has therefore increased compared to the vocabulary used for BERT (Liu et al., 2019).

3. Neural topic modelling

Topic modelling is an important task in NLP because it can help humans to understand large document collections. This would be challenging otherwise as the amount of data has immensely increased in recent years. It is an unsupervised approach which has been successfully applied for text analysis for nearly twenty years. The goal of a topic model is to find a set of latent topics from a collection of documents. Each topic is supposed to be represented by an interpretable semantic concept (Zhao et al., 2021).

In the beginning, a lot of focus was on LDA, a Bayesian probabilistic topic model, which was explained in section 2.1. When neural networks were used for topic modelling, a new research area emerged, called neural topic models (NTMs; Zhao et al., 2021). NTMs can also be applied to NLP tasks such as summarization, text generation or translation which was not feasible for LDA.

The focus of this thesis is on NTMs with pre-trained language models such as BERT. Note, however, that also neural topic models exist that do not rely on pre-trained models, e.g. an autoregressive NTM (Larochelle and Lauly, 2012).

This Chapter will first describe Top2vec which outputs distributed representations of topics, given a document collection. This approach set the basis for BERTopic, an algorithm that performs density-based clustering on low-dimensional embeddings generated with BERT. It is used in this thesis to perform topic modelling of tweets written by German politicians. In the end, methods to evaluate the resulting topics will be explained.

3.1. Top2vec: Distributed representations of topics

Top2vec is a technique introduced by Angelov (2020) that leverages word and document embeddings, which are described in section 2.3, to create continuous vector representations of topics. While word2vec (Section 2.3.1; Mikolov et al., 2013a) produces distributed representations of words that are able to capture semantic and syntactic word relationships, doc2vec (Section 2.3.2; Le and Mikolov, 2014) extends word2vec by additionally generating distributed representations of documents.

According to Griffiths et al. (2007), "the association between words [and documents] depends on the distance between them in a semantic space". Therefore, semantically similar documents and words, respectively, should be close to each other in the embedding space. Since doc2vec calculates both, vector representations of words and documents, Angelov (2020) uses this approach to jointly learn word and document embeddings which are hence represented in the same semantic space. Angelov (2020) sees this space as a continuous representation of topics, where a topic is represented by a dense area of documents because they are close to each other and are therefore similar. The topic vectors can then be computed, for example, as an average of all document vectors that belong to the same dense area. Representative words for each topic are determined by finding those word vectors that are the closest to each topic vector in semantic space. Finally, top2vec assumes that the number of topics is given by the number of dense areas that the document vectors form (Angelov, 2020).

The resulting document vectors of doc2vec in the embedding space, that typically consists of around 300 dimensions, are very sparse. Hence it is difficult to find dense areas and it would need high computational cost. Therefore, Angelov (2020) uses the algorithm Uniform Manifold Approximation and Projection in order to reduce the dimensions of the distributed representations of documents. Dense areas can then be found in low-dimensional space with the density-based clustering algorithm Hierarchical Density-Based Spatial Clustering of Applications with Noise (Angelov, 2020). Both algorithms will be explained in more detail in sections 3.2.2 and 3.2.3, respectively.

Top2vec set the basis for other approaches that use dimension reduction and density-based clustering in order to find semantically similar documents and topics, such as BERTopic (Grootendorst, 2020a). Although BERTopic is similarly structured as top2vec, it does not use doc2vec to produce vector representations of documents, but it relies on BERT models. The following section will further describe BERTopic.

3.2. Clustering of embeddings by pre-trained language models with BERTopic

BERTopic which was introduced by Grootendorst (2020b) is a topic modelling technique that uses dimensionality reduction and density-based clustering to create topics of embedded texts. The algorithm can be described in roughly three steps which are shown in figure 3.1. First, the documents at hand need to be embedded which can be performed by using BERT, but also by any other embedding technique. Second, Uniform Manifold Approximation and Projection (UMAP) reduces the dimensionality of the resulting embeddings whilst keeping a significant part of the high-dimensional structure in lower dimension (McInnes et al., 2018). Additionally, these low-dimensional embeddings are being clustered using HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise). Each created cluster then consists of semantically similar documents. Third, representative words need to be extracted from each cluster to determine the topics, which is done via class-based TF-IDF (Term Frequency - Inverse Document Frequency). If the document embeddings are generated within BERTopic, the coherence of words within each topic will be improved. This can be achieved by using Maximal Marginal Relevance. In the following, each step will be explained more in-depth.

3.2.1. Embedding Documents

In this step, the documents are converted into numerical vectors in order to feed them to machine learning algorithms. There are many ways to achieve this, e.g. with the bag-of-words approach which does not rely on neural networks or Doc2Vec as explained in section 2.3. Furthermore, BERTopic is able to use transformer-based models which show state-of-the-art results. One huge advantage of using BERT for embedding documents is that there are many pre-trained models available which are ready to be used. In this work the sentence transformer models are used since the resulting representations usually work well for document-level embeddings. The defaults in BERTopic are set to two sentence transformers, depending on whether one selects a multilingual model. The default model for English texts only, ‘all-MiniLM-L6-v2‘, is an all-round model that is trained on a large and diverse dataset of over one billion training pairs. If the data consists of non-English texts, the parameter *language* needs to be changed to ‘multilingual‘. Then, the sentence transformer model ‘paraphrase-multilingual-MiniLM-

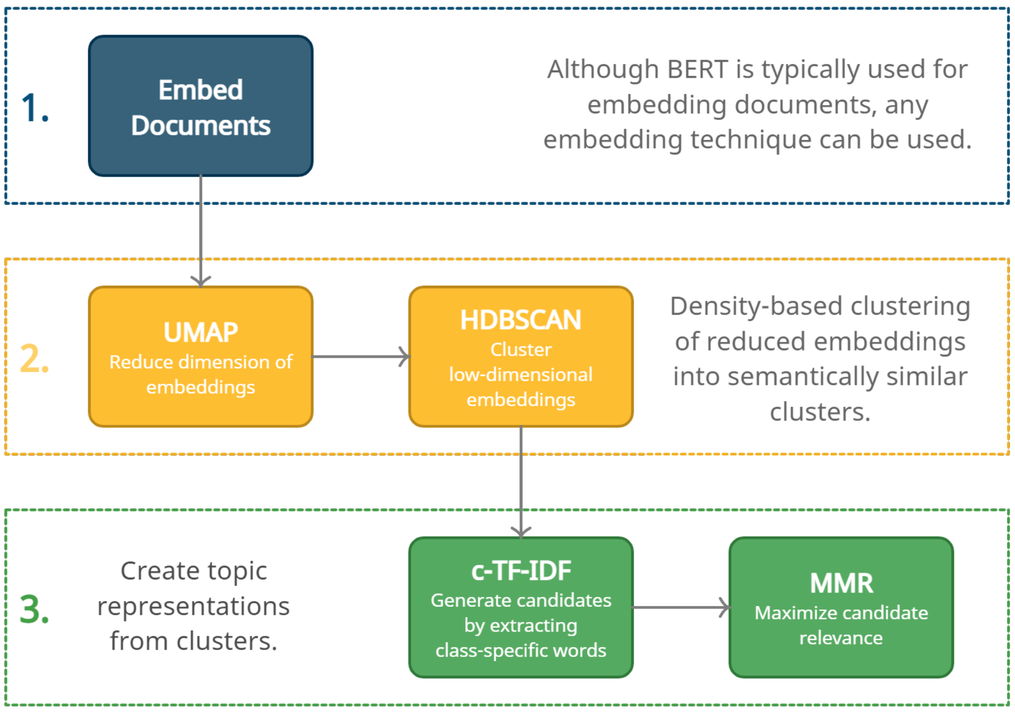

1.
Embed Documents
Although BERT is typically used for embedding documents, any embedding technique can be used.
2.
UMAP
Reduce dimension of embeddings
HDBSCAN
Cluster low-dimensional embeddings
Density-based clustering of reduced embeddings into semantically similar clusters.
3.
Create topic representations from clusters.
c-TF-IDF
Generate candidates by extracting class-specific words
MMR
Maximize candidate relevance

Figure 3.1.: The algorithm of BERTopic by Grootendorst (2020b)

L12-v2‘ will be used to embed the documents. It is similar to the English model, however it works for more than 50 languages and is therefore a bit larger.

3.2.2. Algorithm of UMAP

UMAP is a manifold learning technique by McInnes et al. (2018) that performs non-linear dimension reduction. The theoretical foundations are based on topological data analysis, manifold theory and Riemannian geometry (McInnes et al., 2018). UMAP can also be described as a weighted graph because it works in terms of fuzzy simplicial sets that are constructed of simplicial complexes which consist of 0- and 1-simplices (Hatcher, 2000). These fuzzy simplicial sets of local manifold approximations are being connected to build a topological representation of the data in high dimensional space. The low dimensional data is then supposed to have a fuzzy topological structure that is as similar as possible which can be achieved by minimizing the cross-entropy of these two representations. To construct a fuzzy topological representation one needs to approximate a manifold on which the data is assumed to be located. In the following, a little introduction to topology and simplicial complexes will be given. After that, the method for approximating the manifold will be explained. Then, it will be discussed how to find and optimize the corresponding low dimensional representations. To conclude this section, an overview of the algorithm of UMAP will be given.

Topological data analysis, simplicial complexes and simplicial sets

Topological spaces can be constructed out of simple combinatorial components with the help of simplicial complexes (Jänich, 2005, Chapter 7). These simplicial complexes are built by glueing together some building blocks which are called simplices. According to Dieck (2008), a simplicial complex $C = (E, S)$ contains a set E of vertices and a set S of finite non-empty subsets of E. A set $s \in S$ which has $q + 1$ elements is called a q-simplex of C, where q is also called the dimension of s. An example can be seen in figure 3.2 in which a 0-simplex is just a point or a vertex, a 1-simplex is a line or an edge between two 0-simplices, a 2-simplex is represented as a triangle and a 3-simplex as a tetrahedron. A simplicial complex then constructs topological spaces by glueing together these simplices. A simplicial complex has n dimensions if it consists of at least one n- simplex but no $(n + 1)$-simplices. In the case of UMAP, the simplicial complexes consist of only one dimension and are therefore a graph Dieck (2008).

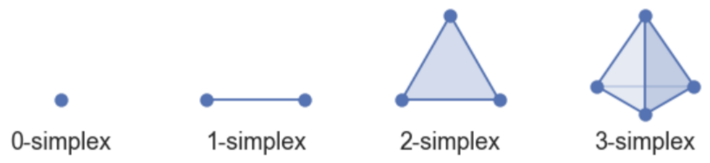


Figure 3.2.: Examples of simplices in low dimensions, figures are adopted from McInnes (2018).

Simplicial sets are more abstract, but they can create a broader class of topological spaces. May (1967) defined a simplicial set K as a graded set indexed on the non-negative integers together with maps $\partial_i : K_q \to K_{q-1}$ and $s_i : K_q \to K_{q+1}$, $0 \leq i \leq q$, which satisfy the following identities:

$$\begin{array}{rrcll}
(i) & \partial_i \, \partial_j & = & \partial_{j-1} \, \partial_i & if \quad i < j, \\
(ii) & s_i \, s_j & = & s_{j+1} \, s_i & if \quad i \leq j, \\
(iii) & \partial_i \, s_j & = & s_{j-1} \, \partial_i & if \quad i < j, \\
& \partial_j \, s_j & = & identity & = \partial_{j+1} \, s_j, \\
& \partial_i \, s_j & = & s_j \, \partial_{i-1} & if \quad i > j+1
\end{array}$$

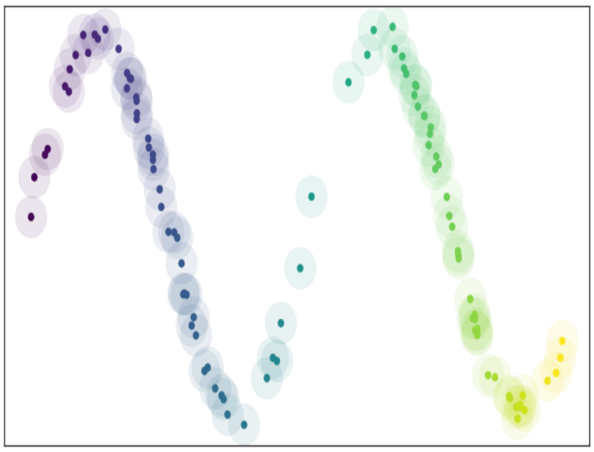
Here, the elements of K_q are called q-simplices, ∂_i and s_i are called face and degeneracy operators. A simplex x is degenerate if $x = s_i \, y$ for a simplex y and degeneracy operator s_i. If this is not true, then x is non-degenerate.

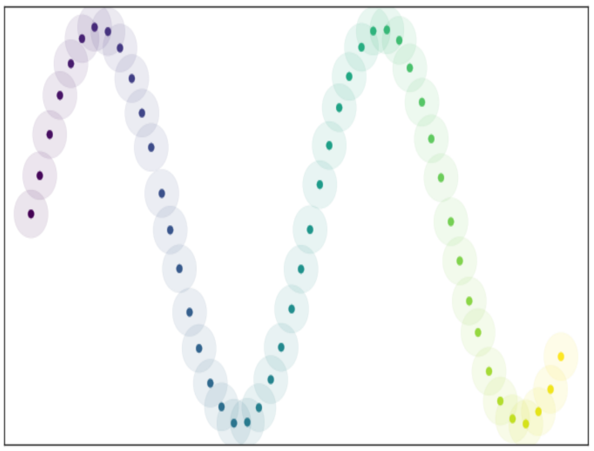
Uniform distribution of data on a manifold.

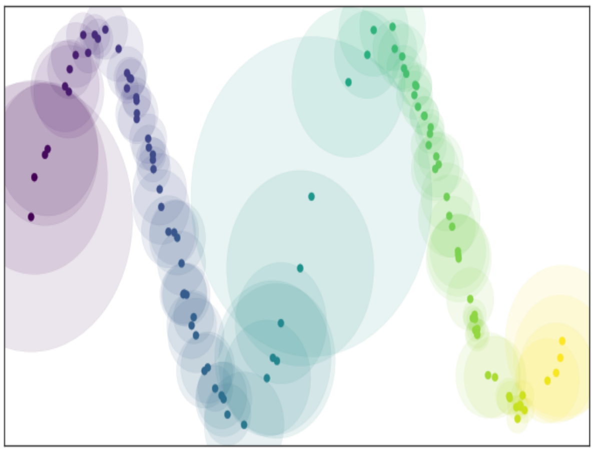
In this step, the manifold which the data is assumed to lie on will be approximated using geodesic distance. Belkin and Niyogi (2003) have stated in their work on Laplacian eigenmaps that the data is assumed to be uniformly distributed on the manifold. However, in practice, real-world data usually does not behave that way. Figure 3.3 demonstrates that assumption. The manifold is constructed in figure 3.3a by creating balls of some fixed radius around each data point. As the example data set only consists of finite samples, one cannot be sure that it truly is an open cover. In the example of non-uniformly distributed data, not all samples are covered by the approximated manifold. If the radius was chosen too large, the simplicial complex would turn into just a few high dimensional simplices and cannot capture the manifold structure anymore (McInnes, 2018). However, one can see in figure 3.3b that a suitable radius can easily be selected if the data is uniformly distributed to ensure that the cover actually connects the whole manifold.

Therefore, McInnes et al. (2018) assume that the data actually is uniformly distributed on the manifold and even if it seems as it is not, that must be due to the fact that the notion of distance is varying across the manifold. Moreover, the authors assume that the manifold has a Riemannian metric which is not inherited from the ambient space and one can then "find a metric such that the data is approximately uniformly distributed with regard to that metric" (McInnes et al., 2018).

Let the input data be $X = \{X_1, ..., X_n\}$, $\mathcal{M}$ the manifold the data lies on, and g be


(a) Distribution of example data with ball around each data point.


(b) Uniformly distributed data with ball around each data point.


(c) Open balls around data points with locally varying metric.

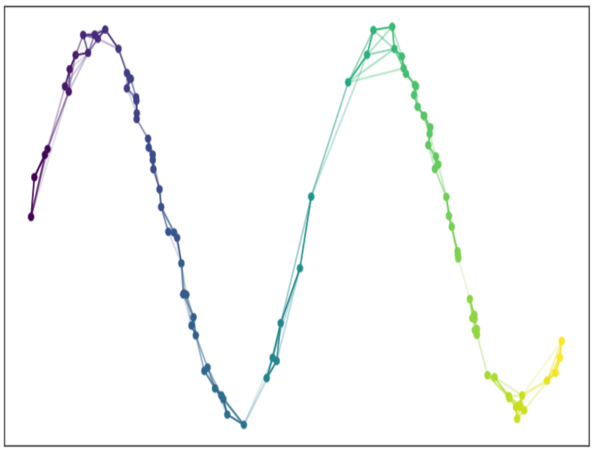

(d) Data as a graph. Data points are the vertices which are connected with weighted edges.

