Technische Hochschule Ingolstadt

Faculty of Computer Science
B.Sc. Artificial Intelligence

Bachelor's Thesis

Comparison of Deep Learning and Word Embedding Approaches for German Fake News Detection

Julian Willweber

First examiner: Prof. Dr. Beate Navarro Bullock
Second examiner: Prof. Dr. Sören Gröttrup
Issued on: 20.06.2023
Submitted on: 03.08.2023

Abstract

The ever-growing amount of fake news that is spread via social media and other digital platforms poses an increasingly dangerous risk for society. The mass of news has become too large to check for fake news manually. Therefore, automatic fake news detection systems have to be developed. Because most of the current research on fake news detection is done in English language, the goal of this thesis is to reach an overview of multiple approaches for German fake news detection.

The overview is gained by testing and comparing various machine learning algorithms, namely, BERT, DistilBERT and the Support Vector Machine combined with multiple word embedding algorithms. The algorithms were trained and tested on the FANG-COVID dataset. While the different models and word embedding algorithms achieved varying results in performance, all models were capable of achieving accuracy scores over 90%. This promises potential for their application and future research in German fake news detection.

Contents

List of Tables

List of Figures

Abbreviations

SVM Support Vector Machine

NLP Neural Language Processing

BERT Bidirectional Encoder Representations from Transformers

TF-IDF Term Frequency Inverse Document Frequency

TF Term Frequency

IDF Inverse Document Frequency

CBOW Continuous-Bag-Of-Words

LSTM Long-Term Short-Term

BiLSTM Bidirectional LSTM

RNN Recurrent Neural Network

MCC Matthews Correlation Coefficient

KNN K-Nearest Neighbour

RF Random Forest

CNN Convolutional-Neural-Network

GRU Gated Recurrent Unit

DT Decision Tree

LR Logistic Regression

1 Introduction

With social media platforms growing ever more significant and further reaching, these platforms' influence on society also rapidly increases. News and headlines are now spread predominantly through short tweets or posts on Instagram or Facebook instead of traditional newspapers. Although news can spread faster and reach more people through digital media than traditional newspapers [5], it has dangerous downsides. Not only qualified journalists and publishers spread information using these platforms, but also bad actors who try to manipulate broad masses of people. They spread misinformation, made-up stories, and reality-bending statements to create their own narrative and influence elections [6], for example.

A recent example showing the dangers of misinformation and fake news is the corona pandemic. Quickly after the pandemic began, the amount of fake news increased rapidly [7]. People started spreading misinformation over Facebook, Twitter, and other platforms, trying to make the world believe that the coronavirus doesn't exist or that the pandemic is just a scam. A few months later, as soon as discussion about a possible vaccine came up, the waves of fake news became even larger [8]. This led to the problem that many people started believing that the vaccine was terrible and, in turn, decided not to get vaccinated, risking their health. In their review paper Skafle *et al.* write:

> "... 19 studies ... implied that the misinformation spread on social media had a negative effect on vaccine hesitancy and uptake" [9, p. 1].

Detecting fake news has been an essential topic for big social media corporations for quite a while since they became aware of the problem and started to recognize their responsibility [10]. So far, German information was mostly left behind because these companies mainly focus on news and fake news written in English. Justus Mattern *et al.* [11] recognized this problem and created one of the first fake news datasets for German fake news detection. The dataset consists of about 41,000 news articles collected from all over the web and is used as the starting point for this thesis.

This thesis aims to test and compare various algorithms and models from Word Embedding, Machine Learning, and Deep Learning. Namely, these are the Support Vector Machine (SVM) in combination with CountVectorizer embeddings, Term Frequency Inverse Document Frequency (TF-IDF) embeddings and Word2Vec embeddings, a fully finetuned Bidirectional Encoder Representations from Transformers (BERT) model, a BERT model used as feature extractor only in combination with a classification head, and a DistilBERT model. The models are trained using various versions of the FANG-COVID dataset, which is preprocessed appropriately. Every model is tested and evaluated in various configurations to increase the performance of each model. Finally, the models and their performance results are compared and examined for advantages and disadvantages.

By the end of this thesis, a clear overview of the listed models and their performance in detecting German fake news should be established. This should be seen as a foundation and direction to build upon and advance the research in German fake news detection.

2 Theoretical Background

This section will review the main theoretical aspects and background knowledge for the models and algorithms used in section 5. First, it will go over the classic Machine Learning approaches, SVM, and the accompanying word-embedding algorithms CountVectorizer, TF-IDF, and Word2Vec before going deeper into BERT and DistilBERT.

2.1 Support Vector Machine

The SVM was introduced by Corinna Cortes and Vladimir Vapnik [12]. It is an algorithm that can be used for classification and regression problems. The idea behind the algorithm is that during training, so-called Support Vectors are used to fit a hyperplane through data points. This boundary is supposed to divide the various classes as well as possible. The SVM was chosen as naive algorithm because it has been shown to perform particularly well in text classification [13]. [13] has shown evidence that in comparison to other naive classification algorithms, the SVM performs better due to its suitability for text data properties, such as high dimensional input spaces and only few irrelevant features in the data, to name a few.

Briefly, the SVM functions as follows [14]. To fit the hyperplane in between the data, there are two types of kernels, linear kernels, and nonlinear kernels. Linear Kernels are used when the data points can be linearly separated. When drawing the boundary, the Support Vectors are used to fit the hyperplane between the data points, which best separate the classes. The Support Vectors are the data points that lie directly on the boundary and are the points closest to the hyperplane. And since they are the closest to the hyperplane, they are also the most critical data points for finding and fitting the most optimal hyperplane for the data. Their margin to the hyperplane is then maximized to find the boundary that separates the data points the furthest from each other. In other words, the SVM tries to maximize the margin width to find the hyperplane with the biggest margin to the support vectors of both classes. Since the data points might be harder to separate sometimes, there are two

types of margins: hard and soft. When the SVM works with hard margins, the data points are not allowed to cross the margin and can only be directly on the margin border. When calculating with soft margins instead, data points are allowed to lie within the margin to a certain degree, which has to be chosen beforehand. This enables the SVM to fit a hyperplane between the data points even when creating a strictly linear boundary is impossible. Additionally to linear kernels, there are also the already mentioned nonlinear kernels. These kernels can be used when dividing the data points using a linear kernel is not possible. This can be done by using the so-called Kernel Trick, where the data points are scaled up to a higher dimension, where they can then be separated and then scaled back to the original dimension. This way, nonlinear problems can be modeled, too[14].

2.2 CountVectorizer

The CountVectorizer is a technique for turning a corpus of documents into numerical representations. It can be found in the Python library scikit-learn[15]. The CountVectorizer counts the tokens within a document and then stores the values inside a matrix. This matrix can then be fed into a classification algorithm [16].

2.3 TF-IDF

Various methods for preprocessing the text were used as input to the SVM. The first one is TF-IDF [17, p. 117-119]. TF-IDF is a method in the field of Information Retrieval and is used to represent documents in a form that gives information about their content. The first component, the Term Frequency (TF), is a score that describes the frequency of a term t_i within a single document D. The more often t_i appears in a document, the higher the TF score. To normalize this score for terms that occur very often in a given document, the score of every t_i is divided by the total number of terms, $\sum x$, in a document [17].

$$tf(t_i, D) = \frac{f(t_i, D)}{\sum x} \tag{1}$$

The other part, Inverse Document Frequency (IDF), measures how important

and how much information a term t_i contains in the context of the entire corpus. Suppose t_i often occurs within a corpus of documents, D. In that case, the number of documents containing t_i, dft_i, is high and the probability that t_i is a generic word is very high, which in turn means that t_i contains little to no information regarding the subjects of the different documents. Therefore, t_i should be ranked very low, and the algorithm shouldn't pay much attention to that specific term, if any [17].

$$idf(t_i) = log\frac{N}{dft_i} \tag{2}$$

Finally, when both scores are multiplied, the result is the TF-IDF score. This score provides information about the relevance of a term inside a document in relation to its relevance inside other documents in the corpus. For example, if a term appears very rarely in the majority of documents in the corpus but very often inside a few selected documents, then this indicates that this specific term holds a lot of information about the topic of these particular documents. Also, this can be used as an indicator for the topic of a document. If there is one term that has a very high TF-IDF score, then this particular document is probably about a topic that is related to that specific term [17].