Figure 3.3.: Example of the approximated manifold of specific data by McInnes et al. (2018).

the Riemannian metric on $\mathcal{M}$. By assuming the data to be uniformly distributed on $\mathcal{M}$ with respect to g, any ball of fixed volume around the data points of X should consist of approximately the same number of data points, while not taking into account where on the manifold it is centred. This is being visualized in figure 3.3c.

Furthermore, a ball around X_i contains exactly the k-nearest neighbours of this data point and one can compute "geodesic distance from X_i to its neighbours by normalising distances with respect to the k^{th} nearest neighbour of X_i" (McInnes et al., 2018). To put this formally, let $d : X \times X \to \mathbb{R}_{\geq 0}$ be a dissimilarity measure of X. For each x_i, the set $\{x_{i_1}, ..., x_{i_k}\}$ of the k-nearest nearest neighbours under the metric d is being computed. Then, for each x_i, McInnes et al. (2018) defines ρ_i, the distance to the k^{th} nearest neighbour, and σ_i. Let

$$\rho_i \;=\; \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k,\; d(x_i, x_{i_j}) > 0\}.$$

This assures that every data point is connected to at least another x_i. Furthermore, set σ_i to be the value such that

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i}{\sigma_i}\right) \;=\; \log_2(k).$$

The local Riemannian metric for each data point is defined by the selection of σ_i as it corresponds to the normalisation factor. This results in a local metric space associated with each point of X which needs to be merged into a global structure. This can be achieved by using fuzzy simplicial sets which means that being a k-nearest neighbour to X_i is not binary anymore, but a weighted, fuzzy value between zero and one. As UMAP only deals with 0- and 1-simplices, this can also be represented as a weighted directed graph $\bar{G} = (V, E, w)$. The graph consists of vertices V which are the data points of X, edges between points E and the weight function w for the edges. The set of directed edges can be formed by $E = \{(x_i, x_{i_j}) \mid 1 \leq j \leq k,\; 1 \leq i \leq N\}$ and the weight function acording to McInnes et al. (2018) by

$$w((x_i, x_{i_j})) \;=\; \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right).$$

Figure 3.3d depicts this weighted graph for the example data set, where each weight is represented by the thickness of the connecting line (edge).

Finding and optimizing a low dimensional representation.

The goal in this step is to find a low dimensional representation $Y = \{Y_1, ..., Y_n\} \subseteq \mathbb{R}^d$ of the data X that has a topological structure that is as similar as possible to the one of X. The manifold for Y is given a priori and is commonly just $\mathbb{R}^d$, meaning the euclidean space. As the manifold and the metric are already known, the fuzzy topological representation can be computed directly. The distance to the nearest neighbour of any point X_i should also be taken into consideration. Therefore, UMAP has a hyperparameter *min_ dist* which determines the expected distance between nearest neighbours in the embedded space.

The algorithm of UMAP only considers 0- and 1-simplices as more would result in higher computational cost. Therefore, the fuzzy simplicial sets can also be described as a weighted graph as shown in figure 3.3d and one can use cross-entropy to compare the representations of X and Y. Let A denote the fuzzy set of all possible 1-simplices, $w_h(a)$ and $w_l(a)$ are the weights of the 1-simplex a in the high and low dimensional case, then the cross-entropy C will be

$$C((A, w_h), (A, w_l)) \mathrel{\widehat{=}} \sum_{a \in A} \left(w_h(a)\ log \left(\frac{w_h(a)}{w_l(a)} \right) + (1 - w_h(a))\ log \left(\frac{1 - w_h(a)}{1 - w_l(a)} \right) \right).$$

The embedding of Y can be optimized with respect to C by using stochastic gradient descent. The optimization problem at hand is to minimize the error between the high and low dimensional topological representations (McInnes et al., 2018; McInnes, 2018).

The UMAP algorithm and its hyperparameters

To conclude, the algorithm of UMAP consists primarily of two steps: the data at hand first needs to be represented on a manifold which can be achieved by using fuzzy simplicial sets to construct a weighted k-neighbour graph. In the second step, this graph needs to be embedded in lower dimensions while preserving the topological structure.

UMAP has some important hyperparameters that have an impact on the resulting low-dimensional embedding. One hyperparameter is *n_ neighbours* which controls the number of neighbours of a point to be considered when approximating the manifold. Small values for *n_ neighbours* will result in a fine-grained and detailed manifold structure, but by potentially losing the ‘big picture‘. Higher values, on the other hand, capture manifold structure on a larger scale but may lose the detailed structure. The default value for *n_ neighbours* is 15. Another hyperparameter is *min_ dist*. It controls the construction of fuzzy simplicial sets for the low-dimensional embedding by determining how close points are allowed to be to each other. Low values for *min_ dist* will represent more structure of the approximated manifold as higher values because they allow densely packed regions. If *min_ dist* is chosen to be higher, the points in the embedding are forced to spread out which does not capture the structure of the manifold as faithfully as lower values do. The default value for *min_ dist* is 0.1. If one wants to determine the number of dimensions, the data should be reduced to, one can define the hyperparameter *n_ components* whose default is 2. Another hyperparameter is *metric*. It defines how the distance in the ambient space of the input data is computed. The default is the euclidean metric (McInnes et al., 2018). Although there are many more parameters of UMAP, these are the most important ones.

3.2.3. Algorithm of HDBSCAN

Campello et al. (2013) introduced a clustering algorithm that extends DBSCAN by incorporating hierarchy. Unlike DBSCAN, hierarchical density-based spatial clustering of applications with noise is able to find clusters with varying densities and is also more robust to the selection of parameters. The algorithm works in several steps. First, the space is being transformed to better distinguish possible outliers and noise from other data points. Second, a minimum spanning tree is built from that to then convert it into the cluster hierarchy which can be described as a dendrogram. In the next step, HDBSCAN needs to simplify that dendrogram by condensing the cluster tree. The last step is to extract the clusters which can be achieved by maximizing the overall ‘stability‘ of the set of clusters.

Transforming the space

Before the clustering is done, the algorithm transforms the space, the input data $X = \{x_1, ..., x_n\}$ lie in, in order to find the data points x_i that are noise. HDBSCAN uses single linkage clustering

$$D_{SL}(C_r, C_s) \;=\; \min\{d(a_i, a_l) \mid a_i \in C_r,\ a_j \in C_s\}$$

which denotes the distance d between two clusters C_r and C_s as the minimum distance between the members of the two clusters. However, it can be sensitive to noise. Therefore, Campello et al. (2013) use a different metric to measure the distance of data points which locates objects of noise further away from the other ones. The core distance of a data point $x_p \in X$ is the distance from x_p to its k^{th}-nearest neighbour and is denoted as $d_{core}(x_p)$ with k being an input parameter of HDBSCAN. Points in areas that have a low density, therefore have probably a high core distance. The new metric is the mutual reachability distance that is defined for two objects x_p and x_q with respect to k as

$$d_{mreach}(x_p, x_q) \; = \; \max\{d_{core}(x_p), d_{core}(x_q), d(x_p, x_q)\}$$

where $d(x_p, x_q)$ is the original distance metric between these two objects. Furthermore, Campello et al. (2013) define that a data point $x_p \in X$ is called an ϵ-core object for every $d_{core}(x_p) \leq \epsilon$. The dissimilarity matrix of size $n \times n$ consists now of the mutual reachability distance between each point. The data X can now also be seen as a graph, the mutual reachability graph, with the objects as vertices and the mutual reachability distance between the respective pair of data points as the weight of each edge.

Consider the threshold value ϵ which starts high and gets steadily lower. The general idea is to drop any edges that have a weight above ϵ and treat the resulting single points $d_{core}(x_p) > \epsilon$ as noise. This can be done in a hierarchical way by building a Minimum Spanning Tree of the mutual reachability graph from which a dendrogram can be derived. Then, the edges can be dropped in decreasing order of the weights (Campello et al., 2013; McInnes et al., 2016).

Hierarchy Simplification and Cluster Extraction

The HDBSCAN hierarchy of the data X can also be visualized in a dendrogram. Campello et al. (2013) state that these plots are difficult to interpret and process for large and noisy data. Therefore, they want to extract a summarized tree of only considerable clusters from the dendrogram. For that reason, the parameter minimum cluster size min_{clSize} is set. At each split of the hierarchy, the newly created cluster should contain at least min_{clSize} objects. If it does not, these points fall out of the cluster and become noise. If a cluster got split in two clusters that have at least min_{clSize} data points, then this is considered a true split which will be kept in the tree. After looking at each node, the tree will have become smaller and easier to interpret as it lost some data points to noise.

To find the optimal global solution of cluster extraction, Campello et al. (2013) describe the problem of maximizing the overall stability. To describe that, let define the density threshold $\lambda = \frac{1}{\epsilon} \in [0, \infty)$. It is intuitive that if λ increases, ϵ, the threshold that determines which edges of the tree are dropped, decreases and thus the clusters will get smaller. The following concept is adapted from the notion of *excess of mass* by Müller and Sawitzki (1991). If the density level λ is being increased, then a cluster C_i appears at the level $\lambda_{min}(C_i)$. To compare the stabilities of nested clusters which may appear in the tree, Campello et al. (2013) introduced the *Relative Excess of Mass* of a cluster that arises at level $\lambda_{min}(C_i)$ as

$$E_R(C_i) \;=\; \int_{x \in C_i} (\lambda_{max}(x, C_i) \;-\; \lambda_{min}(C_i))\; dx$$

where $\lambda_{max}(C_i)$ denotes the density level at which C_i is spilt or vanishes. As the HDB-SCAN hierarchy has finite data X and a density threshold that is associated with each hierarchical level, the stability of a cluster C_i can be adapted to

$$S(C_i) = \sum_{x_j \in C_i} (\lambda_{max}(x_j, C_i) \;-\; \lambda_{min}(C_i)) \;=\; \sum_{x_j \in C_i} \left(\frac{1}{\epsilon_{min}(x_j, C_i)} - \frac{1}{\epsilon_{max}(C_i)} \right).$$

Here, $\lambda_{min}(C_i)$ is the minimum level at which this cluster exists, $\lambda_{max}(x_j, C_i)$ is the threshold level beyond which the data point x_j does not belong to cluster C_i anymore and $\epsilon_{min}(x_j, C_i)$ and $\epsilon_{max}(C_i)$ are the related values for the threshold ϵ.

To find the best set of non-overlapping clusters, one needs to start at the leaf nodes and declare them as selected clusters. Then, the algorithm decides at each node C_i bottom-up, while ignoring the root node, whether the cluster selection up-to-then of C_i's subtrees or C_i itself should be selected as a cluster. This decision is made at each node C_i by updating the total stability $\hat{S}(C_i)$ of the clusters that are selected in the subtree rooted at C_i. $\hat{S}(C_i)$ is then defined as

$$\hat{S}(C_i) = \begin{cases} S(C_i), & \text{if } C_i \text{ is a leaf node} \\ \max\{S(C_i),\, \hat{S}(C_{i_l}) + \hat{S}(C_{i_r})\} & \text{if } C_i \text{ is an internal node} \end{cases} \tag{3.1}$$

with $\hat{S}(C_{i_l})$ and $\hat{S}(C_{i_r})$ being the left and right children of C_i. In the end, HDBSCAN returns a flat, non-overlapping clustering of the input data (Campello et al., 2013).

3.2.4. Idea of class-based TF-IDF

Once the clusters are created by using HDBSCAN, the next step is to find representative words for each cluster. Term Frequency - Inverse Document Frequency (TF-IDF) compares the importance of words between documents and penalizes words that are frequent across all documents by rescaling the frequency of words. Term Frequency (TF) is a measure of how many times a word w appears in a document d:

$$tf_{w,d} = \frac{n_{w,d}}{\text{Number of words in the document}}$$

with n being the number of times a word w occurs in a document d. Inverse Document Frequency (IDF) measures the importance of a word w:

$$idf_w = \log\left(\frac{\text{Number of texts}}{\text{Number of texts with word } w}\right).$$

The TF-IDF score can then be computed for each word in the vocabulary. The higher the score, the more important is hence the word (Huilgol, 2020):

$$tfidf_{w,d} = tf_{w,d} \cdot idf_w$$

Grootendorst (2020b) now treats all documents that belong to the same topic as a single document and then applies TF-IDF. This would return those words per topic that are most important and unique within each cluster and disregard those that are frequent across all topics. The author called this method class-based TF-IDF, abbreviated c-TF-IDF.

3.2.5. Maximal Marginal Relevance

By using c-TF-IDF, representative words for each topic are found. However, these words do not necessarily describe a coherent topic. Therefore, Maximal Marginal Relevance (Carbonell and Goldstein, 1998) can be applied to improve the coherence of words by removing those words that do not contribute to a topic. Furthermore, it reduces the number of synonyms that occur as topic representatives, hence it diversifies the words that describe a topic (Grootendorst, 2020b).

MMR was introduced by Carbonell and Goldstein (1998) and is "a method for combining query-relevance with information-novelty in the context of text retrieval and summarization". Given a user query, an intuitive notion is to find those documents that are most similar to this query. However, if there exist many potentially relevant documents that are redundant to each other, it is favourable to find those documents that consider the query-relevance on the one hand but also find documents with new information on the other hand. The Maximal Marginal Relevance criterion is therefore supposed to re-rank documents in order to reduce their redundancy while also considering their relevance given the user query. A high marginal relevance means that it maintains the relevance of the query while having a minimal similarity to documents that have already been selected. Therefore, the goal is to maximize marginal relevance and MMR is then defined as

$$MMR = \underset{D_i \in R \setminus S}{\arg\max} \left[\lambda \, \mathrm{Sim}_1(D_i, Q) - (1 - \lambda) \max_{D_j \in S} \mathrm{Sim}_2(D_i, D_j) \right].$$

Carbonell and Goldstein (1998) describe the parameters used in this definition as follows:

> “[...] C is a document collection (or document stream); Q is a query or user profile; $R = IR(C, Q, \theta)$, i.e., the ranked list of documents $[D]$ retrieved by an IR system, given C and Q and a relevance threshold θ, below which it will not retrieve documents (θ can be degree of match or number of documents); S is the subset of documents in R already selected; $R \setminus S$ is the set difference, i.e, the set of as yet unselected documents in R; Sim_1 is the similarity metric used in document retrieval and relevance ranking between documents (passages) and a query; and Sim_2 can be the same as Sim_1 or a different metric.“

Furthermore, λ is the diversity parameter that is tunable by the user. A higher λ gives higher accuracy, whereas a lower λ delivers a higher diversity.

3.2.6. Selection of parameters in BERTopic

One can set many parameters when using BERTopic. In the following, some important parameters will be named with their value in brackets that we use for topic modelling. One can either choose English or multilingual as language `(language = ‘multilingual‘)` for the model, as well as the number of words that should be extracted per topic `(top_n_words = 10)`. Furthermore, the diversity of the topic representatives can be chosen with a value between 0 and 1 `(diversity = 0.4)`.

Additionally, one can use custom models for the 3 steps: embedding documents, reducing dimension with UMAP and clustering low-dimensional embeddings with HDBSCAN. To embed the documents, one can use Sentence Transformers as it is the default, but any other embedding technique as well. The UMAP model that is used for dimension reduction considers the k-nearest neighbour points of each data point `(n_neighbours = 15)`, reduces the embeddings to certain dimension `(n_components = 10)` and the distance in the ambient space between objects is computed by using the cosine metric `(metric = ‘cosine‘)`. Given two vectors a and b, the cosine similarity can be computed as

$$S_C(a,b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}}$$

Moreover, one can determine how close data points are allowed to be to each other `(min_dist = 0.0)`.

The model for clustering with HDBSCAN returns clusters that consist of at least 15 documents `(min_cluster_size = 15)`. Furthermore, a parameter can be set that affects the amount of documents that are declared as noise `(min_samples = 1)`. The higher it is, the more points will be labelled as noise (McInnes et al., 2016). In order to get the most persistent clusters the Excess of Mass algorithm `(cluster_selection_method = ‘eom‘)` is used. Lastly, the distance between instances is calculated by using the Euclidean metric `(metric = ‘euclidean‘)`.

4. Application to Twitter data

This chapter describes the application of neural topic models, specifically BERTopic (Section 3.2), to Twitter data from German politicians. Twitter is a social media platform where users can write and interact with messages which are also called ‘tweets‘. The interactions comprise, e.g. to post, to like or retweet tweets.

According to Schmidt (2017), the use of social media by political parties and individual politicians has long been the focus of research. There is a recognisable focus on the use of these instruments, such as Twitter or Facebook, in election campaigns as these are phases in which democratic societies particularly debate different political goals and proposals. In Germany, social media played an important role for the first time in 2009, although Twitter, specifically, was used only by a few candidates. In the federal election campaign in 2013 and 2017, the use of Twitter then continued to rise (Schmidt, 2017).

Since political goals are more and more discussed in social media, it is interesting to understand the communication behaviour of candidates as well as the topics that are debated and their sentiment towards it. In order to determine sentiments of discussed topics, the topics first need to be generated. The focus of this thesis is to extract topics from Twitter data written by German politicians. This chapter first describes the data, specifically the tweets, and then moves on to give a short overview of the political system in Germany. Lastly, different results of topic modelling with BERTopic are presented and compared.

4.1. Data

In order to apply neural topic models to Twitter data, the tweets need to be scraped first. As a scholar at LMU (Weingärtner, 2021) performed a sentiment analysis with the same tweets, he kindly provided the data. The tweets were pulled from the Twitter API by using the `academictwitteR` package (Barrie and Ho, 2021), for which the usernames are needed. For that purpose, a data set by Stier et al. (2018) is used which contains detailed information of members of the German government. Specifically, it contains information on 2516 politicians, e.g. about their respective surname, first name, age, sex, party affiliation, place of residence, place of birth, profession and Twitter-URL. The latter determines whether a candidate has a Twitter account. Since 1294 candidates do not have one, the data set downscales to 1222 politicians whose tweets were scraped.

Tweets were pulled in the period of 25th March 2017 until 24th September 2017. This, therefore, covers six months leading up to the German federal election 2017 as election day was on September 24th. Out of 1222 candidates, only 811 politicians were active on Twitter during that time and 269000 tweets in total were pulled from the Twitter API along with their respective metadata. The metadata contains information, e.g. on how often a tweet was retweeted or the exact date and time it has been posted. Furthermore, we know whether a tweet is an original tweet, a retweet, a reply to a different text or a quote. The data set consists of around 120000 retweets, 93000 original tweets, 39000 replies and approximately 19000 quotes. Although each tweet can be associated with its author, the most important subject for topic modelling are the tweets as the approach of BERTopic (Section 3.2) does not need any metadata.

Figure 4.1 shows the number of tweets that were posted each day from 25th March until 24th September 2017. During this time period, approximately 1467 tweets were posted on average each day. It starts in April at around 1000 tweets per day and increases until the end of September to approximately 2000 tweets a day, although it varies throughout the months. A peak can be seen at the beginning of September. The famous duel for chancellorship in Germany was aired on September 3rd and candidates of all parties referred to that on Twitter which explains the peak.

When writing a tweet, users can use smileys or hashtags to underline their statement. Furthermore, one can tag other people in their message when referring to someone, for example. These are some of the reasons why the tweets need to be properly prepared before applying topic models. The process of data cleaning will be described in the

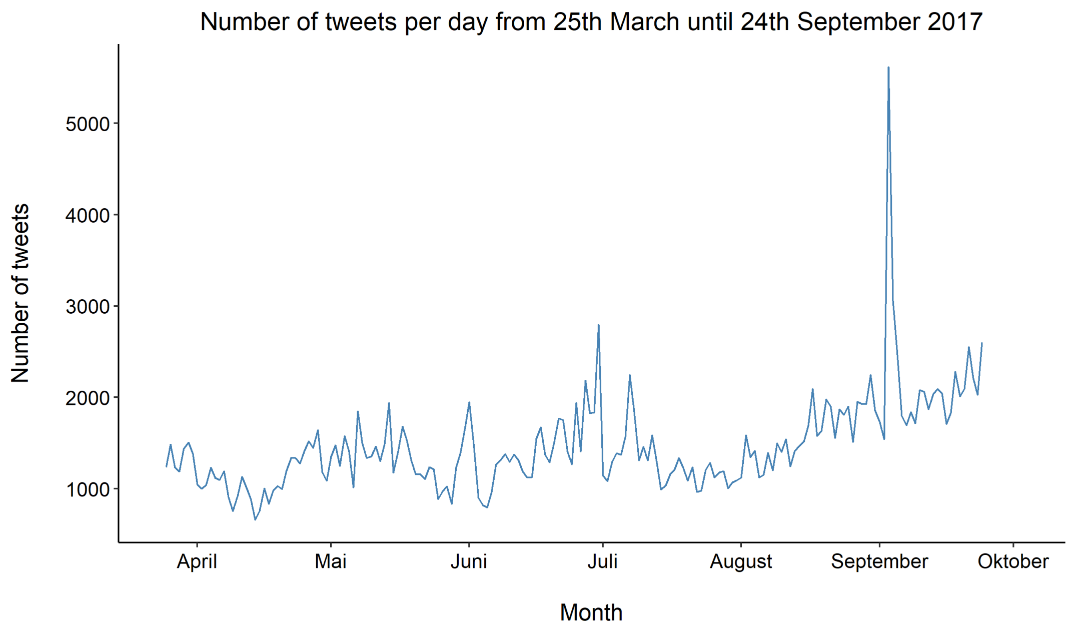


Figure 4.1.: Line plot that shows the number of tweets per day in the period of 25th March until 24th September 2017.

following in more detail.

Data preprocessing

As results of any machine learning algorithm rely on its input, text data, which can be messy and might contain unwanted texts such as URLs or smileys, needs to be preprocessed. Some parts of preprocessing are adopted from Weingärtner (2021) which are specified in the Appendix. Since he did his analysis in the statistical software R, preprocessing for this thesis was also done in R, while Python was then used for topic modelling. We decided to use Python for topic modelling as it provides an extensive collection of NLP tools and libraries.

What got conspicuous after taking a closer look at the tweets at hand, was that some tweets were cut off at the end. Table 4.1 shows the number of times a tweet was cut off in total, in a URL, Tag, Hashtag or in a word.

81145 tweets were cut off in total, this concerns therefore around 30% of the data. Most of the tweets that are cut off are retweets, specifically 81113 cut retweets. If a person retweets a tweet of a different user, the content does not change, but it gets marked as a retweet by adding ‘RT @...‘ in the front. Examples of cut tweets and the original tweets can be seen in table 4.2.

Cut tweets	**Number of times**
In total	81145
In a word	33618
In URL	29761
In Hashtag	13247
In Tag	4519

Table 4.1.: Number of times a tweet is cut off.

Original Tweet	**Scraped Tweet**
In 7 Tagen können wir uns aufregen, dass rechte Hetzer im Bundestag sitzen. Oder wir können diese 7 Tage nutzen, das zu verhindern.	RT @MartinSchulz: In 7 Tagen können wir uns aufregen, dass rechte Hetzer im Bundestag sitzen. Oder wir können diese 7 Tage nutzen, das zu v...
Ein Bericht aus den Schulen in meinem Kiez. Man stelle sich vor,Pfarrer würden sich verhalten wie Neuköllns Imame... https://google.de/amp/amp.berlin	RT @RobertRossmann: Ein Bericht aus den Schulen in meinem Kiez. Man stelle sich vor,Pfarrer würden sich verhalten wie Neuköllns Imame... ht…
Wenn Wahrnehmung und Fakten deutlich auseinanderfallen, besteht das Risiko, dass Stimmungen das Bild prägen. @jensspahn #DialogBMF	RT @BMF_Bund: Wenn Wahrnehmung und Fakten deutlich auseinanderfallen, besteht das Risiko, dass Stimmungen das Bild prägen. @jensspahn #Dial…

Table 4.2.: Examples of cut tweets.

Since each tweet can be identified by its unique id, the cut tweets can be replaced with the original tweet if the author of the original tweet is one of the 811 politicians. Otherwise, the tweet can not be changed. We were able to replace 15511 cut tweets with their respective original tweet. Hence 65634 tweets (24.3%) are still cut off at the end.

The next step of preprocessing was to replace German umlauts and ligature s‘. After that, we removed all URLs, the ‘RT‘ at the beginning which classifies a text as a retweet, the cut-off word at the end of a tweet, smileys, extra whitespaces, line breaks, symbols and special characters. Furthermore, some hashtags can be split back into their original words if they were written in camel case. Lastly, the hashtag symbol ‘#‘ and Tags ‘@...‘ were removed from the tweets and all words were converted into lower case.

Data set	**# Duplicates**	**# Empty**	**# Not German**	**Resulting # Tweets**
Little preprocessing with duplicates	51221	8121	14089	**247690**
Little preprocessing without duplicates	51221	8121	11805	**198753**
Strong preprocessing with duplicates	52468	8395	13728	**247777**
Strong preprocessing without duplicates	52468	8951	11243	**197237**

Table 4.3.: Number of duplicate, empty, non-German and resulting tweets for the four data sets.

If one uses a topic model with a bag-of-words assumption such as LDA, some more preprocessing steps are usually done, e.g. removing stopwords, digits or punctuation. However, BERT is a pre-trained language model that is supposed to understand the context which indicates to keep as much context as possible in tweets. Nevertheless, tweets are very short because Twitter only allowed texts that are no longer than 140 characters in that period [1]. Since URLs or Tags have been removed, a tweet might have already lost important context. Therefore, we decided to use two differently preprocessed data sets: little and strong preprocessed tweets. Little preprocessing means the data preparation that was described in the last paragraph and strong preprocessing additionally removes all digits, punctuation and stopwords.

The resulting two data sets with little and strong preprocessing, respectively, are further used to create two more data sets: one that consists of only unique tweets and one that contains some duplicate tweets. If someone retweets a text of a person whose tweets were pulled from Twitter as well, the tweet exists at least two times in the data set. To examine whether the results differ between data sets with and without duplicated tweets, the topic models will be applied to four data sets: little preprocessing with duplicates denoted as D_l, little preprocessing without duplicates $D_{l,u}$, strong preprocessing with duplicates D_s and strong preprocessing without duplicates $D_{s,u}$. As some candidates only tweeted URLs, some tweets are empty character strings after being preprocessed and were also removed.

[1] Twitter increased the maximum length to 280 characters in November 2017.

Although the tweets are written by German politicians, the language does not necessarily be German. Therefore, we used the package `fasttext` proposed by Joulin et al. (2016b) and Joulin et al. (2016a) to identify the language of each tweet. Finally, we removed the tweets from all four data sets that were not written in German. Table 4.3 particularly shows the number of resulting tweets per dataset after preprocessing. The number of words per tweet is depicted in figure 4.2 for each data set.

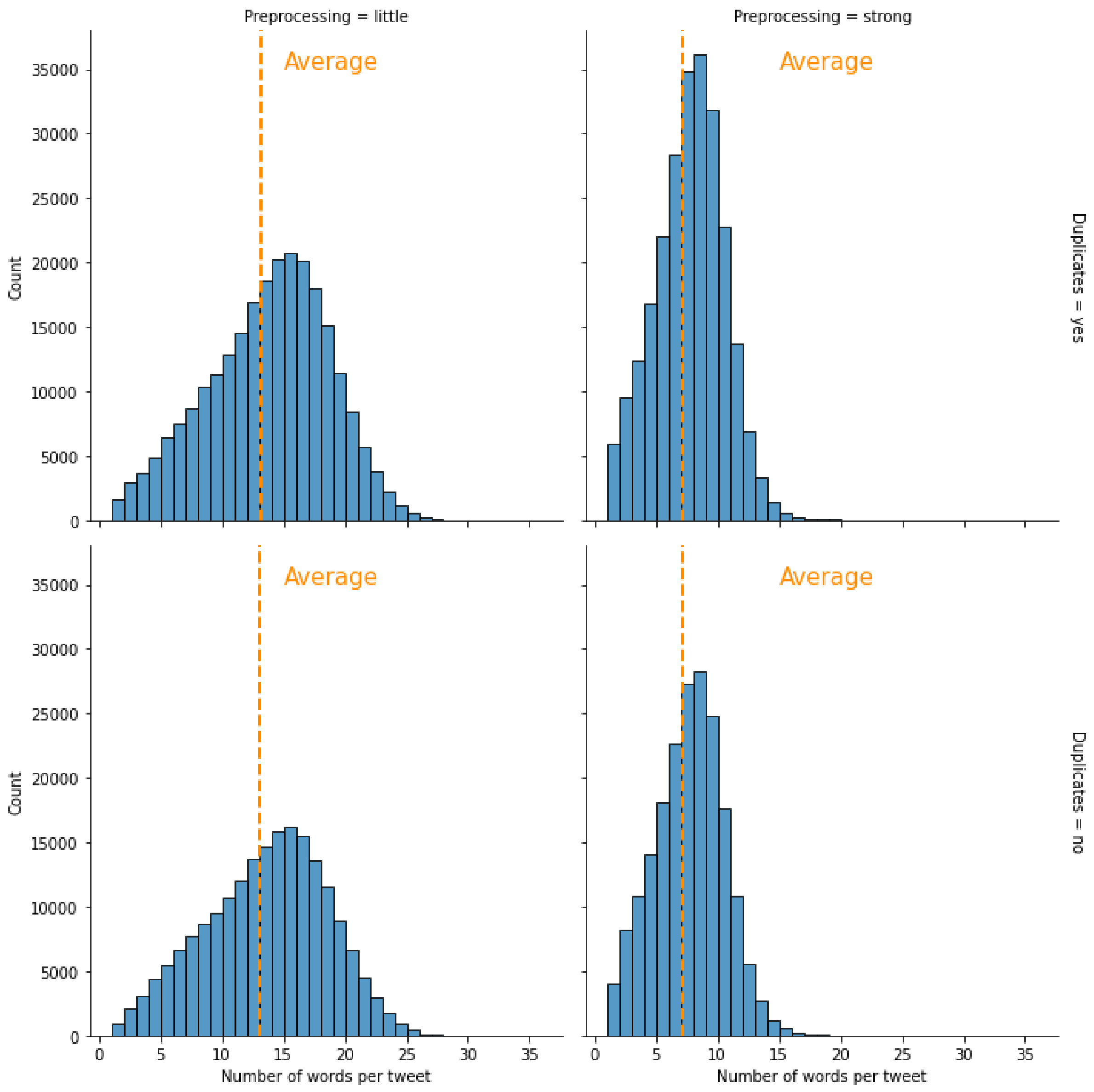


Figure 4.2.: Histogram of the number of words per tweet per data set.

Both data sets with little preprocessing consist of tweets that comprise up to 28 words and the average amount of words are 13 which is represented as the orange, dashed line. The mode of both data sets is 15 words. While the distribution of the number of words is similar for the data sets with little preprocessing, the difference is the count of tweets. The data set with duplicates contains about 20000 tweets that consist of 15 words. The dataset without duplicates in contrast comprises around 16000 tweets of 15 words each. This discrepancy is due to removing 51221 duplicate tweets in this data set. The same pattern can be observed for both data sets with strong preprocessing, although the average tweet contains 7 words, the mode is 8 words and the maximum amount of words in a tweet is 18 and 19 words, respectively.

This concludes the preprocessing. In summary, topic models will be applied to four different data sets, with little and strong preprocessing as well as with and without duplicate tweets, respectively. It will be examined whether this leads to different results. The next part will give an overview of the political system in Germany.

4.2. Political system in Germany

This section is supposed to give a general overview of the political system in Germany and its multi-party system. The Basic Law of the Federal Republic of Germany was signed in 1949 and came into force the same year. Accordingly, the principles of democracy and republic, the welfare state, the federal state and constitutional state apply in Germany (Marschall, 2018, p. 31). The principle of democracy comprises amongst other things multi-party system, equal opportunities for every party and sovereignty of the people (Marschall, 2018). Furthermore, dictatorship is excluded. In regular general elections, the people themselves determine who should govern them. They can choose between competing parties. Whoever receives the majority of the electoral votes then governs, but only for a certain period of time. Once a party is in power, it must be able to be voted out again. German Federal elections take place every four years (Thurich, 2011).

Since Germany has a multi-party system, the parties and their respective manifestos play a central role when it comes to elections. The tweets that are considered for topic modelling are written by candidates running for one of the major parties *CDU, CSU, SPD, FDP, Bündnis90/Die Grünen, Die Linke* and *AfD*. Figure 4.3 depicts the number of tweets that each party posted. The order of the parties in this plot reflects the election result in the federal election 2017, where *CDU* got most votes (26.8%) and *CSU* received the least (6.2%; Der Bundeswahlleiter, 2017).