2.4 Word2Vec

Word2Vec was introduced in 2013 by Mikolov *et al.* [3] from the Google research team. Word2Vec is a method for representing words as vectors. The vector representation of a word has various advantages over working with the actual word. With Word2Vec, it is possible to model the relationship and context between two words. Mikolov *et al.* give the following example:

> "... vector("King") - vector("Man") + vector("Woman") results in a vector that is closest to the vector representation of the word Queen..." [3, p. 2], [18]

Behind the term Word2Vec, there are two versions of the algorithm. One is based on the Continuous-Bag-Of-Words (CBOW) model for learning the relationships between words, and the other uses a skip-gram model. Both variants are similar in the

way they work. The CBOW architecture tries to predict the current word based on the words that come before and after it [3]. In contrast, the skip-gram architecture takes the current word and tries to predict the context words around it. In Figure 1, the two architectures are displayed. On the left is the CBOW architecture and on the right the skip-gram architecture.

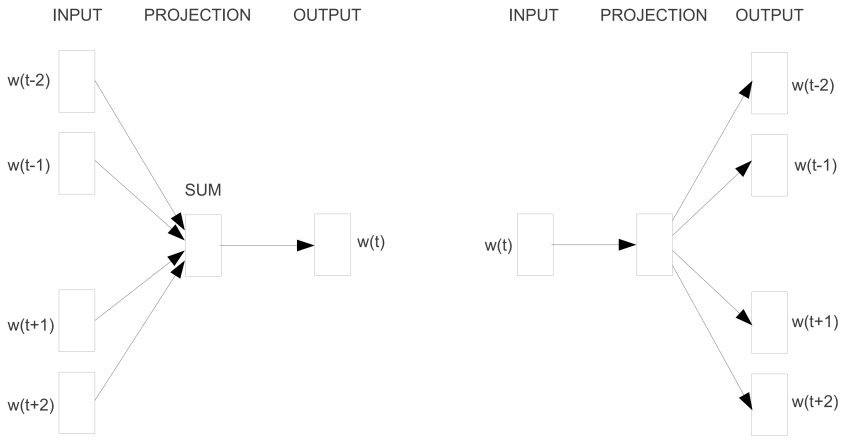

Figure 1: Visual comparison of the two Word2Vec architectures, CBOW on the left, skip-gram on the right. Source: [3, p. 5]

2.5 BERT

For the deep learning approach, BERT was chosen, which is a neural network. Neural networks are the basis for deep learning and have performed well in tasks like text classification [19].

2.5.1 Regular BERT

BERT is a language representation model proposed by Devlin *et al.* [1]. BERT was published in 2018 and since then has been applied to various use cases. Before going into BERT, one should take a look at the architecture of the transformer. The transformer architecture was proposed by Vaswani *et al.* [20] in 2017. In their paper "Attention Is All You Need," they introduce a neural language model architecture consisting of encoder and decoder layers. Encoder layers take input data and encode it to extract information. Decoder layers then take the encoded representation of

Model	#Hidden Layers	Hidden Size	#Self-Atten. Heads	#Params
$BERT_{BASE}$	12	768	12	110M
$BERT_{LARGE}$	24	1024	16	340M

Table 1: Comparison of the architectures of the different model versions of BERT. Source: [1, p. 4173]

the data and decode it again. These layers take advantage of a concept called multi-head (self-) attention. Attention can be explained as a mechanism that is used "to draw global dependencies between input and output" [20, p. 2]. In short, multi-head attention is a special version of the attention mechanism. This function takes a query and a key-value pair in the form of vectors as input and maps these to an output, which is also a vector. The key value pairs can be described as "memory keys and values" [20, p. 5] that "come from the output of the encoder" [20, p. 5]. Vaswani *et al.* describe the attention mechanism's output as follows:

> "...weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key." [20, p. 3]

In multi-head attention, this concept is expanded to be used on multi-dimensional inputs. Instead of going over one input only, it goes over multiple inputs in parallel, making it possible to capture more information from the input. The resulting outputs of the single attention heads are then concatenated into one output, which in turn is rescaled to the correct dimension, representing the final output.

Multi-head attention is used in both the encoder and decoder layers. The architecture of BERT is based on the architecture of the Transformer, except that it only uses encoder layers and doesn't rely on any decoder layers. Devlin *et al.* [1] publish two versions of BERT, one called $BERT_{BASE}$ and $BERT_{LARGE}$[1]. The difference between the two is mainly in the architecture and model size.

[1] https://github.com/google-research/bert

2.5.2 Training

The training of BERT consists of two different stages. The first stage is the so-called pre-training. During pre-training, BERT is trained on two different tasks. Devlin *et al.* call the first task "Masked LM" [1, p. 4174]. During this task, a percentage of tokens in the input are masked, and the model tries to predict these masked tokens. The second task is Next Sentence Prediction. During this task, the model is presented with two sentences, and the model is supposed to decide if the second sentence is the actual sentence following the first one or if it is not, based on the context within the sentences [1].

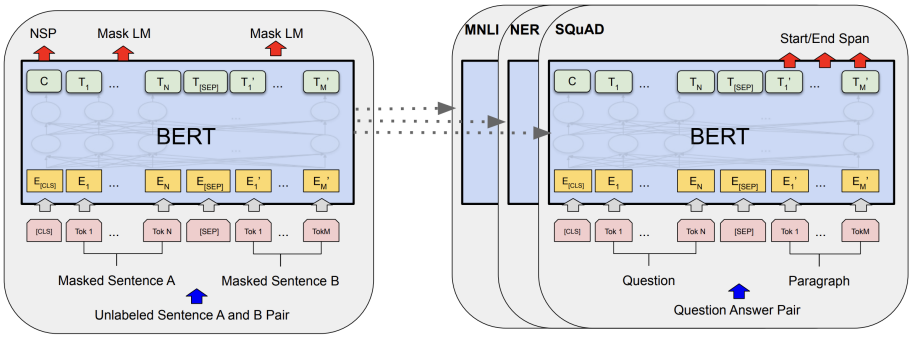

Figure 2: On the left, the raw pre-trained model that can be fine-tuned into one of the models on the right. Source: [1, p. 4173]

After pre-training, the result is a model which can then be fine-tuned on various different tasks. This process is called "transfer learning" [1, p. 4173]. Examples of possible fine-tuning tasks are:

- Text Classification [21]
- Sentiment Analysis [22]
- Named Entity Recognition [23]

Finetuning a BERT model is done by taking the pre-trained model and adding specific layers at the end of the model depending on what tasks to finetune the model on. The task of finetuning BERT is visualized in Figure 2 [1]. Depending on how one wants to finetune BERT, there are multiple ways to do that. The BERT model

can be kept frozen, i.e. the layers of the initial model are not trained, and only the task-specific layer is finetuned. Another option is that the full model is unfrozen, and every layer gets updated during training. The last option is the in-between: only selected layers of the BERT model are unfrozen and updated during training [24].

2.5.3 Tokenizer

Another central part of BERT besides the model itself is the tokenizer. The tokenizer uses WordPiece tokenization to preprocess the input [1, p.4174]. It is responsible for turning the input text into a representation BERT can work with and understand. Figure 3 shows the different levels on which the tokenizer works. The final input to the model consists of three different embeddings, the token embeddings, the segment embeddings, and the position embeddings, which are all summed together [1].