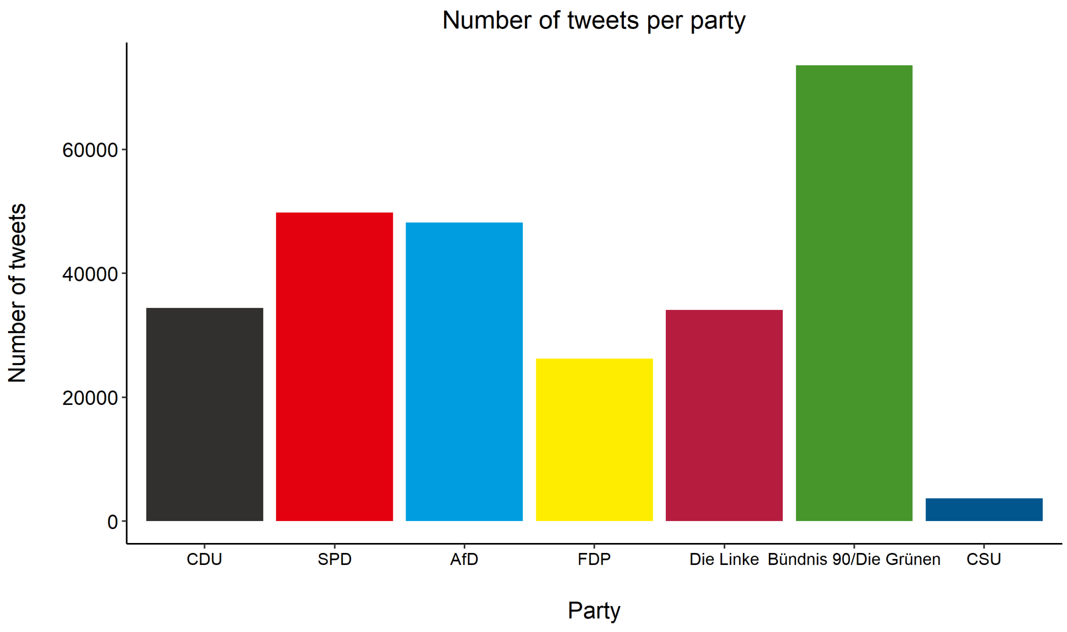


Figure 4.3.: Barplot that shows the number of tweets per party.

The Green party *Bündnis90/Die Grünen* were most active of all as they posted more than 73000 tweets in the period of 25th March 2017 until 24th September 2017, which comprises around 26% of all tweets. The social party *SPD* released around 50000 tweets, closely followed by the *AfD* with approximately 48000 tweets. The conservative *CDU* and *Die Linke* both tweeted roughly 34000 messages, the liberal *FDP* about 26000, while the Bavarian sister party of the *CDU*, the *CSU*, wrote around 3700 tweets.

4.3. Evaluation

This section evaluates three different results of topic models[2]. All of them were generated with BERTopic proposed by Grootendorst (2020b). The algorithm of BERTopic first transforms the tweets into numerical vector representations with some embedding model that can be chosen by the user. Then the dimensions of these vector representations are reduced by using UMAP in order to cluster the low-dimensional vectors with HDBSCAN

[2]All codes can be found here: `https://github.com/annegriddl/Neural-topic-modeling`

which results in the topics. The representative words for each topic can be found by using class-based TF-IDF (see section 3.2). A method of BERTopic in Python is to update the representative words of a topic model. This will be done for the resulting topics of the data sets with little preprocessing to prevent stopwords to become representatives. Furthermore, we decided to use three different embedding models: a multilingual Sentence-BERT (SBERT), German BERT and GottBERT. The latter is a RoBERTa model (Section 2.5.2) and the other two are both based on BERT (Section 2.5.1). In the following sections, the resulting topics of the three different embedding models, SBERT, German BERT and GottBERT, will be first described and then compared.

4.3.1. Sentence-Transformers

SBERT was proposed by Reimers and Gurevych (2019) and is based on the pre-trained BERT network. It is able to derive fixed-length vectors for input sentences where semantically similar sentences are close to each other in vector space and can be found by using a similarity measure such as Euclidean distance or cosine similarity (Reimers and Gurevych, 2019). While BERT needs to compare every combination to find the most similar pair of sentences in a collection, SBERT is more efficient. Furthermore, it maintains the accuracy from BERT and shows state-of-the-art results on tasks such as Semantic Textual Similarity (Reimers and Gurevych, 2019). Since SBERT was only for English texts until then, Reimers and Gurevych (2020) introduced the multilingual Sentence-BERT which maps translated sentences and their respective original text to the same location in vector space. The multilingual SBERT can be applied to more than 50 different languages, including German. SBERT was trained on two different data sets: SNLI (Bowman et al., 2015) which comprises 570000 sentence pairs and the MultiNLI data set (Williams et al., 2018) with 430000 sentence pairs (Reimers and Gurevych, 2019). The specific multilingual SBERT model that is used in this thesis is called ‘paraphrase-multilingual-mpnet-base-v2‘ (Reimers, 2022).

Since topic modelling is an approach of unsupervised learning and the real topics are unknown beforehand, it is difficult to find the ‘perfect‘ number of topics. In this thesis, we fitted BERTopic models with SBERT as embedding model for different numbers of topics $k = \{10, 20, 30, 40, 50\}$. Although there exist more than 1000 topics for each data set at the beginning, BERTopic offers a method that reduces the number of topics to a certain amount. This can be achieved by “iteratively merging the least frequent topic with the most similar one based on their c-TF-IDF matrices“ (Grootendorst, 2020a). The topics, their sizes, as well as their representations are then updated. The lower the number of topics k, the more tweets are assigned to noise by HDBSCAN. For the sake of reproducibility, a seed is set to 1997. However, the results do vary, especially the size of the topics. When applying the model several times on the same data set, the resulting topics are very similar, but the order and therefore their respective size differs. Therefore, we advise saving the topic model after it had been fit as the results can not be reproduced otherwise.

While the representative words of topics for $k = \{10, 20, 30, 40, 50\}$ always seem to be semantically meaningful and coherent for each topic, the models for $k = 50$ topics contain most information. Although the topics may be smaller with $k = 50$, the topics are more specific than the ones of models with 20 topics. Furthermore, most of the resulting topics of the four different data sets are the same when using 50 topics, only the size of the topics differs. When the number of topics decreases, the topics of the four data sets vary more because the sizes of the respective topics get more relevant. Moreover, the smaller the number of topics, the more tweets are labelled as noise, probably because the tweets of the omitted topics can not be re-allocated to different, existing clusters.

Nevertheless, many topics for $k = 20$ for the four data sets, little and strong preprocessing as well as with and without duplicates, respectively, are similar when ignoring their size. The labels of the topics are shown in table 4.4 for $k = 20$ and each data set. Note that these topics are labelled subjectively by consulting the top ten representative words for every topic and data set. Furthermore, the table does not reflect the size of each topic as the order was changed to display similar topics next to each other. Topics that occur in every model are *Climate change*, the *Chancellorship duel*, *Taxes* and *Twitter*. Other topics that can frequently be found are, e.g. *Police, Digitization, Freedom of press with reference to Turkey* or *Education.* Topics that appear in only one model are, for example, *Sea rescue of refugees, Extremism, Pension* or *German Federal Armed Forces.*

Looking at table 4.4 gives the impression that the tweets have been labelled successfully. However, every topic model of these four declared more than 80% of the tweets as noise, since the algorithm of HDBSCAN is used to cluster the low-dimensional embeddings. HDBSCAN is able to label those tweets as noise that are considered as outliers (see section 3.2.3). The fitted model for data set D_l assigned only around 10% of the tweets to real topics and the model for the data set $D_{l,u}$ approximately 15%. Moreover, about 14% of the duplicate tweets that were strongly preprocessed were allocated to topics and approximately 16% of the tweets in data set $D_{s,u}$. Since the fitted topic models produce similar outputs, we will now focus the evaluation on the model for unique tweets and strong preprocessing because it assigns the greatest percentage of tweets to topics. Furthermore, the model with $k = 20$ topics will be evaluated for better clarity. The representative words of the model fitted with 50 topics with the data set $D_{s,u}$ are shown in the Appendix.

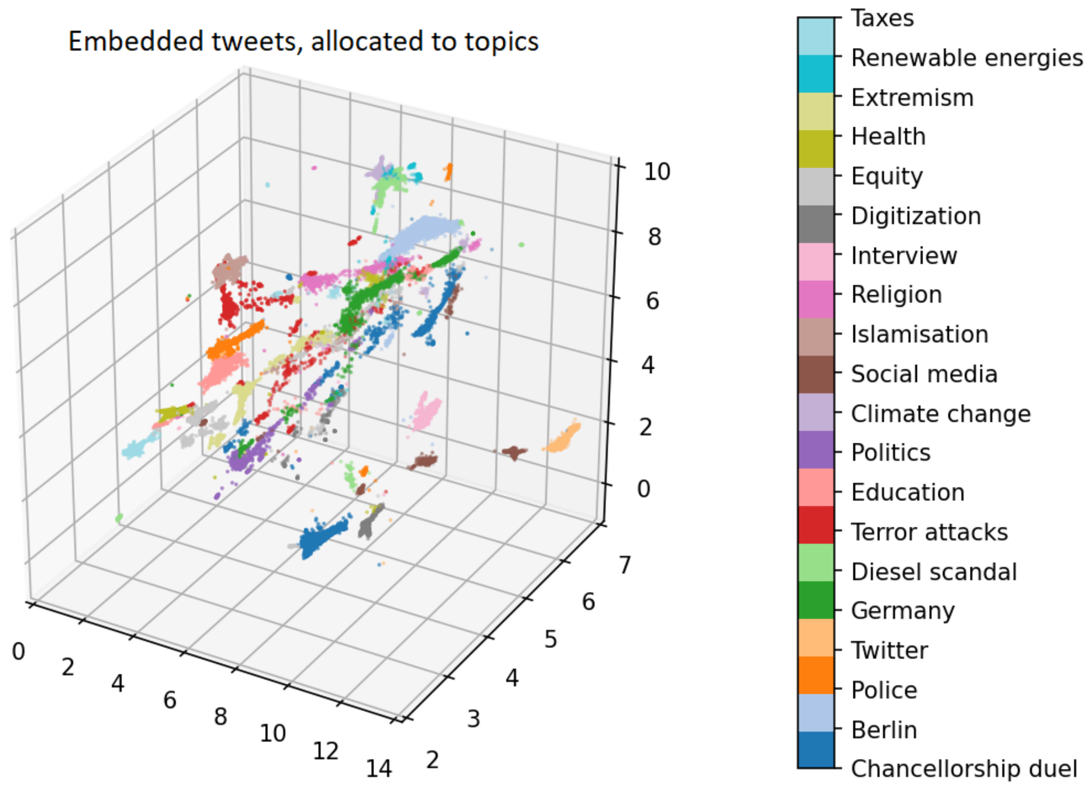


Figure 4.4.: Scatterplot of embedded tweets that are reduced to three dimensions using UMAP. Only tweets are shown that are assigned to a topic, therefore no noise included.

The low-dimensional tweet embeddings with their respective topic label are depicted in figure 4.4. While the embeddings get reduced to 10 dimensions in the BERTopic algorithm, the vector representations were reduced into three-dimensional space with UMAP in order to visualize the topics. The topics do form recognisable cluster, especially when moving away from the centre. The topics of tweets in the centre, in contrast, are more widespread. Tweets that are assigned, for example, to the topics concerning *Terror attacks, Extremism, Germany* or *Duel for Chancellorship*, blend into each other.

	Little preprocessing		**Strong preprocessing**	
Topic	**Duplicated tweets**	**Unique tweets**	**Duplicated tweets**	**Unique tweets**
0	Climate change	Climate change	Climate change	Climate change
1	Thankfulness	Thankfulness	Thankfulness	Renewable energies
2	Police	Politics	Police	Police
3	Digitization	Digitization	Digitization	Digitization
4	Chancellorship duel	Chancellorship duel	Chancellorship duel	Chancellorship duel
5	Election campaign	Election campaign	Chancellorship duel	Social media
6	Europe	Berlin	Berlin	Berlin
7	Freedom of press (Turkey)	Freedom of press (Turkey)	Freedom of press (Turkey)	Interview
8	Opinion polls	Opinion polls	Opinion polls	Politics
9	Equity	Human rights	Equity	Equity
10	Diesel scandal	Europe	Diesel scandal	Diesel scandal
11	Foreign affairs	Religion	Foreign affairs	Religion (Christianity)
12	Islamisation	Humans	Immigration	Islamisation
13	Taxes	Taxes	Taxes	Taxes
14	German Federal Armed Forces	Child poverty	Child poverty	Terror attacks
15	Pension	Health	Health	Health
16	Twitter	Twitter	Twitter	Twitter
17	Education	Education	Congratulations	Education
18	National debt (Greece)	Immigration	Sea rescue of refugees	Extremism
19	Violence at G20 summit	Violence at G20 summit	Terror attacks	Germany

Table 4.4.: Topics that are generated with SBERT for text embedding in BERTopic for all four data sets.

Tweets that are recognizable as distinct clusters belong to the topics, e.g., *Twitter, Taxes, Islamisation, Interview* or *Education*. Furthermore, the topics *Renewable energies, Climate change* and *Diesel scandal* are very close to each other which is plausible since their contents are linked. Also, the vector representations of tweets concerning the topics *Twitter, Social media* and *Interview* are located in the same direction, away from the centre.

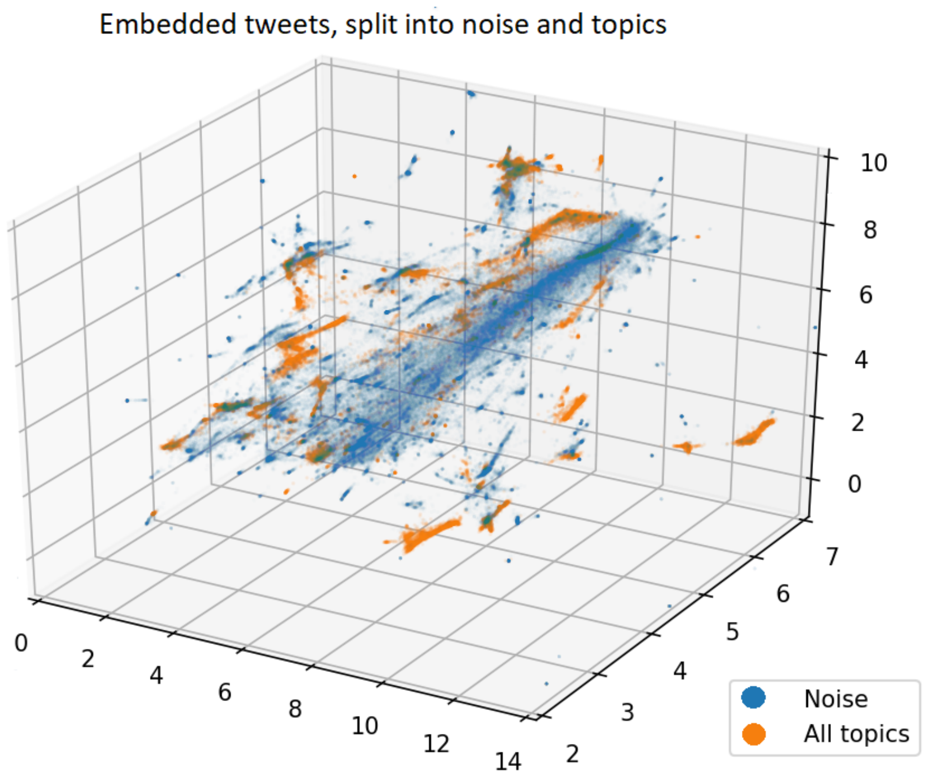


Figure 4.5.: Scatterplot of all embedded tweets, including noise, that are reduced to three dimensions using UMAP.

Figure 4.5 then shows all embedded tweets that were allocated to noise, coloured in blue, and the vector representations of tweets that belong to a topic in orange. This exemplifies that the majority of tweets are labelled as noise since the blue points dominate the plot. Nevertheless, especially the topics that are located further away from the centre do stand out.

The topics are labelled by consulting the top 10 representative words in German of each cluster which are shown in table 4.5. While some topics are more general, for example the topics with label *Berlin, Germany, Politics* or *Extremism*, most are easy to label such as topics regarding *Terror attacks, Religion* or *Education*. Furthermore, this table shows the number of tweets that are assigned to each topic. The topic concerning the *Duel for Chancellorship* contains the most, specifically 3230 tweets, while the smallest topic, which consists of 960 tweets, represents *Taxes*. Remember that the data set contains around 197000 tweets. Therefore, the sizes of the topics are just a small fraction of the total amount of given tweets. Nevertheless, the representative topics give the impression that the tweets of each resulting topic are coherent and semantically similar.

Following this, it is interesting whether the tweets that are assigned to the same topic are coherent. Table 4.6 lists representative tweets for the topics -1 *(Noise)*, 0 *(Duel for Chancellorship)*, 1 *(Berlin)*, 5 *(Diesel scandal)*, 6 *(Terror attacks)* and 19 *(Taxes)*. Those tweets are found by using a method of BERTopic which outputs representative documents for each topic. Most of the tweets shown in this table, but also other tweets that are not displayed here, seem to fit in their respective topic. However, there exist also tweets that we could assign to different topics. Tweets that subjectively do not represent their allocated topic are, for example

"laessig meldet barack obama zurueck" in topic *chancellorshipduel* and

"vorsitzender finanzausschuss us senat haelt unternehmenssteuerplaene donald trump kaum realisierbar" in topic *taxes*.

Tweets that are labelled as noise consist in turn of a variety of different topics. Some examples are also shown in table 4.6. Some tweets that can be found in noise are very short, such as "stimmt" and are therefore probably difficult to assign to a specific topic. Other tweets, for example

"herzlichen glueckwunsch geburtstag lieber" or

"danke kanzlerin merkel fordert erdogan insbesondere freilassung denizyuecel",

would fit into topics that are build when increasing the number of topics to $k = 38$ (see table B.2), but can not be assigned to any current topic with $k = 20$. But, then again, there exist also tweets labelled as noise that could have been put in topics, for example

"aufruf tegel schliessen zukunft oeffnen" in topic *Berlin* and

"herr schulz falsch verstanden denke seit minuten merkelvsschulz" in topic *Duel for Chancellorship*.

Topic	Count	Label	Representative words
0	3230	Chancellorshipduel	tvduell, schulz, tv, martin, duell, merkel, martinschulz, kanzlerduell
1	2855	Berlin	berlin, berliner, txl, btw, tegel, berlins, mitte, bundestag
2	2253	Police	polizei, polizisten, hamburger, hamburg, via, sicherheit, bundespolizei, polizeigewalt
3	2106	Twitter	retweeted, tweet, tweets, twittern, servicetweet, retweeten, linksfraktion
4	1737	Germany	traudichdeutschland, deutschen, holdirdeinlandzurueck, deutsche, btw, opposition, bayern, bundestag
5	1628	Diesel scandal	diesel, dieselgate, autoindustrie, diesesgipfel, dobrindt, automobilindustrie, fahrverbote, abgasskandal
6	1440	Terror attacks	barcelona, terror, terroristen, london, opfer, angehoerigen, anschlag, manchester
7	1381	Education	schulen, bildung, schule, bildungspolitik, lehrer, kooperationsverbot, studiengebuehren, ausbildung
8	1358	Politics	politik, politiker, politische, wolf, politikwechsel, menschen, politischer, political
9	1355	Climate change	klimaschutz, klimawandel, klima, klimakrise, darumgruen, trump, klimapolitik, planeten
10	1350	Social media	facebook, livestream, live, zensur, fb, meinungsfreiheit, whatsapp, netzdg
11	1257	Islamisation	islam, muslime, terror, islamisierung, oezoguz, ramadan, islamismus, islamisten
12	1207	Religion	kirche, kirchentag, christen, kirchen, religionsfreiheit, religion, stephanuskreis, religionen
13	1168	Interview	interview, sommerinterview, radio, interviewt, podcast, schulz, merkel, spitzenkandidatin
14	1163	Digitization	digitalisierung, digitale, bildung, digitalebildung, digitalen, schulen, agenda, zukunft
15	1150	Equity	gerechtigkeit, soziale, zeitfuermehrgerechtigkeit, zeitfuergerechtigkeit, sozial, zeit, zeitfuermartin, marktwirtschaft
16	1115	Health	pflege, gesundheit, pflegekraefte, patienten, weltgesundheitstag, gesundheitspolitik, antibiotika, personal
17	1024	Extremism	linksextremismus, populismus, rechtsextremismus, extremismus, linksextremisten, rechtsextreme, rechtspopulisten
18	963	Renewable energies	energiewende, windkraft, windenergie, erneuerbare, kohleausstieg, energien, klimaschutz, energiepolitik
19	960	Taxes	steuern, steuerkonzept, steuer, euro, panamapapers, einkommen, steuerzahler, steuersenkungen

Table 4.5.: Top 8 representative words for each of 20 topics with their respective label of the unique tweets with strong preprocessing.

Topic	Label	Representative sentences
-1	Noise	abschiebestopp afghanistan zeigt erneut merkels regierung fuerchtet negative schlagzeilen konsequenzen. na bitte zivil kenne schaerferes. herzlichen glueckwunsch geburtstag lieber. gut weiterhin erfolg. tod rechtsstaats kaempfen demokratie verein rechtsmissbrauch unterstuetzen. waehlen gehen saarland. stolz pulseofeurope brauchen unterstuetze gerne. fall griechenland loesen bevor fall italien eintritt. stimmt. danke kanzlerin merkel fordert erdogan insbesondere freilassung deniz yuecel. gute richtige entscheidung. aufruf tegel schliessen zukunft oeffnen. herr schulz falsch verstanden denke seit minuten merkelvsschulz.
0	Chancellorshipduel	erst glueck kam pech schulz ruft sieger duell. kanzlerduell zwei sozialdemokraten streiten schulz merkel. laessig meldet barack obama zurueck.
1	Berlin	weiterbetrieb txl fordert einzelmeinung o position uebernimmt bund mrd kosten. guten berlin. danke koblenz bekommt verstaerkung berlin klasse.
5	Diesel scandal	fahrverbote kalte enteignung autofahrer. herr dobrindt fahrverbote versagt grenzwerte einzuhalten schlagabtausch. diesel betrug resultat verflechtungen politik industrie lobbyismus parteispenden sagt.
6	Terror attacks	terror stoppen aufhoeren hinzusehen koennten freunde familien besser abschied nehmen. erschuettert traurig wann hoert terror endlich. thoooooomas fehlt.
19	Taxes	vorsitzender finanzausschuss us senat haelt unternehmenssteuerplaene donald trump kaum realisierbar. aufregen steuereinnahmen hoch nie einordnung. heinrich schoeller reeder millionen euro steuergeld geschenkt bekommt

Table 4.6.: Example representative tweets for certain topics of the unique tweets with strong preprocessing.

Furthermore, figure 4.6 is a heatmap of the similarity matrix between topic embeddings that are generated with the sentence transformers for the unique tweets with strong preprocessing. The topic embeddings are the weighted average of word embeddings, that represent their topic, based on their c-TF-IDF value[3]. The similarity of these topic embeddings is then computed by using cosine similarity S_C. Therefore, each rectangle represents a similarity score of two topics. The squares that lie on the diagonal all have a similarity score of 1, since two, equal topics are being compared.

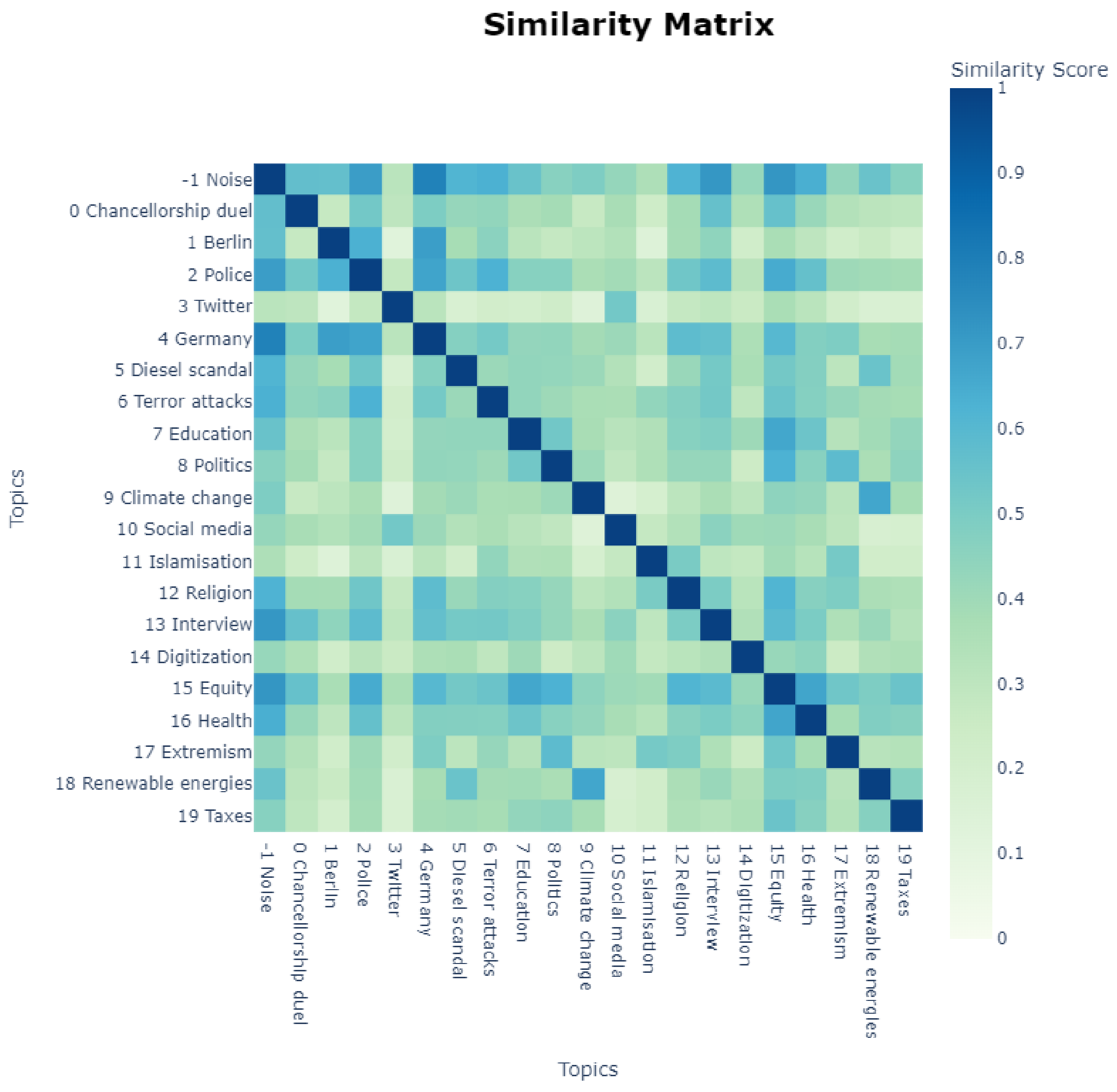


Figure 4.6.: Heatmap of the topic‘s similarity matrix, based on cosine similarity, which is adopted from Grootendorst (2020a). Unique tweets with strong preprocessing are embedded with Sentence BERT.

[3]See https://github.com/MaartenGr/BERTopic/blob/master/bertopic/_bertopic.py in lines 1527-1562 for more information about creating topic embeddings

The two topics with the lowest similarity score are labelled as *Twitter* and *Berlin.* These topic embeddings are therefore the most diverse. Remarkably, the cluster *Twitter* has low similarity scores with all other topics (all below 0.35), except with the topic *Social media* which results in a score of 0.52, approximately. Topics with a high similarity in contrast are for example *Berlin* and *Germany* ($S_C = 0.69$), *Police* and Germany ($S_C = 0.67$), *Climate change* and *Renewable energies* ($S_C = 0.67$) and *Equity* and *Education* ($S_C = 0.66$). Since one would expect that these topics may be similar, this shows that SBERT is able to capture semantics of tweets and that it generates meaningful topics.

However, topics such as *Germany, Interview* and *Equity* show the highest similarity score in combination with the cluster of noise, specifically $S_C = 0.79$, $S_C = 0.72$ and $S_C = 0.72$, respectively. Therefore, these topic embeddings are similar to the one of noise and might also underline that some tweets should be rather in existing topics than in noise.

4.3.2. German BERT

The German BERT model by Deepset (2019) makes working with German text data more efficient. It was pre-trained on 12 GB of German text data, specifically on German Wikipedia data (6GB of raw text), the OpenLegalData dump (2.4GB; Ostendorff et al., 2020) and news articles (3.6GB; Deepset, 2019).

The ten words that describe each resulting topic of the strongly preprocessed, unique tweets best, are shown in table 4.7. When looking at the words, it is difficult to derive meaningful topics from that since many words are rather general, for example, words in topics 3, 5 or 11. However, meaningful words that may contribute to labelling a topic, are often mixed. Topic 7, e.g. consists of the following words translated into English

> “Education, Islamisation, Examine Candidates, Equity, Constitution, Climate protection, Monday, Racism, Meaning, Economy“.

Each of those words may form its own cluster, but here they describe the same topic. This pattern can also be seen in topic 19. Nevertheless, some clusters can be labelled, for example topic 8 might be about *Congratulations*, topic 9 about *Opinion polls* since

Dimap[4] and Ipsos[5] do psephology. Furthermore, topic 11 could consist of tweets regarding the *Duel for Chancellorship*. All other resulting topics are more difficult to label. Table 4.7 furthermore displays the number of tweets in each topic. It is interesting to see that around 137000 tweets (70%) are assigned to topic 0, whereas only 25.5% are labelled as noise. After this first, huge topic, the clusters get much smaller, topics 1 to 19 particularly only comprise approximately 4.9% in total of all tweets.

Since 70% of the tweets are assigned to topic 0, table 4.8 contains examples of representative documents for both, topic 0 and noise. We can see that the cluster noise really consists of mixed topics, such as expansion of the fiber optic network, rent and a topic concerning elections. The other four tweets in noise are rather short and are more difficult to assign to a topic. The tweets "sowas bereit live" and "gleich gehts los" may refer to the duel for chancellorship, since they are announcements which can be often found in the tweets at hand.

When looking at the tweets in topic 0, they do not seem to be coherent. The first text in table 4.8 relates to libraries and book, whereas the second tweet refers to a murder trial in Germany. The other tweets are also very short and are about closing borders, closing the airport Tegel in Berlin, the 'Landtag' in Magdeburg and probably about the election results. We can see, therefore, that topic 0 also acts as a noisy cluster.

When comparing the representative words for each topic with SBERT and German BERT as embedding models, we notice that the words generated with SBERT form more coherent topics and are easier to label than the words resulting from German BERT as embedding model.

[4]Visit https://www.infratest-dimap.de/ for more details.
[5]Visit https://www.ipsos.com/de-de for more details.

Topic	Count	Representative words
-1	50348	danke, dabei, merkel, ehefueralle, wahlkampf, wahl, leider, klar, wohl, na
0	137274	berlin, nrw, danke, politik, gruenen, traudichdeutschland, neue, europa, tag, seit
1	1977	bpt, traudichdeutschland, schlussrunde, merkel, kandidatencheck, wahlprogramm, land, einfachmachen, muenster, gute
2	1498	merkel, migranten, europa, einwanderer, oesterreich, neues, erdogan, nrw, fluechtlinge, wegen
3	656	frohe, ostern, gott, peinlich, waere, wohl, welt, weiss, abend, waehlt
4	524	wahlprogramm, programm, wirtschaft, werbung, bund, arbeitnehmer, bilanz, investitionen, jahre, luft
5	437	wahr, wissen, leider, tut, lass, passiert, schlecht, ernst, stimme, halt
6	434	oh, hae, och, waere, besuch, oha, spaet, schoen, ok, nich
7	417	bildung, islamisierung, kandidatencheck, gerechtigkeit, grundgesetz, klimaschutz, monday, rassismus, bedeutung, wirtschaft
8	415	herzlichen, dank, glueckwunsch, tolles, unterstuetzung, toller, schliesse, besuch, grossartige, glueckwuensche
9	384	sonntagsfrage, landtagswahl, dimap, landesparteitag, ard, kandidaten, ipsos, faz, neuwahlen, umfrage
10	384	interview, infostand, online, hinzugefuegt, lesen, infos, besuch, bade, stupid, eingebettet
11	335	tvduell, merkel, martin, schlusswort, tipp, strunz, moderator, update, telefonliste, abend
12	334	zukunftwirdausmutgemacht, mutlufuermitte, wahlkampf, hassistkeinemeinung, schauenwirnichtlaengerzu, innenstadt, wartenwirnichtlaenger, weilwirdichlieben, einervonuns, zukunft
13	315	willkommen, veranstaltung, unserer, gaeste, gmuend, bericht, termine, themen, informative, lorch
14	287	gehoert, geschaetzt, geaendert, gefaellt, gefruehstueckt, abgeholt, aufgeschrieben, leider, geprueft, gekehrt
15	268	schoenen, tag, allerseits, schoenes, nachmittag, laeuft, heimat, druecke, schoene, urlaub
16	260	eurovision, fedidwgugl, rumaenien, kanonen, team, durchhalten, punkte, frueher, leben, ganz
17	259	genau, ganz, finde, leider, unsinn, schlecht, wohl, entgeht, trifft, darauf
18	219	oh, ltwsh, hach, spdbpt, aehm, ueberall, csuler, oekofete, sed, jusos
19	212	rentenkonzept, diskriminierung, freiheit, sicherheit, ideologie, verfassungsschutz, wertegemeinschaft, voraussetzungen, mehrwertsteuer, existenz

Table 4.7.: Top 10 representative words for the unique tweets with strong preprocessing. German BERT is used to embed the tweets.