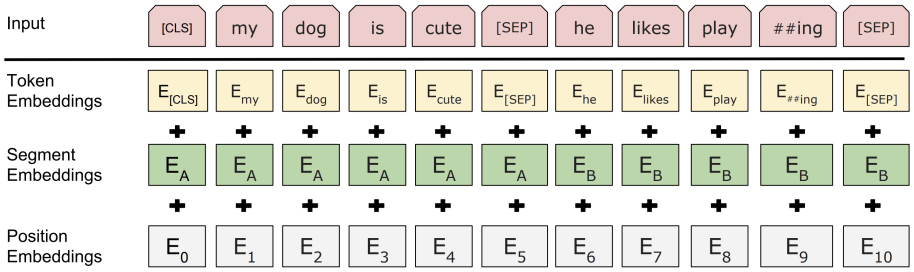

Figure 3: The steps the tokenizer takes to encode the input for the BERT model. Source: [1, p. 4175]

2.5.4 DistilBERT

Sanh *et al.* [2] used the technique called knowledge distillation and created a smaller version of BERT. In short, knowledge distillation can be described as

> "a compression technique in which a compact model ... is trained to reproduce the behavior of a larger model ... or an ensemble of models." [2, p. 2].

For the BERT model, Sanh *et al.* decided to shrink the distilled model and lower the number of layers in the model compared to the original BERT model. The resulting

model is significantly smaller and has comparable performance in accuracy to the original BERT model, all while being faster at calculating the prediction.

Model	#Layers	#Params. (Mio.)	Acc. (IMDb)
$BERT_{BASE}$	12	110	93.46
DistilBERT	6	66	92.82

Table 2: Comparison of BERT and DistilBERT. Source: [2, p. 2-3]

Table 2 compares some key aspects between BERT and DistilBERT. Mainly the layers are reduced to only half of the initial BERT model. With this reduction in layers, the parameters are also reduced from 110 million down to 66 million. Because of this difference in size and parameter count, DistilBERT is faster to train than BERT while keeping a comparable accuracy score.

2.6 Metrics

To evaluate the models and gain insight into their performance, evaluation metrics have to be used. The following section will introduce the evaluation metrics used.

2.6.1 Accuracy

The first metric is the accuracy score. It is calculated by counting the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

In other words, the accuracy score is a metric that gives a score about how many available examples were classified correctly. The accuracy score of a classifier lies between 0 and 1.

2.6.2 F1

The F1-score is another measure to calculate the accuracy of a binary classification task. It is the harmonic mean of the precision and recall of a classification.

$$Precision = \frac{TP}{TP + FP} \quad (4) \qquad Recall = \frac{TP}{TP + FN} \quad (5)$$

Using the results from (4) and (5), the final F1-score is then calculated as follows:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (6)$$

The F1-score of a classifier can lie between 0 and 1.

2.6.3 Matthews Correlation Coefficient

Matthews Correlation Coefficient (MCC) is another metric.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (7)$$

The MCC of a classifier lies between -1 and 1, where -1 means that the predicted class is always wrong, 0 means that the prediction is random, and 1 means that the classifier classifies everything correctly.

3 Fake News

The following section introduces the term "fake news" and gives insight into the different types of fake news and forms of media it can appear in. After that, it will go over the problems of the spread of fake news and finally give an overview of the current state-of-the-art in fake news detection.

3.1 Definition

Before diving deeper into the recognition of Fake News, it's important to distinguish between the different types of false news. While some would classify every news article that's factually off as Fake News, a more nuanced distinction should be made. First, it is important to consider the terms misinformation, disinformation, and malinformation. Wardle and Derakhshan [25, p. 20] divide fake news into three categories:

Misinformation: Misinformation describes the spread of wrong information without any ill intentions behind sharing the news. The one sharing it believed it to be true before sharing.

Disinformation: Disinformation describes the spread of wrong information with bad intentions behind the sharing. The person sharing it knows the information to be false but spreads it anyway for financial gain, power, or manipulation of others.

Malinformation: Malinformation describes the spread of information whose roots are based on facts but is altered or manipulated to support a false narrative.

The listed categories can come in various forms of publication, and according to Collins *et al.* [26] and Medeiros and Braga [27], they can be further divided into the following types:

Clickbait: Clickbait is a form of a news article that exaggerates the news and uses attention-grabbing headlines to make people click on the article, which generates income for the publisher [26].

Propaganda: Propaganda can be defined as false or incorrect articles that are used to make a large group of people believe a certain story or lie that is created by political entities [26] [27].

Satire & Parody: Satire and Parody often exaggerate or reframe recent news in a humorous manner [26]. This is done using irony or sarcasm, which can be misunderstood or misinterpreted.

Hoax: A hoax can be defined as fake news that is deliberately spread to deceive the public, often aimed at an individual that is supposed to be damaged [26].

Conspiracy Theory: A conspiracy theory is a story created to explain events or the actions of usually powerful entities, blaming a conspiracy without any proof for the story or information provided in it [27].

Rumor: A rumor is news that is not clearly confirmed but is still spread on social media sites [27].

Biased News: Biased news is often news that leans towards a political direction and has a biased view on current events or people [27].

Another aspect of fake news is the different types of media it can be packaged in. Not only can written text be fake news but also the following:

- Fake News in the form of images (Generated images, photos taken out of context)
- Fake News in the form of videos (Deep Fakes, AI-generated videos.)
- Fake News in the form of written content
- Fake News in the form of audio content (Propaganda podcasts, ill-intentioned broadcasts)

Especially generated images have become increasingly dangerous over the past year. Tools like Stable Diffusion and DALL-E 2 have made AI image generation widely available to the general public with few restrictions. An example of how easily the generated images deceive people is the example of the pope wearing a

designer down jacket instead of his usual robe. Even though one could unmask the image as fake by taking a closer look at it and its details, many people were fooled by it and believed the image to be true [28].

Another recent development is the increasingly easier generation of Deep Fake videos. With the help of AI, one person's face can be taken and projected onto another person's face in a video [29]. This happens in a way that even the emotions, mimic, and voice is transferred to the fake video. This can be used for manipulation as well since anyone who knows how to create these videos can create fake videos of almost anybody else without their consent.

3.2 Problems

The past has repeatedly shown that fake news when left alone and unaddressed, can cause a lot of trouble and should not be ignored. During events that are out of the ordinary, fake news is often used to create a certain narrative around the event. This is done to manipulate unassuming people into believing something that might not be true but suits a certain agenda or influences their actions, i.e., voting for a certain candidate over another, refusing to get vaccinated, etc. All these influences can cause great harm to the general public. If enough people are successfully influenced to vote for a particular party, this may change the result of important elections, changing the political ruling for the coming years [6].

Another example is the corona crisis. Through the influence of Telegram groups and various other social networks, false rumors about the vaccines were spread. This caused a significant part of society to decide not to get vaccinated [9]. These two examples demonstrate why taking care of fake news is so important and not to ignore them or let them run its course.

Identifying fake news with the bare eye is a challenge in itself. Fake news often looks like reliable news from reputable sources but, in reality, falls into the category of disinformation or malinformation. For the unknowing reader, this can become a problem. Without warning that the consumed news is false, one can easily fall into the trap of believing and spreading them even further.

3.3 State-Of-The-Art

In the following section, a look will be taken at the state-of-the-art in fake news detection. First, a method using BERT proposed by Mattern *et al.* [11] on a German dataset will be discussed. Afterward, the section will go into English fake news detection, introducing some of the most promising methods for detecting fake news according to Villela *et al.* [30].

3.3.1 BERT

Mattern *et al.* test the performance of BERT on the FANG-COVID dataset [11]. More information on the dataset in section 4. They take a pre-trained model and unfreeze 6 of the 12 layers. These layers are then finetuned on the classification task of the dataset. Mattern *et al.* utilize only the first 512 tokens due to the limitation of BERT. More on this limitation in section 5.

Mattern *et al.* achieve state-of-the-art results on the dataset, reaching an accuracy of 0.981, precision of 0.966, recall of 0.976, and F1 score of 0.966.

3.3.2 Stacking Method

The Stacking method, proposed by Jiang *et al.* [4], is a multi-model approach that uses multiple algorithms from machine learning and deep learning to generate the prediction. The following machine learning algorithms are used: Logistic Regression (LR), Decision Tree (DT), K-Nearest Neighbour (KNN), Random Forest (RF), and SVM. And the following deep learning algorithms are used: Convolutional-Neural-Network (CNN), Long-Term Short-Term (LSTM), and Gated Recurrent Unit (GRU).

These various algorithms and models then receive inputs created with different embedding algorithms. The DT model, RF1, KNN, LR, and SVM receive input embedded with TF-IDF. The deep learning models, CNN, LSTM, and GRU, receive GloVe embeddings as input. And another Random Forest Model, RF2, receives TF embeddings as input. The predictions of all these individual models are passed on to another RF Model, RF3, which makes the final prediction using all previous

predictions.

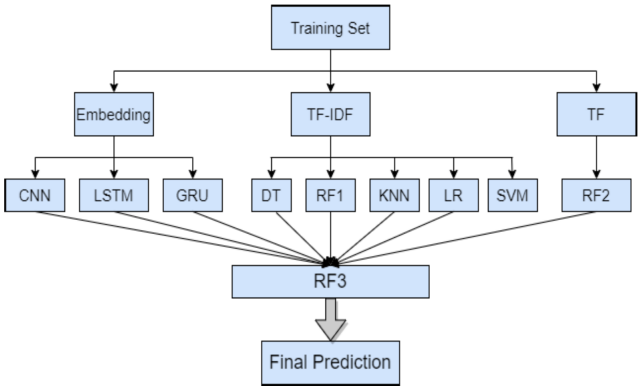

Figure 4: The stacking architecture proposed by Jiang *et al.*. Source: [4, p. 22634]

To evaluate the performance of their model, Jiang *et al.* use the ISOT [31], [32] dataset, and the KDnugget dataset[2]. The ISOT dataset consists of about 45.000 news articles labeled as "fake" or "real." Of these articles, 23.400 are fake news articles, roughly half of the dataset.

Performance-wise, the proposed stacking model reaches an accuracy of 99.94% and precision, recall, and f1 score of 100% each on the ISOT dataset. On the KDnugget dataset, the model achieves an accuracy of 96.05%, precision of 97%, recall of 96%, and f1 score of 96%, too.

3.3.3 Bidirectional LSTM

Jiang *et al.* [33] propose a Bidirectional LSTM (BiLSTM) model for detecting fake news. In contrast to traditional LSTM models, BiLSTM models consist of regular LSTM models and, therefore, can process data from two directions. When processing a sentence, for example, the BiLSTM can process the sentence starting from the beginning and end simultaneously. This enables the model to capture information and details a normal LSTM could not.

The BiLSTM model was trained on the ISOT dataset. Jiang *et al.* use 80% of the dataset for training the model and for testing the remaining 20%. The BiLSTM model has an accuracy score of 99.82%

[2]https://www.kdnuggets.com/2017/04/machine-learning-fake-news-accuracy.html

4 Dataset

This section introduces the FANG-COVID dataset by Justus Mattern *et al.* [11]. The dataset was released in 2021 and is one of the first datasets for German fake news. The following section will go over the dataset and give insight into its content.

4.1 Structure

The dataset consists of about 41.242 samples stored inside `.json` files. Every `.json` file consists of the following features:

`url`	The article's URL
`date`	Date and time when the article was published
`source`	Source website from where the article was taken
`label`	Label, which tells you if the article is fake or real
`header`	The heading of the article
`article`	Contains the full-text news article
`twitter-history`	The article's social context from Twitter

Table 3: Listing of all different features of one dataset and the data contained in them.

For the work in this thesis, mainly the label and article were important. The remaining parts of the dataset were not used. The label tells if a given article is fake or real news and is kept binary, meaning it tells only if an article is fake or real, not in between, like half-correct, for example. The field article contains the full-text news article without any preprocessing, containing tabs, special characters, etc. Finally, the Twitter history contains information about tweets that were posted on Twitter and reference the given article. It contains information about statistics like likes, retweets, replies, etc. Additionally, it stores information about the poster of every tweet, like followers, friends, number of tweets posted, etc.

4.2 Statistics and Visualizations

Figure 5 visualizes the distribution of real and fake news articles in the dataset. The dataset contains about double the number of real news articles compared to fake news articles, which results in a ratio of $\frac{2}{3}$ real to $\frac{1}{3}$ fake news articles.

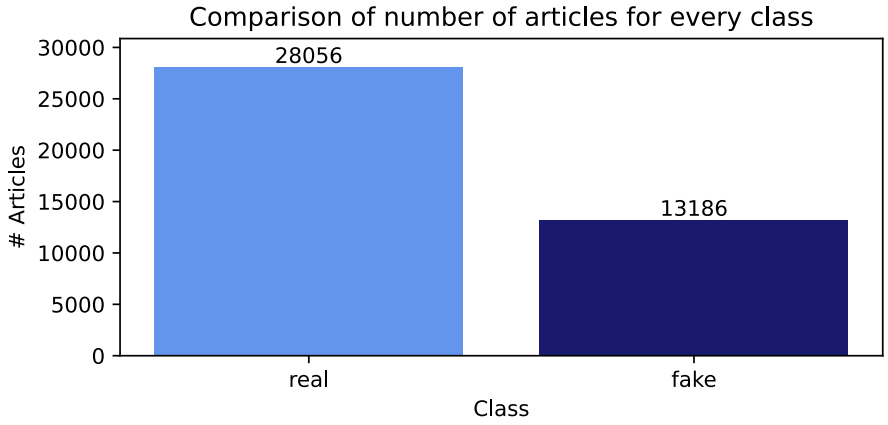

Figure 5: Comparison of the number of articles for every class.

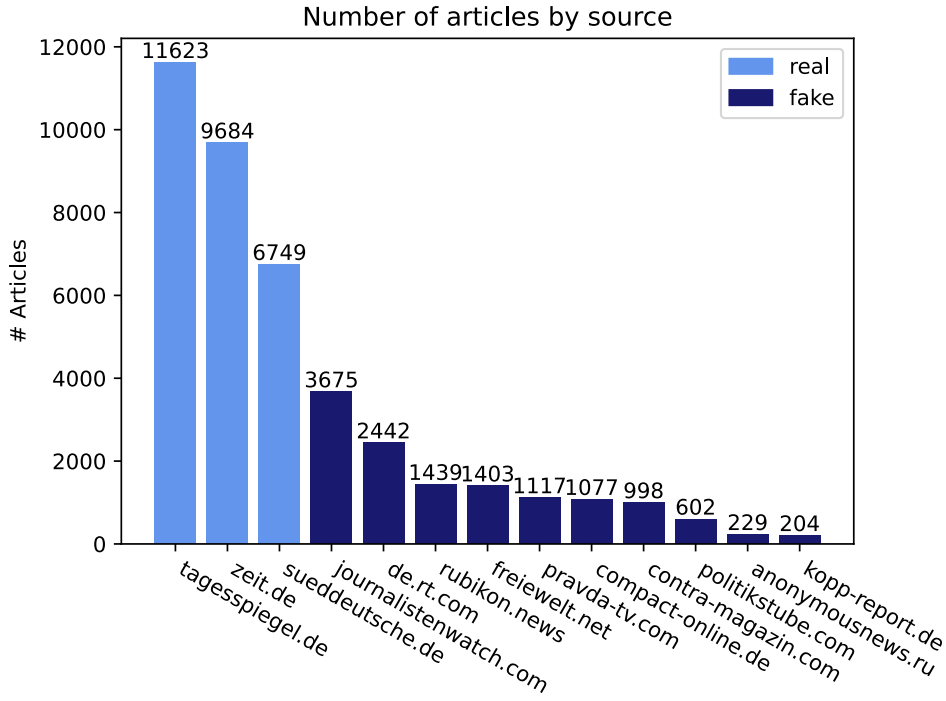

Figure 6: The number of articles by source. Colored by fake and real.