Topic	Representative sentences
-1	glasfaserausbau dornroeschenschlaf setzt stattdessen
	mietpreisbremse ausbauen statt abschaffen linke
	wahlomat getestet ergebnis eindeutig daher wohl waehlen
	gut gerne leben
	sinne menschen
	sowas bereit live
	gleich gehts los
0	erhalten bibliotheken kuenftig buecher stange
	fremdwahrnehmung verantwortung gutachter sass nsu prozess frau zschaepe via
	grenzen schliessen
	tegelschliessen ruhe geniessen
	gestern landtag magdeburg
	gibt offizielles ergebnis

Table 4.8.: Example representative texts for the unique tweets with strong preprocessing that were either allocated to noise or topic 0.

Furthermore, figure 4.7 depicts the heatmap of the similarity matrix between topic embeddings that are generated with German BERT for the tweets of the data set $D_{s,u}$. In this figure, the top two representative words of each topic are displayed on both the x-axis and y-axis. Each rectangle also represents a similarity score of two topics here, but note that the scale of the similarity score starts at 0.5 instead of 0. Additionally, we see that each combination of two topics has a score higher than 0.7 whereas the lowest similarity score with SBERT as embedding model was approximately 0.12. Hence, we know that the topic embeddings generated with SBERT are more diverse than the ones by German BERT. The two topics that are most similar are topics 0 and 2 with a similarity score of 0.9845. Topic 0 is very general, whereas topic 2 might be about *Immigration*. The lowest similarity scores can always be found when combining topic 12 with others. The representative words of topic 12 seem to be hashtags that could not have been split back into their original words. The lowest similarity score is 0.7486 of topics 12 and 5.

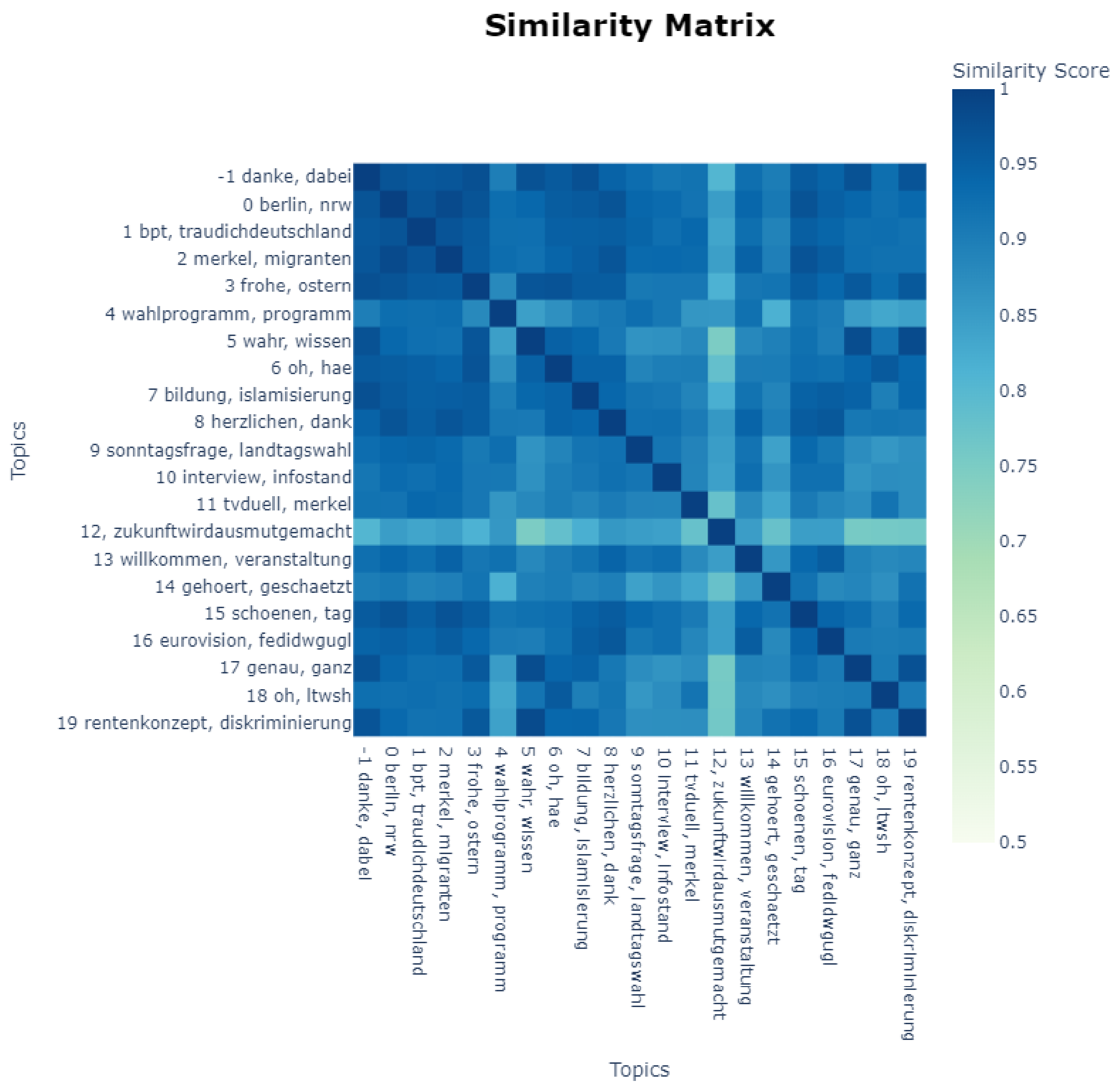


Figure 4.7.: Heatmap of the topic‘s similarity matrix, based on cosine similarity, which is adopted from Grootendorst (2020a). Unique tweets with strong preprocessing are embedded with German BERT.

4.3.3. GottBERT

GottBERT which was introduced by (Scheible et al., 2020), is a RoBERTa model for German only. The model was trained on the German parts of the OSCAR[6] data set which comprises 145GB of text.

Here, GottBERT was used to embed the tweets. The representative words are then found by using class-based TF-IDF. The top ten representative words for the application of this model on the strongly preprocessed, unique tweets are shown in table 4.9. These topics are also not labelled because here again the words within a topic are mixed and some words are very general and are therefore not useful in order to label the topics. Furthermore, some words occur in more than one topic. For example, the word "campaign" represents topics 1, 2, 5, 6, 14 and 18, therefore in total six topics.

Topics that might be labelled are for example topic 4 as *Congratulations* and cluster 19 as *Duel for Chancellorship*. Note that these two topics can also be found when using German BERT as embedding model. Moreover, around 90% of the tweets are assigned to noise. This exceeds both values of topic modelling with SBERT (84% to noise) and German BERT (25.5% to noise).

Figure 4.8 displays the heatmap for this topic model. The first 15 and the last 3 topics are apparently very similar to each other because all similarity scores are higher than 0.96. Topics 16 and especially 15 stand out. Although they are very similar with a similarity score of 0.9535, topics 15 and 16 have an average score of 0.79 and 0.91, respectively, to all other topics. The most dissimilar pair are topics 13 and 15 with a score of 0.7533. Here, topic 15 seems to be very general, whereas topic 13 might be about *Turkey*.

[6]Visit https://oscar-corpus.com/ for more details.

Topic	**Count**	**Representative words**
-1	177011	btw, danke, gut, nrw, wahlkampf, tvduell, gruenen, wahl, ganz, neue
0	3466	genau, na, glaube, gut, danke, tweet, link, thema, ganz, willkommen
1	1958	btw, einzelfall, tzt, deinestimme, rawert, stimme, wahlkampf, migrantenkriminalitaet, unterwegs, gera
2	1690	berliner, leben, nrw, politik, schueler, wahlkampf, polizisten, polizei, gut, militaer
3	1392	migranten, ak, migrant, facebook, altenkirchen, terror, migration, zensur, trump, gold
4	1243	dank, herzlichen, vielen, danke, frohe, muslimen, engagement, super, schoenen, ergebnis
5	1132	wahlkampf, gestern, leben, neue, terror, ganz, tvduell, btw, fluechtlinge, gruenen
6	938	nrw, ms, landtag, landtagswahl, sh, btw, holstein, mai, wahlkampf, ergebnis
7	853	btw, gut, neue, unserer, thema, wurde, tvduell, gehts, migranten, gestern
8	843	steineke, abend, btw, besuch, gute, viele, austausch, gaeste, interessante, veranstaltung
9	781	berliner, neue, gestern, gruenen, migranten, besuch, gut, wurden, nrw, gast
10	774	btw, zeit, lsa, buerger, nrw, deutschen, kinder, sonntag, stimmen, anmelden
11	739	wurde, gespannt, verstehen, darf, na, gut, sonntag, zahn, jahren, zeit
12	686	danke, guter, hinweis, folgen, engagement, super, kommentar, lieber, btw, oh
13	647	tuerkeireferendum, istanbul, verhaftet, hdp, geheimdienst, tuerkischer, diktatur, akp, rheinmetall, bundeswehr
14	628	gruenen, klimaschutz, gerechtigkeit, stimme, zweitstimme, platz, gut, gesundheit, wahlkampf, katrin
15	514	sehen, ordnung, passt, btw, spass, urlaub, ok, warten, viele, zeichen
16	499	kampf, ehe, partei, kommentar, terror, reden, ging, kinder, facebook, leichteren
17	490	btw, film, gestern, themen, interessante, rente, sicherheit, dank, danke, frankfurt
18	483	btw, teampaul, tzt, regierungsprogramm, wahlkampf, gut, erzgebirge, tvduell, gehts, kanzlerin
19	470	tvduell, duell, nachlesen, kanzlerduell, strunz, rente, kanzlerin, rum, schroeder, trump

Table 4.9.: Top 10 representative words for the unique tweets with strong preprocessing. GottBERT is used to embed the tweets.

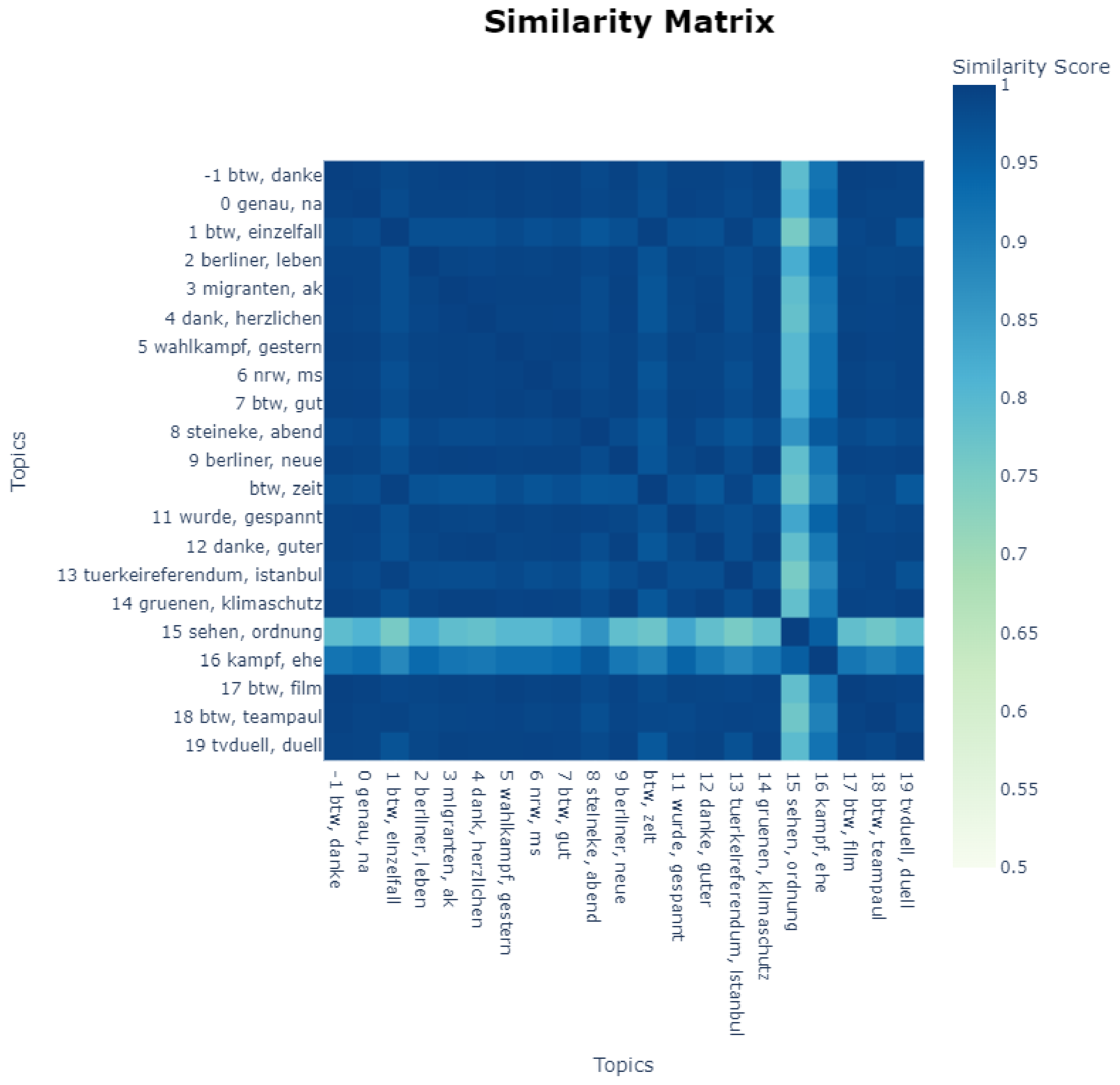


Figure 4.8.: Heatmap of the topic‘s similarity matrix, based on cosine similarity, which is adopted from Grootendorst (2020a). Unique tweets with strong preprocessing are embedded with GottBERT.

5. Discussion and Outlook

The last chapter showed that meaningful topics can be built when using Sentence BERT as embedding model for BERTopic. However, the resulting topics when using German BERT or GottBERT to embed the tweets, do not seem to capture the meaning of a tweet since the topics can not be easily interpreted. This is interesting to see because both, GottBERT and German BERT are German models only, whereas the model used with SBERT is a multilingual one. According to Scheible et al. (2020), "multilingual models are inferior to monolingual models", which could not be confirmed throughout this thesis. The author of BERTopic, Grootendorst (2020b), advises using sentence transformers (SBERT) to embed documents since they are of good quality and show state-of-the-art results on tasks such as semantic textual similarity (Reimers and Gurevych, 2019). As a result, we did see that the usage of SBERT as embedding model lead to the most meaningful and coherent topics.

Nevertheless, many tweets (around 84%) are labelled as outliers because they have low specificity towards the resulting, cohesive topics. But we also noticed that tweets in noise are allocated to new topics when the number of topics is increased. While we evaluated topic models with 20 topics for the sake of simplicity, 50 topics also form coherent topics and would therefore also be reasonable. Note that table B.2 shows the representative words for $k = 50$ and the topic model with SBERT as embedding model.

In the following section, we propose an idea, how the topic model may still be advanced and how the number of tweets assigned to noise might be reduced. After that, specialized approaches are described that can be applied when using short texts for topic modelling. The idea is to aggregate certain, connected tweets into longer documents by leveraging its meta-data.

5.1. Possible ways to reduce noise

The algorithm BERTopic which was used for topic modelling in this thesis, performs all steps, embedding documents, reducing dimensions, clustering low-dimensional vector representations and applying class-based TF-IDF, in a single pass. If we want to try different values for the hyperparameters, all steps hence need to be repeated all over again. Therefore, it is worth considering performing all steps successively in order to be able to optimize the parameters, for example when using UMAP or HDBSCAN. This may lead to even better results where those tweets that could have been allocated to topics are not labelled as noise anymore, since we are able to customize our models.