Two types of sources are used in the dataset. There are real news and fake news sources. The real news sources, marked in light blue in Figure 6, are tagesspiegel.de, zeit.de, and sueddeutsche.de. The fake news sources, marked in dark blue in Figure 6, are journalistenwatch.com, de.rt.com, rubikon.news, pravda-tv.com, compact-online.de, contra-magazin.com, politikstube.com, anonymousnews.ru, and kopp-report.de. The real news sources were selected based on the rating of media experts and the fake news sources were selected through the use of fact-checking orgnizations that classified the selected fake news sources as unreliable [11, p. 80]. When looking at the number of published articles per source, the real news sources have a higher output than fake news sources. With 11,623 articles, tagesspiegel.de has the highest output of articles about the coronavirus. When comparing these numbers to the numbers of the fake news sources, one can see that the average output of these sources is much lower than that of real news sources.

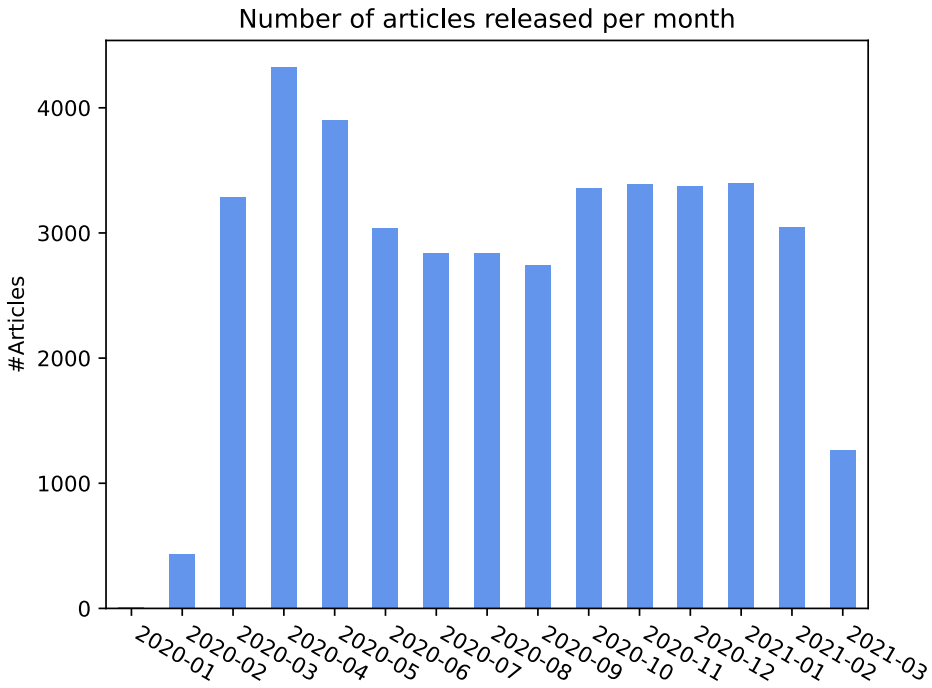

Figure 7: Number of articles released every month.

In Figure 7, one can see the span of time from which the articles were taken. The earliest articles were published in January 2020, while the latest articles were

published in March 2021. When looking at the distribution of articles over time, a few events can be observed. First, one can see the rise in the number of articles around the coronavirus outbreak in March 2020. The peak in the number of articles is in April 2020, the first full month of articles after the outbreak in mid-March. Lastly, during the summer 2020, there is a dip in published articles, and right before the winter months, a rise in the number of articles can be observed.

4.3 GermanFakeNC Dataset

The fake news dataset GermanFakeNC was published by Inna Vogel and Peter Jiang [34]. The dataset was, according to Vogel and Jiang, the first German dataset of its kind. The articles in the dataset were collected from December 2015 to March 2018. It was fact-checked on a claim basis, meaning every claim was fact-checked and received a label, not only the entire article. The articles received two scores, and the first one was an overall score ranging from 0.1 to 1, with 0.1 meaning the article is mostly true, while a rating of 1 indicates fake news. The second rating is described by Vogel and Jiang as a ratio that gives a score over how big the proportion of fake claims is in relation to the full article. This ratio ranges from 1 to 9, with 1 meaning an article contains up to 25% false claims, 2 meaning that 50% of an article is incorrect, 4 meaning the full article is incorrect, and 9 meaning the article didn't contain any claim. The corpus contains 4500 reliable news articles and 490 fake news articles. Out of these 490, 384 received a ratio rating of 1 or 2, 82 received a rating of 3, and 20 articles received to score 4 [34]. The reliable news articles were collected from Frankfurter Allgemeine Zeitung and Süddeutsche Zeitung, while the fake news articles were collected from websites like www.allesroger.at and www.compact-online.de.

5 Experimental Setup

The following section will give insight into the versions of the used dataset and the types of preprocessing applied to them. Afterward, the training processes of the models will be explained in detail.

5.1 Datasets

Before the dataset was used in any way, it was first brought into the correct form. Initial *.json* files included in the dataset after download were converted and combined into one *.csv* file, which contains the full fang-covid dataset. The file can be found under *2_fang-full-raw.csv*.

In the next step, the "labels" column, which consisted of "fake" and "real", was encoded into a binary representation. This was done to match the requirements of the models, that the labels have to be in numeric form. The file can be found under *3_fang-full-raw_labels-binary.csv*.

To streamline the process of training models, irrelevant columns were removed from the dataset, so only the article and label columns were left. These columns were renamed to "text" and "label_pred". The file can be found under *4_fang-articles_raw.csv*.

The next step of data preprocessing was the sanitation of the articles column. As Mattern *et al.* stated, some articles contain indicators that the articles are fake or real, which creates bias in the data [11, p. 89]. To reduce this bias, steps similar to those of Mattern *et al.* [11, p. 89] and additional other steps were taken to clean up the data. The steps were the following:

- The phrase "*Bleiben Sie aufmerksam!*" was removed
- The tags "*rt/dpa*" and "*ns/sna/tm*" were removed
- Tags like "*(dpa)*" at the end of articles were removed
- The phrase "*von [Name of author]*" was removed
- "*...*" was removed as well

- The token "COMPACT" was removed
- The phrase "Hier bestellen" was removed
- Often repeating paragraphs like "*Illuminatenblut: Die okkulten Rituale...*", "*Wenn Sie mehr über die heimlichen Machenschaften...*", etc. were removed.

A full list of paragraphs that were removed from the dataset can be found in the appendix A.1. These clean-up operations were made using the regex library. Cases that could not be removed in an automated manner were treated manually using the replace all function of VS Code. The file containing this dataset can be found under *5_fang-articles_sanitized.csv*.

The two datasets *4_fang-articles_raw.csv* and *5_fang-articles_sanitized.csv* were then further preprocessed using the following steps:

Lowering. All characters in the datasets were lowered. This reduced the variance for words that appear in upper and lowercase.

Removal of Special Characters. Every character that is not a letter or a number was removed from the dataset.

Tokenization. All articles of the dataset were split up into single tokens. This was done at every space character.

Stopword Removal. All stopwords from the German language were removed using the stopword list from the library SpaCy[3]. This was done to remove the most common words of the German language, which usually have no relevant meaning for a document, and thus could be removed without losing the document's meaning.

Lemmatization. All tokens were lemmatized to increase the probability of words and not have their probabilities split up among multiple forms of the same word. The resulting datasets can be found in the files *6_fang-articles_tok-lem_raw.csv* and *7_fang-articles_tok-lem_sanitized.csv*.

The total length of the dataset was not changed throughout the preprocessing and stayed the same as in the initial dataset.

[3] https://spacy.io/

5.2 SVM

The first model is the naive model, using the SVM algorithm. The SVM was trained with input from embedding algorithms. For the training, the data had to be pre-processed and cleaned. Therefore, the datasets *6_fang-articles_tok-lem_raw.csv* and *7_fang-articles_tok-lem_sanitized.csv* were used. The datasets were split into a training split containing 80% of the dataset and a test split containing the other 20%.

Parameter	**Value**
C	0.1, 1, 10
Kernel	linear, rbf, sigmoid, poly
gamma	scale, auto
Vector Length	300, 400
W2V Arch.	CBOW, Skip-Gram

Table 4: Hyperparameters used for finetuning the SVM models.

For finetuning the SVM, various hyperparameter combinations were tested. The search space for the hyperparameters can be seen in Table 4. The search pattern used for combining the parameters was a mix of random search and grid search. Before there was any knowledge about the model's performance on the dataset, the parameters were combined randomly. After gathering some information about what worked and what didn't, the search became more grid-search-oriented.

The word embedding algorithms CountVectorizer and TF-IDF kept the same configuration throughout the testing. The Word2Vec algorithm, however, was tested in various configurations. The word vector length was tested with lengths of 300 and 400. Additionally, both architectures of the Word2Vec algorithm, CBOW and Skip-Gram, were tested.

5.3 BERT

The following section will go over all aspects of the finetuning process of the BERT and DistilBERT models, providing information about the training environment, model configurations, and the training process. Afterward, the text-splitting procedure is explained in detail.

5.3.1 Paperspace

Both the BERT and DistilBERT model were trained on cloud GPUs provided by *Paperspace*[4]. To begin the training on the Paperspace cloud, a plan has to be selected. For this thesis, the free-tier plan was chosen. In the next step, one has to create a VM instance. Here one can choose from multiple different VM operating systems. For this thesis, the ML-in-a-Box environment was chosen, which is a Ubuntu 20.04 installation with many machine learning tools already pre-installed. The GPU used was model Nvidia Quadro M4000 which has 8GB of video RAM. The cloud GPU can be accessed via ssh from the terminal and runs Ubuntu 20.04 in the ML-in-a-Box environment with a disk size of 50GB. For training the models, the free-tier plan was used, which limits the GPU usage to 6 consecutive hours, after which the machine and the training have to be restarted. The pricing for the entry-level Nvidia GPU is 0.45$ per hour. On this GPU model, BERT takes about 2 hours for one epoch, which equals about 0.90$ per epoch. Because of the limit of 6 hours of consecutive training and an average training time of over 2 hours, the maximum number of epochs tested was two.

Training the models locally was not possible due to incompatibilities between MacOS and the PyTroch library. Also, training the models on the Google Cloud Platform was not an option since the cloud GPUs provided by Google are not easily accessible for private/average customers without special access plans.

[4] https://www.paperspace.com/

5.3.2 Implementation

The training script is based on the implementation by Chris McCormick and Nick Ryan [35]. The pre-trained BERT model is taken from the HuggingFace transformers library [36]. The models contained in this library are already pre-trained, implemented in Pytorch, and ready for finetuning. For this thesis, the pre-trained BERT model "*bert-base-german-cased*"[5] was used. For fine-tuning, the model type has to be selected depending on what the finetuning task should be. For the case of classification, the finetuning head is selected.

Two different model configurations were chosen for the training. The first configuration is the full unfrozen BERT model. In this model, every layer is unfrozen and gets updated during the training. The other configuration is the frozen BERT model. With this configuration, only the classification head that is added on top of the BERT model gets trained and receives updates.

The classification head is added via *BertForSequenceClassification()* for the BERT model and the tokenizer is initialized via *BertTokenizer()*.

5.3.3 Training

For training the BERT model, the datasets *4_fang-articles_raw.csv* and *5_fang-articles_sanitized.csv* were used. The unprocessed datasets, without any tokenization, etc., were chosen to keep the data as close to the original training data as possible and to give the model "the maximal document context provided by the data" [1, p. 4179]. During training, the datasets were fed directly into the tokenizer, and the encoded inputs were then passed on directly to the model. The datasets were split into two parts, the train, and the test split. The train split was made of 80% of the initial dataset, and the other 20% went into the test split.

When training the full BERT model, the only possible batch size was 8. The reason for this was the VRAM of the GPU, which reached its limit with batches of 8. Any undertaking to increase the batch size beyond 8 resulted in an error during runtime, causing the system to run out of memory.

[5] https://huggingface.co/bert-base-german-cased

During the training, various combinations of hyperparameters were tested. The values of these hyperparameters were mainly chosen according to the recommendations of Devlin *et al.* [1, p. 4183-4184].

Parameter	**Value**
Epochs	1, 2, 3, 4
Batch sizes	8, 16, 32
Learning rates	$2e^{-5}$, $3e^{-5}$, $5e^{-5}$

Table 5: The different values and hyperparameters that were combined and tested.

5.3.4 Split Texts

Since the input for the tokenizer is limited to 512 tokens, which includes split words, i.e. "playing" is split up into "play" and "##ing", which counts as two tokens, everything that is left after these 512 tokens are thrown away and not factored into training.

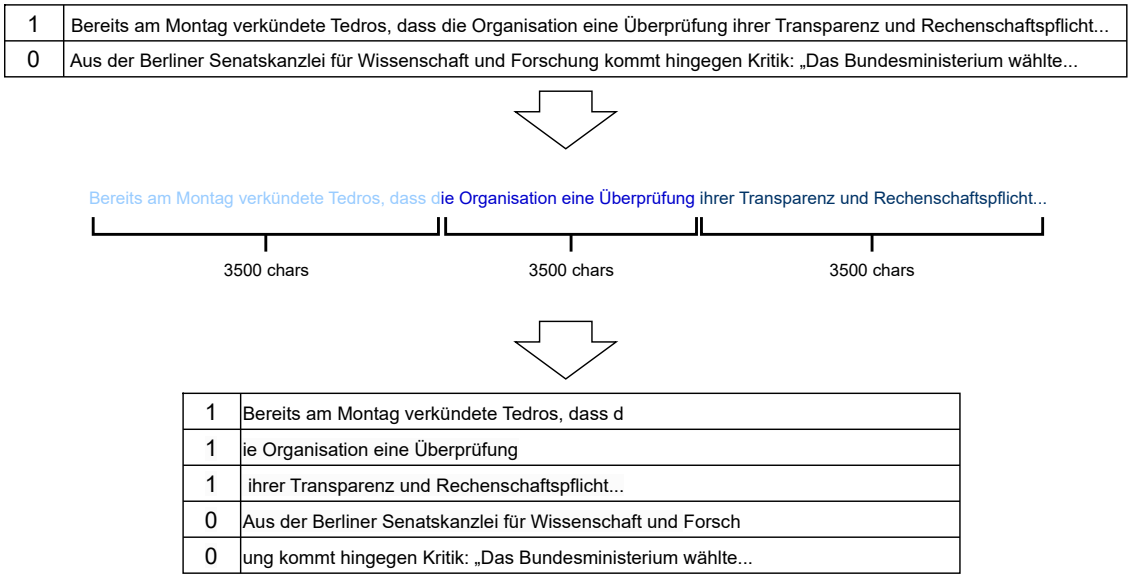

Figure 8: A visualization of the algorithm used to split the articles.

To not leave the rest of the dataset unused, every article that was longer than 3600 chars got split up after 3600 chars and separated into parts. These parts were

then stored inside a new dataframe and labeled with the same label as the initial article. To determine the length of the splits, first, the articles were separated every 500 space chars (" "). Then, the char length for each of these splits was calculated and averaged. This average lay at about 3600 chars. Every article was split after 3600 chars and the resulting parts of the articles were then added back to the corpus without any relation to one another.

While splitting the articles, some stayed as one because they didn't exceed the length limit of 3500 chars. The articles which exceeded the limit got split up and created more data points of their respective class. For example, a fake news article split into 4 parts creates 4 fake news data points. This changed the distribution between real news and fake news articles.