Another possibility to reduce the size of the cluster noise is to calculate the probability of a tweet belonging to any topic by setting `calculate_probabilities = True` in BERTopic. By doing this, we can re-allocate outliers to the topic with the highest probability after the model had been trained. Note, however, that the extraction of topics needs an immense computation time when calculating the probabilities of all topics per document and should therefore only be used if we have less than 100000 documents (Grootendorst, 2020a). Furthermore, we should probably define a threshold below which the documents can not be re-allocated to topics since the concept of noise does make sense. The purpose of noise is to collect all those tweets that can not be assigned to any topics. Especially when looking at the tweets at hand, we noticed that many tweets just consist of one or two words after preprocessing. The approach of labelling those tweets as noise may therefore be reasonable.

5.2. Specialized approaches for short texts by leveraging meta-data

Although tweets may vary in length, many of them can be very short[1], which may lead to a lack of context in each tweet (Feng et al., 2020). Since the average amount of words in tweets with strong preprocessing are 7 words and the average tweet with little preprocessing consists of 13 words, the tweets at hand that are fed to embedding models are in fact short texts.

[1]In 2017, the maximum length of a tweet was set to 140 characters.

There exist so-called “aggregation strategies“ that are specifically used in social media contexts that leverage their meta-data. Weng et al. (2010), for example, aggregated tweets written from the same authors and then trained a topic model. Furthermore, Hong and Davison (2010) compare the results of LDA models with different aggregation strategies such as aggregated user profiles or aggregation of all texts that contain a certain term for all terms in the training corpus. Although we did not use LDA for topic modelling, but instead a neural topic model, our data consists apart from the tweets of meta-data which could be leveraged to aggregate the tweets. First, we know the author of each tweet and could therefore aggregate all tweets based on the user. However, in a time period of 6 months, it would not be surprising if the topics will vary, especially since politicians need to address recent events and news. Second, we know the exact date and time, a tweet was posted. Hence, we could aggregate the tweets based on the date which would capture the turn of events. While a time window of one day may be too large, we could think about aggregating all tweets that were posted in the same hour and the same day. Third, it is specified whether a tweet is a retweet, original tweet, reply or a quote. Moreover, a unique tweet ID is given, referring to the original tweet. We could thus recap a conversation, for example, by aggregating the original tweets with the reply to it. However, it would be necessary that the authors of the respective original tweet belong to those candidates whose tweets are scraped from Twitter. Lastly, we can extract all hashtags in texts and aggregate the tweets based on those hashtags. Since “hashtags can be viewed as topical markers, an indication to the context of the tweet or as the core idea expressed in the tweet“ (Mehrotra et al., 2013), this may also be a promising approach.

6. Conclusion

In this thesis, we applied BERTopic, a neural topic model, to tweets written by German politicians. The first step of this algorithm is to embed the tweets at hand. We used three different models to generate these vector representations: Sentence BERT, German BERT and GottBERT. While SBERT is a multilingual model, the other two are for German texts only. We found that the sentence transformer model is most appropriate in order to find meaningful and coherent topics. For that reason, we fitted BERTopic with the sentence transformer on four different data sets: tweets with little and strong preprocessing and unique and duplicate tweets, respectively. The results of all four data sets in combination with SBERT as embedding model lead to very similar results. Moreover, we discovered that the topic models with 50 topics hold the most information when comparing them to the fitted models with 10, 20, 30 and 40 topics. Since we started initially with 269000 tweets, it is obvious that many different topics are probably being discussed. Although choosing 50 topics seemed most appropriate, we decided to evaluate the models with 20 topics in more depth to obviate information overload.

When taking a closer at the model with 20 topics and SBERT as embedding model, we found that not only the representative words for each cluster are coherent, but also the representative tweets. Nevertheless, some tweets that are labelled as noise, could have also been assigned to existing clusters. This might be resolved by calculating the probabilities of all topics per document and re-allocate those tweets in noise above a certain threshold to their highest respective topic probability. As the unique tweets with strong preprocessing allocated the greatest percentage of tweets (16%) to topics, we focused for further analysis on this data set. Moreover, the unique tweets with strong preprocessing were visualized in a three-dimensional space, where many topics were visible as distinct clusters. Other topics were more widespread but still recognizable as clusters. Lastly, we examined a heatmap that visualized the cosine similarity scores of each pair of topic embeddings. We found that those clusters that share a similar topic, also have a higher similarity score and those that describe different topics thus have a lower score.

When the topic model with German BERT as embedding model and the unique tweets with strong preprocessing was fit, it was interesting to see that only 25.5% of the tweets were labelled as noise, whereas 70% are assigned to topic 0. Nevertheless, the representative words of the topics are very general and it is, therefore, difficult to label each topic. Hence, it seems that BERTopic in combination with SBERT generated more valuable and meaningful topics. This suspicion strengthens when looking at the heatmap that depicts the similarity matrix of the topic embeddings generated by German BERT. Not only do the scores start at a higher threshold, but also many similarity scores are close to one.

A similar pattern can be observed when generating the embeddings with GottBERT. The representative words for each topic are rather general and the topics can therefore not be labelled. Nearly all topic embeddings have moreover a cosine similarity score close to one. A difference compared to German BERT as embedding model is that approximately 90% of the tweets are allocated to noise.

For all results we have evaluated in this thesis, we used the model BERTopic introduced by Grootendorst (2020a). It is a powerful topic modelling algorithm because it combines transformer-based models such as BERT with TF-IDF in order to find representative words for each topic and to interpret the topics. Furthermore, it is arguably straightforward to apply and comprises great visualization tools. It generated meaningful and cohesive topics when combining it with Sentence BERT as embedding model, although 24.3% of the tweets were cut off at the end. It would be interesting to see for future topic modelling how the results will change with those tweets pulled from Twitter that are not cut off in the end.

A. Details on preprocessing of tweets

Since the data was provided by Weingärtner (2021) who had already preprocessed the tweets, we adopted some of his code. The following will specify which passages of preprocessing were transferred.

```r
library(tidyverse)
library(stringr)
library(stringi)
library(quanteda)
library(data.table)
library(Rcpp)
library(qdapRegex)

tweets <- readRDS("data_tweets.rds")

little_preprocessing <- function(tweets) {
  tweets <- rm_url(tweets)
  tweets <- gsub("\\b+RT", "", tweets)
  tweets <- gsub("@\\S+", "", tweets)
  tweets <- gsub("\\s[[:graph:]]+...$", "", tweets)
  tweets <- stringi::stri_trans_general(tweets,
    "Any-Latin") # Weingaertner
  tweets <- stringr::str_replace_all(tweets,
    c("\u00c4" = "Ae",
      "\u00e4" = "ae",
      "\u00d6" = "Oe",
      "\u00f6" = "oe",
      "\u00dc" = "Ue",
      "\u00fc" = "ue",
      "\u00df" = "ss")) # Weingaertner
  tweets <- stringr::str_replace_all(tweets,
    pattern = "\\n",
    replacement = " ") # Weingaertner
  tweets <- gsub("&amp;", "und", tweets)
  tweets <- gsub("&gt;", "", tweets)
  tweets <- gsub("&lt;", "", tweets)
  pattern <- stringr::str_c(c(
    "\U0022",
    "\U0027",
    "\U2018",
    "\U2019",
    "\U201C",
    "\U201D",
    "\U201E",
    "\U201F",
    "%",
    " http([^ ]*)",
    "http([^ ]*)",
    "\\\n"),
    collapse = "|") # Weingaertner
  tweets <-  stringr::str_remove_all(tweets, pattern) # Weingaertner
  tweets <- gsub("[^ -~]", "", tweets)
  tweets <- gsub('[[:punct:]]', ' ', tweets) # Strong preprocessing
  tweets <- gsub("\\d", " ", tweets) # Strong preprocessing
  tweets <- gsub("^[[:space:]]*", "", tweets)
  tweets <- gsub("[[:space:]]*$", "", tweets)
  tweets <- gsub(" +", " ", tweets)

  return(tweets)
}

tweets$text <- little_preprocessing(tweets[, "text"])

# The following passage is fully adopted by Weingaertner
pattern_hashtag <- "(#)[[:alnum:]]+"
pattern_camelcase_hashtag <- "#(.)+[:upper:][:lower:]{2,}"
pattern_split_camelcase <- "(?<=[:lower:])(?=[:upper:])"

tweets <- as.data.table(tweets)
tweets[, text := lapply(
  .I,
  function(i) {
    components <- unlist(stringr::str_split(text[i], " "))
    case_numbers <- which(stringr::str_detect(
      components, pattern_camelcase_hashtag))
    cases <- components[case_numbers]
    solved_cases <- sapply(
      stringr::str_split(cases, pattern_split_camelcase),
      function(j) paste0(c(j), collapse = " "))
    components[case_numbers] <- solved_cases
    paste0(c(components), collapse = " ")})]

tweets[, text := stringr::str_remove_all(text, pattern_hashtag)]
```

Listing A.1: Preprocessing example

This part of preprocessing was performed in R. Everything following this preprocessing, for example extracting and deleting tweets in non-German languages, was done in the scope of this thesis in Python [1].

[1] All codes can be found here: https://github.com/annegriddl/Neural-topic-modeling.git

B. Details on topics

This part comprises more tables with representative words for the four data sets with little and strong preprocessing as well as with unique and duplicate tweets for 20 topics. Furthermore, the representative words for 50 topics and the data set with strong preprocessing and unique tweets is displayed. Moreover, the list of stopwords is shown in table B.1

Stopwords						
ab	aber	afd	alle	allem	allen	aller
alles	als	also	am	an	ander	andere
anderem	anderen	anderer	anderes	anderm	andern	anderr
anders	auch	auf	aus	bei	beim	bin
bis	bist	cdu	cducsu	csu	da	dafuer
damit	dann	darum	dafuer	der	den	des
dem	die	das	dass	dass	derselbe	derselben
denselben	desselben	demselben	dieselbe	dieselben	dasselbe	dazu
dein	deine	deinem	deinen	deiner	deines	denn
derer	dessen	dich	dir	du	dies	diese
diesem	diesen	dieser	dieses	doch	dort	durch
deutschland	deutschlands	ein	eine	einem	einen	einer
eines	einig	einige	einigem	einigen	einiger	einiges
eigentlich	einmal	er	ihn	ihm	es	etwas
euer	eure	eurem	euren	eurer	eures	fuer
fdp	gegen	gerade	gewesen	hab	habe	haben
hat	ham	hatte	hatten	haette	haetten	hallo
hier	hin	hinter	ht	heute	ich	mich
mir	immer	ihr	ihre	ihrem	ihren	ihrer
ihres	euch	im	in	indem	ins	ist
ja	jede	jedem	jeden	jeder	jedes	jene
jenem	jenen	jemand	jener	jenes	jetzt	kann
kein	keine	keinem	keinen	keiner	keines	koennen
koennte	machen	man	manche	manchem	manchen	mancher
manches	mein	meine	meinem	meinen	meiner	meines
meines	mehr	mit	muss	musste	muessen	morgen
gruen	nach	nein	nicht	nichts	noch	nun
nur	ob	oder	ohne	schon	se	spd
sehr	sein	seine	seinem	seinen	seiner	seines
selbst	sich	sie	ihnen	sind	so	solche
solchem	solchen	solcher	solches	soll	sollte	sollten
son	sondern	sonst	ueber	um	und	uns
unsere	unserem	unseren	unser	unseres	unter	uhr
tl	viel	vielleicht	vom	von	vor	waehrend
war	warum	waren	warst	was	weg	weil
weiter	welche	welchem	welchen	welcher	welches	wenn
werde	werden	wie	wieder	will	wir	wird
wirst	wo	wollen	wollte	wuerde	wuerden	zu
zum	zur	zwar	zwischen			

Table B.1.: List of used stopwords.

Topic	Representative words
0	berlin, berliner, txl, tegel, berlins, mitte, btw, bundestag
1	tvduell, tv, duell, schulz, merkel, martin, moderatoren, kanzlerduell
2	polizei, polizisten, hamburger, hamburg, bundespolizei, polizeigewalt, sicherheit, hh
3	retweeted, twitter, tweet, tweets, twittern, servicetweet, retweeten, linksfraktion
4	traudichdeutschland, deutschen, holdirdeinlandzurueck, deutsche, opposition, bayern, kraft, bundestag
5	schulen, bildung, schule, bildungspolitik, lehrer, kooperationsverbot, studiengebuehren, ausbildung
6	politik, politiker, politische, politischen, politisch, wolf, politikwechsel, political
7	klimaschutz, klimawandel, klimakrise, darumgruen, klimapolitik, trump, planeten, klimaziele
8	facebook, livestream, live, zensur, fb, whatsapp, meinungsfreiheit, netzdg
9	islam, muslime, islamisierung, terror, oezoguz, ramadan, islamismus, islamisten
10	kirche, kirchentag, christen, kirchen, religionsfreiheit, religion, stephanuskreis, religionen
11	interview, sommerinterview, radio, interviewt, podcast, btw, nachhoeren, spitzenkandidatin
12	digitalisierung, digitale, digital, bildung, digitalebildung, digitalen, schulen, agenda
13	gerechtigkeit, soziale, zeitfuermehrgerechtigkeit, sozialen, sozial, marktwirtschaft, zeitfuermartin, zeit
14	pflege, gesundheit, pflegekraefte, patienten, weltgesundheitstag, versorgung, gesundheitspolitik, antibiotika
15	linksextremismus, populismus, rechtsextremismus, extremismus, rechtsextreme, rechtspopulisten, extremisten
16	diesel, autoindustrie, automobilindustrie, luft, fahrverbote, dieselgipfel, autos, stickoxide
17	energiewende, windkraft, wind, windenergie, erneuerbare, energien, kohleausstieg, energie
18	steuern, steuerkonzept, steuer, panamapapers, euro, einkommen, steuerzahler, steuersenkungen
19	journalisten, pressefreiheit, freedeniz, tuerkei, journalist, deniz, yuecel, journalismus
20	familien, familie, familiennachzug, familienpolitik, kinder, wuensche, papa, luftballons
21	migranten, einwanderungsgesetz, migration, illegale, einwanderung, zuwanderung, migrationshintergrund
22	union, sonntagsfrage, nordrhein, bundestagswahl, wahlkampf, btw, wahlumfrage, forsa
23	euro, schulden, eu, banken, millionen, bank, finanzkrise, eurozone
24	italien, libyen, mittelmeer, kuestenwache, eu, seenotrettung, ngos, fluechtlinge
25	demokratie, direkte, namen, europa, sozialdemokratie, demokratisch, waehlen, freiheit
26	glueckwunsch, herzlichen, geburtstag, birthday, happy, glueck, liebe, glueckwuensche,
27	essen, verbraucherschutz, skandal, eier, fipronil, verbraucher, lecker, milch
28	homophobie, queer, heiraten, lgbti, wedding, homosexuelle, ehefueralle, tschetschenien
29	elektromobilitaet, mobilitaet, verkehrswende, elektroautos, elektroauto, autos, verkehrspolitik, radverkehr
30	martin, schulz, martinschulz, kanzlerkandidat, sozialdemokraten, obama, kanzler, zeitfuermartin
31	danke, dank, vielen, unterstuetzung, beste, feedback, freut, veranstaltung
32	minister, ministerin, innenminister, verkehrsminister, finanzminister, ministerpraesident, dobrindt, schaeuble
33	armut, kinderarmut, kinder, ungleichheit, grundeinkommen, kindergrundsicherung, einkommen, armutsbericht
34	protest, demo, proteste, demonstranten, hamburg, friedlich, demonstrieren, hh
35	foto, bild, fotos, bilder, instagram, account, kamera, fotoshooting
36	terror, terroristen, stockholm, terrorismus, manchester, gedanken, angehoerigen, opfern
37	afghanistan, abschiebungen, abschiebestopp, kabul, afghanen, abschiebung, afghane, taliban
38	nazis, nazi, neonazis, neonazi, nonazis, hitler, helden, nationalsozialismus
39	barcelona, london, terror, opfer, angehoerigen, anschlag, trauer, gedanken
40	antisemitismus, israel, juden, antisemitismusdoku, doku, gabriel, jerusalem, antisemiten
41	feminismus, gender, sexismus, frauen, diskriminierung, gleichberechtigung, feministin, geschlechtergerechtigkeit
42	bundeswehr, haushaltsausschuss, mali, soldaten, drohnen, engagement, kampfdrohnen, truppe
43	trump, nordkorea, usa, charlottesville, aussenpolitik, praesident, kim, korea
44	asyl, asylbewerber, asylrecht, refugeeswelcome, asylpolitik, refugees, asylkrise, einwanderung
45	plakate, plakat, haengen, wahlplakate, plakaten, btw, vandalismus, logo
46	diskussion, dialog, interessant, spannende, freue, spannend, spass, interessante,
47	jamaika, koalition, koalitionsvertrag, nrwkoalition, nrw, gruene, koalitionsverhandlungen, vertrag
48	hamburg, olaf, scholz, gipfel, nog, tk, hamburgs, gipfels
49	dieselgate, diesel, software, abgasskandal, dobrindt, lobbyismus, dieselgipfel, dieselskandal

Table B.2.: Top 8 representative words for each of 50 topics of the unique tweets with strong preprocessing.

Topic	Count	Representative words
0	2686	polizei, polizisten, hilfe, sicherheit, polizist, polizistinnen, bundespolizei, police, polizeigewalt, polizeieinsatz
1	1998	tweet, twitter, tweets, twittern, retweeten, servicetweet, getwittert, lesen, twittert, retweetet
2	1846	europa, kohl, helmut, pulseofeurope, europe, populismus, europaeer, eu, europaeische, pulse
3	1805	steuerkonzept, steuern, einkommen, entlasten, steuersenkungen, steuerpolitik, steuer, entlastet, zahlen, vermoegensteuer
4	1663	trump, syrien, nordkorea, paris, usa, venezuela, syrer, donald, pariser, agreement
5	1559	grundgesetz, rechtsstaat, gesetz, kinderrechte, verfassungsschutz, gesetze, leitkultur, recht, netzdg, kultur
6	1548	diesel, dieselgate, dobrindt, abgasskandal, fahrverbote, dieselgipfel, luft, autokartell, stickoxide, softwareupdate
7	1507	polizei, polizisten, g20, polizeigewalt, polizist, hamburg, sicherheit, nog20, polizistinnen, gewalt
8	1446	digitalisierung, tvduell, infrastruktur, bildung, zukunft, nix, digital, verkehrsminister, schlechteste, klima
9	1430	klimaschutz, klimawandel, klimakrise, klima, gruen, arktis, irma, ueberschwemmungen, wetter, kohleausstieg
10	1339	danke, vielen, hol, zurueck, follower, dank, landzurueck, land, btw17, folgen
11	1302	martin, schulz, gerechtigkeit, 70, rente, tvduell, merkel, zeitfuer, tvspot, machts
12	1293	euro, griechenland, milliarden, schulden, mio, roaminggebuehren, millionen, verschuldet, eu, bahn
13	1286	deniz, free, journalisten, pressefreiheit, yuecel, tuerkei, haft, journalismus, tagen, inhaftierten
14	1273	islam, muslime, islamisierung, terror, islamismus, islamisten, trau, moschee, islamischen, gehoert
15	1195	sonntagsfrage, bundestagswahl, forsa, btw17, 39, sternl, emnidbams, 24, 65, projektion
16	1036	bildung, schulen, schule, investieren, ausbildung, lehrer, investitionen, studiengebuehren, bildungspolitik, hochschulen
17	981	rente, buergerversicherung, gruensozial, sozialpolitik, forderungen, rentenniveau, gruene, garantierente, befristung, altersarmut
18	920	bundeswehr, incirlik, soldaten, kampfdrohnen, feuerwehr, abzug, drohnen, freiwilligen, waffen, leyen
19	903	woche, waehlen, wahltag, ehe, abstimmen, abstimmung, steineke, sekunden, beschliessen, bundestag

Table B.3.: Top 10 representative words for each of 20 topics of the duplicate tweets with little preprocessing.

Topic	Count	Representative words
0	3230	mann, kohl, helmut, recht, weiss, peinlich, heiner, geissler, herr, kanzlerin
1	2855	europa, eu, pulseofeurope, europe, europaeische, europas, pulse, europaeischen, of, worms
2	2253	digitalisierung, digitale, bildung, digital, digitalen, zukunft, agenda, digitaler, schulen, digitalgipfel
3	2106	bildung, schulen, schule, investieren, ausbildung, investitionen, lehrer, studiengebuehren, bildungspolitik, kinder
4	1737	politik, parteien, partei, politiker, altparteien, opposition, politische, politischen, populismus, politisch
5	1628	berlin, guten, berliner, btw17, bundestag, gehts, stadt, sitzungswoche, tegel, blog
6	1440	tvduell, duell, tv, schulz, merkel, martin, moderatoren, machts, martinmachts, btw17
7	1381	danke, herzlichen, glueckwunsch, geburtstag, dank, vielen, erfolg, happy, birthday, liebe
8	1358	kirche, christen, kirchentag, religionsfreiheit, kirchen, ostern, frohe, wuensche, religion, ramadan
9	1355	sonntagsfrage, bundestagswahl, btw17, insabild, forsa, emnidbams, sternl, 36, programm, 38
10	1350	klimaschutz, klimawandel, klima, klimakrise, gruen, 2030, klimapolitik, kohleausstieg, klimaziele, darumgruen
11	1257	pflege, buergerversicherung, gesundheit, pflegekraefte, versorgung, weltgesundheitstag, hunderttausend, personal, bessere, patienten
12	1207	deniz, journalisten, free, pressefreiheit, yuecel, tuerkei, journalismus, haft, tagen, inhaftierten
13	1168	steuern, steuerkonzept, einkommen, steuersenkungen, panama, soli, entlasten, steuerpolitik, steuer, vermoegensteuer
14	1163	freiheit, zensur, menschenrechte, meinungsfreiheit, netzdg, gesellschaft, maas, grundrechte, sicherheit, offene
15	1150	twitter, tweet, tweets, twittern, retweeten, getwittert, lesen, twitteraccount, servicetweet, twittert
16	1115	kinder, kinderarmut, armut, kind, kinderrechte, kindergrundsicherung, kindern, kindertag, eltern, kinderehen
17	1024	migranten, einwanderungsgesetz, asyl, asylbewerber, migration, illegale, einwanderung, zuwanderung, migrantenkriminalitaet, asylmissbrauch
18	963	polizei, polizisten, hamburg, sicherheit, polizist, polizistinnen, bundespolizei, g20
19	960	bundestagswahl, wahlkreis, kandidaten, kandidatencheck, wdrkandidatencheck, btw17, kandidat, direktkandidat, kandidatinnen, direktkandidatin

Table B.4.: Top 10 representative words for each of 20 topics of the unique tweets with little preprocessing.

Topic	Count	Representative words
0	2262	terror, barcelona, london, terroristen, gedanken, opfer, manchester, islam, opfern, angehoerigen
1	2166	polizei, polizisten, hamburger, hamburg, polizist, nog, bundespolizei, polizeigewalt, sicherheit, hh
2	2076	pressefreiheit, journalisten, tuerkei, freedeniz, deniz, yuecel, haft, erdogan, tagen, journalist
3	2070	twitter, tweet, tweets, retweeten, instagram, account, twittern, selfie, thermilindner, nurmitgruen
4	2024	glueckwunsch, herzlichen, geburtstag, umwelt, birthday, happy, darumgruen, nrw, glueck, liebe
5	2024	sonntagsfrage, bundestagswahl, btw, bams, forsa, emnid, stern, insa, haustuerwahlkampf, dimap
6	2021	dieselgate, diesel, dobrindt, dieselgipfel, verkehrsminister, software, verbrennungsmotor, autokartell, waehlt, dieselskandal
7	1980	armut, kinderarmut, kinder, ungleichheit, grundeinkommen, kindergrundsicherung, kind, arm, einkommen, armutsbericht
8	1891	berlin, berliner, txl, tegel, berlins, mitte, btw, bundestag, agh, tegelschliessen
9	1715	klimaschutz, klimakrise, klimawandel, klima, darumgruen, kindergarten, arktis, merkel, planeten, klimapolitik
10	1684	migranten, einwanderungsgesetz, migration, einwanderung, illegale, zuwanderung, migrationshintergrund, flucht, braucht, brauchen
11	1561	danke, richtig, super, political, correctness, ganz, hinweis, dank, grossartig, seid
12	1525	italien, libyen, mittelmeer, kuestenwache, libysche, seenotrettung, eu, ngos, pdf, libyschen
13	1500	martin, schulz, martinschulz, sommerinterview, zukunftsplan, tvduell, emissionsfreie, auto, merkel, duell
14	1476	tvduell, tv, merkel, schulz, duell, martinmachts, politbarometer, spot, btw, faktencheck
15	1438	pflege, gesundheit, weltgesundheitstag, hunderttausend, pflegekraefte, patienten, versorgung, kompromiss, gesundheitspolitik, personal
16	1428	trump, syrien, nordkorea, paris, usa, venezuela, syrer, donald, pariser, agreement
17	1394	gerechtigkeit, soziale, sozialpolitik, forderungen, gruensozial, gruene, familienbudget, arbeitsmarkt, sozial, sozialer
18	1346	digitalisierung, digitale, bildung, tvduell, agenda, digital, digitalen, moderatoren, schulen, digitalebildung
19	1337	steuern, steuerkonzept, einkommen, steuer, entlasten, panamapapers, mrd, euro, steuersystem, steuersenkungen

Table B.5.: Top 10 representative words for each of 20 topics of the duplicate tweets with strong preprocessing.

References

Angelov, D. (2020). Top2Vec: Distributed Representations of Topics. *CoRR*, abs/2008.09470.

Arun, R., Suresh, V., Veni Madhavan, C. E., and Narasimha Murthy, M. N. (2010). On Finding the Natural Number of Topics with Latent Dirichlet Allocation: Some Observations. In Zaki, M. J., Yu, J. X., Ravindran, B., and Pudi, V., editors, *Advances in Knowledge Discovery and Data Mining*, pages 391–402. Springer, Berlin, Heidelberg.

Aßenmacher, M. (2021). *Comparability, evaluation and benchmarking of large pre-trained language models*. PhD thesis, Ludwig-Maximilians-Universität München.

Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *CoRR*, abs/1607.06450.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate.

Bansal, S. (2016). Beginners Guide to Topic Modeling in Python. `https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/#respond`. Accessed: 2022-01-29.

Barrie, C. and Ho, J. (2021). academictwitteR: an R package to access the Twitter Academic Research Product Track v2 API endpoint. *Journal of Open Source Software*, 6: 3272.

Belkin, M. and Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6): 1373–1396.

Blei, D. M. (2012). Probabilistic Topic Models. *Communications of the ACM*, 55(4): 77–84.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3: 993–1022.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. The Association for Computational Linguistics.

Campello, R. J. G. B., Moulavi, D., and Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In Pei, J., Tseng, V. S., Cao, L., Motoda, H., and Xu, G., editors, *Advances in Knowledge Discovery and Data Mining*, PAKDD 2013, pages 160–172. Springer, Berlin, Heidelberg.

Carbonell, J. and Goldstein, J. (1998). The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Carrigan, M., Debut, L., Gugger, S., Noyan, M., Saulnier, L., Tunstall, L., and von Werra, L. (n.d). How do Transformers work? https://huggingface.co/course/chapter1/4?fw=pt. Accessed: 2022-01-31.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078.

Deepset (2019). Open Sourcing German BERT Model. https://www.deepset.ai/german-bert. Accessed: 2022-02-15.

Der Bundeswahlleiter (2017). Wahl zum 19. deutschen Bundestag am 24. September 2017. Heft 3. https://www.bundeswahlleiter.de/dam/jcr/3f3d42ab-faef-4553-bdf8-ac089b7de86a/btw17_heft3.pdf. Accessed: 2022-02-14.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Dieck, T. (2008). *Algebraic Topology*. EMS textbooks in mathematics. European Mathematical Society.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, 14(2): 179–211.

Feng, J., Zhang, Z., Ding, C., Rao, Y., and Xie, H. (2020). Context Reinforced Neural Topic Modeling over Short Texts. *CoRR*, abs/2008.04545.

Goldberg, Y. (2015). A Primer on Neural Network Models for Natural Language Processing. *CoRR*, abs/1510.00726.

Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Griffiths, T. L., Tenenbaum, J. B., and Steyvers, M. (2007). Topics in semantic representation. *Psychological Review*, 114: 2007.

Grootendorst, M. (2020a). BERTopic: Leveraging BERT and c-TF-IDF to create easily interpretable topics.

Grootendorst, M. (2020b). Topic Modeling with BERT: Leveraging BERT and TF-IDF to create easily interpretable topics. https://towardsdatascience.com/topic-modeling-with-bert-779f7db187e6. Accessed: 2022-01-29.

Hatcher, A. (2000). *Algebraic topology*. Cambridge Univ. Press, Cambridge.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385.

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

Hong, L. and Davison, B. D. (2010). Empirical Study of Topic Modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, page 80–88, New York, NY, USA. Association for Computing Machinery.

Huilgol, P. (2020). Quick Introduction to Bag-of-Words (BoW) and TF-IDF for Creating Features from Text. https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/. Accessed: 2021-12-14.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016a). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016b). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Jänich, K. (2005). *Topologie*. Springer Berlin, Heidelberg.

Larochelle, H. and Lauly, S. (2012). A neural autoregressive topic model. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Le, Q. V. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *CoRR*, abs/1405.4053.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Marschall, S. (2018). *Das politische System Deutschlands*. UTB basics. UVK Verlag, München. 4. aktualisierte Auflage.

May, J. (1967). *Simplicial objects in algebraic topology*. Princeton, NJ: VanNostrand Co.

McInnes, L. (2018). How umap works. https://umap-learn.readthedocs.io/en/latest/how_umap_works.html. Accessed: 2022-01-31.

McInnes, L., Healy, J., and Astels, S. (2016). How HDBSCAN Works. https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html. Accessed: 2022-01-31.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction.

Mehrotra, R., Sanner, S., Buntine, W., and Xie, L. (2013). Improving LDA Topic Models for Microblogs via Tweet Pooling and Automatic Labeling. SIGIR '13, pages 889–892, New York, NY, USA. Association for Computing Machinery.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Müller, D. W. and Sawitzki, G. (1991). Excess Mass Estimates and Tests for Multimodality. *Journal of the American Statistical Association*, 86(415): 738–746.

National Institute of Neurological Disorders and Stroke (2002). Life and Death of a Neuron. https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Life-and-Death-Neuron. Accessed: 2022-01-07.

Nielsen, M. (2015). *Neural Networks and Deep Learning*. Determination Press.

Ostendorff, M., Blume, T., and Ostendorff, S. (2020). Towards an Open Platform for Legal Information. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, JCDL '20, page 385–388, New York, NY, USA. Association for Computing Machinery.

Pai, A. (2020). CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning. https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/#h2_1. Accessed: 2022-01-09.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Phi, M. (2018). Illustrated Guide to Recurrent Neural Networks. https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9. Accessed: 2022-01-18.

Phi, M. (2020). Illustrated guide to Transformers - Step by Step Explanation. https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0. Accessed: 2022-01-25.

Pilehvar, M. and Camacho-Collados, J. (2020). Embeddings in natural language processing: Theory and advances in vector representations of meaning. *Synthesis Lectures on Human Language Technologies*, 13(4): 1–175.

Pitts, W. and McCulloch, W. S. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5: 115–133.

Reimers, N. (2022). Sentence-Transformers. https://www.sbert.net/docs/pretrained_models.html#multi-lingual-models. Accessed: 2022-02-14.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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.

Ruder, S. (2019). *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland, Galway.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, 323(6088): 533–536.

Scheible, R., Thomczyk, F., Tippmann, P., Jaravine, V., and Boeker, M. (2020). GottBERT: a pure German Language Model. *CoRR*, abs/2012.02110.

Schmidt, J.-H. (2017). Twitter-Nutzung von Kandidierenden der Bundestagswahl 2017. Verbreitung, Aktivität und Informationsquellen. *Media Perspektiven*, (12): 616–629.

Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45: 2673–2681.

Stier, S., Bleier, A., Bonart, M., Mörsheim, F., Bohlouli, M., Nizhegorodov, M., Posch, L., Maier, J., Rothmund, T., and Staab, S. (2018). *Systematically Monitoring Social Media: the case of the German federal election 2017*, volume 2018/04 of *GESIS Papers*. GESIS - Leibniz-Institut für Sozialwissenschaften, Köln.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215.

Thurich, E. (2011). Pocket Politik. Demokratie in Deutschland. Bundeszentrale für politische Bildung. Überarbeitete Neuauflage Bonn.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *CoRR*, abs/1706.03762.

Weingärtner, J. (2021). Positive and negative campaigning on Twitter in multiparty systems. Bachelorarbeit (unveröffentlicht).

Weng, J., Lim, E.-P., Jiang, J., and He, Q. (2010). Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA. Association for Computing Machinery.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4): 339–356.

Williams, A., Nangia, N., and Bowman, S. R. (2018). A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 1112–1122. The Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2021). *Dive into Deep Learning*. https://d2l.ai.

Zhao, H., Phung, D., Huynh, V., Jin, Y., Du, L., and Buntine, W. L. (2021). Topic modelling meets deep neural networks: A Survey. *CoRR*, abs/2103.00498.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *The IEEE International Conference on Computer Vision (ICCV)*.

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.

Anne Gritto