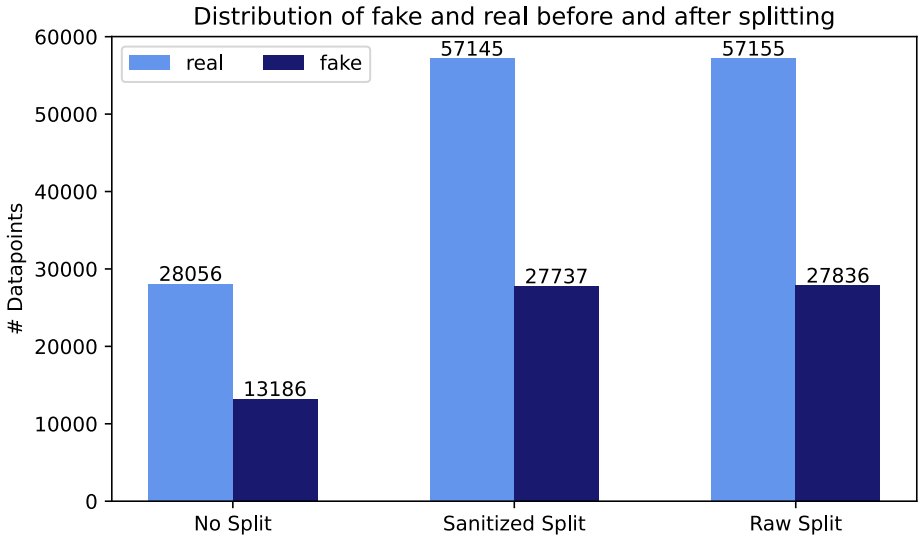

Figure 9: The distribution of real and fake data points in the datasets before and after splitting the articles.

5.3.5 DistilBERT

To compare the performance of the BERT base model against a similar, more efficient model, DistilBERT was chosen. The reasons for choosing DistilBERT were:

- DistilBERT performs comparably to BERT while being more efficient regarding training time.

- Because of the previously mentioned points, DistilBERT is a lot cheaper to train since the training duration is shorter and not as much computational power, in general, is needed, potentially reducing the cost for GPUs and training time.

- A model that was pre-trained on a German dataset is available on the Huggingface model hub[6], which makes it easy to train using the same routine as for the regular BERT model.

The training for the DistilBERT model is almost the same as for the BERT model. Instead of using the BERT classification head and tokenizer, the special classification head for DistilBERT has to be called via *DistilBertForSequenceClassification()* and the *AutoTokenizer()*.

To make the training of DistilBERT as comparable as possible to the training of BERT, the same datasets as for the BERT training, *4_fang-articles_raw.csv* and *5_fang-articles_sanitized.csv*, were used. The datasets were split in 80% for the training set and 20% for the test split, too.

5.4 Technical Details

To summarize the programming-specific details: For the training of BERT and DistilBERT the transformers [36] library by HuggingFace[7] and the library PyTorch [37] were used. The implementation for the SVM, CountVetorizer and TF-IDF the implementations from scikit-learn [15] were chosen. The Word2Vec algorithm was taken from Gensim [38]. The stop word list for the stop word removal preprocessing step was taken from SpaCy[8]. For general data manipulation Pandas [39] and Numpy [40] were used.

The code for replication of this thesis can be found on GitHub[9].

[6] https://huggingface.co/distilbert-base-german-cased
[7] https://huggingface.co/
[8] https://spacy.io/
[9] https://github.com/JulianWillweber/bt-german-fake-news-detection

6 Results

After having gone over the experimental setup in the previous section, this section will review the performance of the different models and highlight different aspects. A detailed look will be taken at the different configurations and their effects on the performance, the various performance measures, and the training durations of the models.

6.1 SVM with CountVectorizer

Table 6 shows the results of SVM training in combination with the CountVectorizer embeddings. The highest performance was reached by the SVM configured with a C value of 10, the rbf kernel and the gamma value set to "scale." It had a training duration of 50 minutes and achieved an accuracy score of 93.03, a F1 score of 0.89 and a MCC score of 0.84. The SVMs trained with the linear kernel and varying C values all reach about the same performance with an accuracy score of about 92, an F1 score of about 0.88 and an MCC of about 0.82.

dataset	C	kernel	gamma	runtime	Acc	F1	MCC
S	0.1	linear	scale	28m	92.30	0.88	0.82
S	1.0	linear	scale	28m	91.88	0.87	0.81
S	10	linear	scale	33m	92.45	0.88	0.83
S	1.0	rbf	scale	27m	92.57	0.88	0.83
S	10	rbf	scale	50m	**93.03**	**0.89**	**0.84**
S	1	rbf	auto	18m	68.40	0.02	0.09
S	1	poly	scale	58m	73.09	0.31	0.32
S	1	poly	auto	15m	68.03	0.00	0.00
R	10	rbf	scale	49m	94.04	0.90	0.86

Table 6: The performance of the SVM using the CountVectorizer embedding algorithm.

While the rbf kernel reached the highest performance when combined with the gamma value set to "auto", the performance dropped and the model reached an accuracy score of 68.40, a F1 score of around 0 and a MCC score of around 0, too. The same can be observed for the poly kernel as well.

When comparing the SVM with the best-performing configuration on the sanitized dataset, it can be observed that the performance in every metric is increased.

6.2 SVM with TF-IDF

Combined with TF-IDF, the SVM can reach an accuracy score of 94.57, an F1 score of 0.91 and an MCC score of 0.87. The performance is reached with the combination of linear kernel, gamma value set to scale and C value of 1.0. The model was trained for 26 minutes. A similar performance was achieved by the model configured with the rbf kernel and C value set to 10. This model took 45 minutes to train. The poly kernel and rbf kernel both reached an accuracy of 68.03 and a F1 and a MCC score of 0. The performance on the raw dataset is increased compared to the highest-performing model trained on the sanitized dataset.

dataset	**C**	**kernel**	**gamma**	**runtime**	**Acc**	**F1**	**MCC**
S	0.1	linear	scale	16m	88.88	0.80	0.74
S	1.0	linear	scale	26m	**94.57**	**0.91**	**0.87**
S	10	linear	scale	42m	93.91	0.90	0.86
S	1.0	rbf	scale	48m	93.45	0.89	0.85
S	10	rbf	scale	45m	**94.28**	**0.91**	**0.87**
S	1.0	rbf	auto	16m	68.03	0.00	0.00
S	1.0	poly	scale	69m	73.08	0.28	0.33
S	1.0	poly	auto	15m	68.03	0.00	0.00
R	1	linear	scale	27m	94.84	0.92	0.88

Table 7: The performance of the SVM using the TF-IDF embedding algorithm.

6.3 SVM with Word2Vec

The highest performance reached by the SVM in combination with the Word2Vec embeddings was achieved using the skip-gram architecture and a vector length of 400. It reached an accuracy score of 92.88, a F1 score of 0.88 and a MCC score of 0.83. The highest performance using the CBOW architecture was reached using a vector length of 300. This model reaches an accuracy of 92.00, a F1 score of 0.87 and a MCC score of 0.81. The training for all models took between 2 to 3 minutes.

The model configuration that reached the highest performance was trained on the raw dataset and achieved even higher results with an accuracy score of 93.21, a F1 score of 0.89 and a MCC score of 0.84.

ds.	**Arch.**	**Vec. len.**	**C**	**kernel**	**gamma**	**runt.**	**Acc**	**F1**	**MCC**
S	CBOW	300	1.0	linear	scale	2m	87.99	0.80	0.72
S	CBOW	300	10	rbf	scale	2m	92.00	0.87	0.81
S	CBOW	400	1.0	linear	scale	2m	88.20	0.81	0.72
S	CBOW	400	10	rbf	scale	3m	91.85	0.87	0.81
S	SG	300	1.0	linear	scale	2m	90.04	0.84	0.77
S	SG	300	10	rbf	scale	2m	92.53	0.88	0.83
S	SG	400	1.0	linear	scale	2m	90.39	0.84	0.78
S	SG	400	10	rbf	scale	2m	**92.88**	**0.88**	**0.83**
R	SG	400	10	rbf	scale	2m	93.21	0.89	0.84

Table 8: The performance of the SVM using the Word2Vec embedding algorithm.

6.4 BERT

The BERT base model, which was trained fully unfrozen, i.e. every layer was updated during training, trained for about an hour per epoch. Every run had a similar batch size of 8. The model that was trained with a learning rate of $2e^{-5}$ achieved an accuracy of 97.76, a F1 score of 0.96, and a MCC score of 0.94. The model

that was trained with a learning rate of $3e^{-5}$ reached the same results, with only the accuracy being a little lower at 97.53. The model trained on the raw dataset achieved an accuracy of 97.98, a F1 score of 0.96, and a MCC score of 0.95. The model that trained for 2 epochs achieved the highest results with an accuracy of 98.21, a F1 score of 0.97 and a MCC score of 0.95.

dataset	**learn r.**	**epochs**	**batch s.**	**runtime**	**ACC**	**F1**	**MCC**
S	2e-5	1	8	2:01:01	97.76	0.96	0.94
S	3e-5	1	8	2:00:10	97.53	0.96	0.94
S	5e-5	1	8	2:00:47	97.21	0.95	0.93
S	2e-5	2	8	4:00:23	**98.21**	**0.97**	**0.95**
R	2e-5	1	8	2:00:04	97.98	0.96	0.95

Table 9: The performance of BERT using various different hyperparameter combinations and the raw and sanitized datasets.

When using BERT only as a feature extractor for the classification head, one achieves the results in Table 9. The highest performance is achieved by the model that was trained for 2 epochs on the sanitized dataset with a learning rate of $2e^{-5}$ and a batch size of 16. It reaches an accuracy of 75.33, a F1 of 0.39, and a MCC of 0.40. The model that was trained with an almost identical configuration, except that the batch size was increased to 32 instead of 16, reaches a lower accuracy of 72.84, a lower F1 score of 0.27, and also a lower MCC score of 0.33. The training times vary, depending on the batch size. For the batch size of 8, one epoch takes about 45 minutes to complete. When increasing the batch size to 16, the training duration increases to 49 minutes. And when increasing it even further to 32, one epoch takes 51 minutes.

dataset	learn r.	epochs	batch s.	runtime	ACC	F1	MCC
S	2e-5	1	8	0:45:17	73.70	30.90	35.75
S	2e-5	2	16	1:38:28	**75.33**	**0.39**	**0.40**
S	2e-5	2	32	1:42:03	72.84	0.27	0.33
R	2e-5	1	8	0:45:19	73.60	30.82	35.20

Table 10: The performance of BERT on the raw and sanitized datasets, finetuning only the classification head.

6.5 DistilBERT

The results from the DistilBERT training can be seen in Table 11. Various hyper-parameter combinations were tested and evaluated. When looking at the training duration, one can see that with every epoch that was added, the duration of the training increased linearly, with one epoch taking about 1 hour and 1 minute. The highest accuracy for a training length of 1 epoch reached the model with a learning rate of $3e^{-5}$, achieving an accuracy score of 96.42. The same is true for a training length of 2 epochs. Here, the model with a learning rate of $3e^{-5}$ reached the highest accuracy as well. For the training lengths of 3 and 4 epochs, only a learning rate of $3e^{-5}$ was tested because the training durations became very long, and thus, the training became pricey as well.

The model that trained for 4 epochs reached the highest accuracy after training for 4 hours and 5 minutes. When looking at the F1 and MCC scores of the models, the results differ from the accuracy scores. The models that trained for 3 and 4 epochs reached the same F1 and MCC scores as the model that trained for 2 epochs. Only the model that trained for one epoch reached a lower F1 score of 0.94 and a lower MCC of 0.92.

When looking at the performance of the model that was trained on the raw dataset, one can observe that the performance of the model is higher on every score compared to the model that was trained on the sanitized dataset with the same

training configuration.

dataset	learn r.	epochs	batch s.	runt.	ACC	F1	MCC
S	2e-5	1	8	1:01:41	95.74	0.93	0.90
S	3e-5	1	8	1:01:43	96.42	0.94	0.92
S	5e-5	1	8	1:01:22	95.99	0.94	0.91
S	2e-5	2	8	2:03:08	97.10	0.95	0.93
S	3e-5	2	8	2:02:59	97.32	0.96	0.94
S	5e-5	2	8	2:02:40	97.14	0.95	0.93
S	3e-5	3	8	3:04:30	97.37	0.96	0.94
S	3e-5	4	8	4:05:33	**97.50**	**0.96**	**0.94**
R	3e-5	1	8	1:01:27	96.85	0.95	0.93

Table 11: The performance of the DistilBERT using the raw and sanitized datasets.

6.6 BERT With Split Texts

For the BERT model that was trained on the split articles, two different configurations were tested. The training time of both models is a little over the 4-hour mark. The performance of the model trained with the sanitized dataset reaches an accuracy of 96.72, an F1 score of 0.95, and an MCC score of 0.93. The model trained on the raw split dataset reached an accuracy of 97.16, a F1 score of 0.95 and a MCC score of 0.94.

dataset	learn r.	epochs	batch size	runt.	ACC	F1	MCC
S	2e-5	1	8	4:07:40	96.72	0.95	0.93
R	2e-5	1	8	4:07:54	97.16	0.96	0.94

Table 12: The performance of BERT using the split-article dataset.

7 Discussion

In the following section, the results of the thesis will be discussed, evaluated and regarded from various different angles. First, the different aspects of the models and their performance will be compared to each other and then a closer look at the dataset will be taken. Before finally discussing the limitations and possible improvements, the problems that arose during the work on this thesis will be discussed.

7.1 Performance

Figure 10 shows the accuracy scores of the best-performing configuration for every model. When comparing the accuracy, one can see that the best-performing model is the BERT model. With a total accuracy of 98.21, it is the best-performing model in this comparison. The second best model is the DistilBERT model. It reaches an accuracy of 97.32, which is very close to the full-size BERT model. The SVM and its word embedding combinations perform surprisingly well, even when compared to the two deep learning models, BERT and DistilBERT. When comparing the embedding algorithms against each other, one can see that TF-IDF reaches the best performance with an accuracy score of 94.57, followed by the CountVectorizer with 93.03. Unexpectedly, the SVM model that received the Word2Vec embeddings performed the worst of the three, reaching an accuracy of 92.88, close behind the CountVectorizer. This result is unexpected because Word2Vec is the most complex and sophisticated embedding technique of the three. This created the expectation that word2Vec would also perform the best, which is not the case. Finally, the worst performance of all is achieved by the BERT model, which was kept fully frozen and only the classification head received training. This model reaches an accuracy of 75.33 and falls far behind the other models, even far behind the worst-performing embedding technique, Word2Vec.

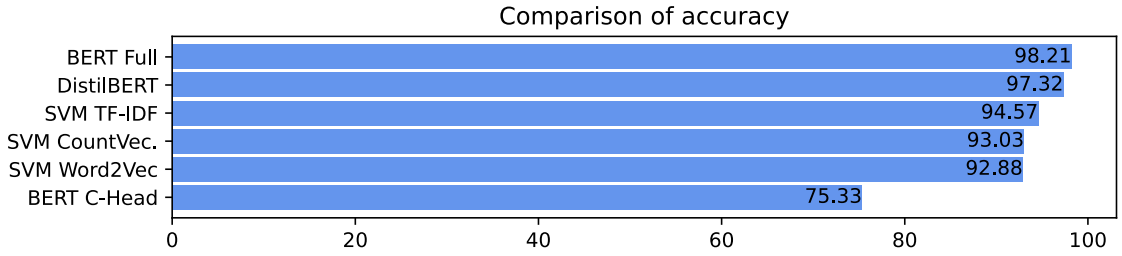

Figure 10: Comparison of the accuracy scores for every model's top performing configuration.

When looking at Figure 11, similar observations can be made. BERT is the top-performing model as well. The order of performance from best to worst is the same, too. One major difference is that the BERT model with the classification head finetuning performs even worse than the other models when looking at its F1 and MCC scores. The gap to the next best model is even larger.

Another interesting observation is that the gap between F1 and MCC score become larger the further down one goes in the ranking. While the gap between the two is only 0.02 for the full-size BERT model, it grows to 0.04 for the SVM with TF-IDF embedding and finally is the largest for the SVM with Word2Vec, reaching 0.05.

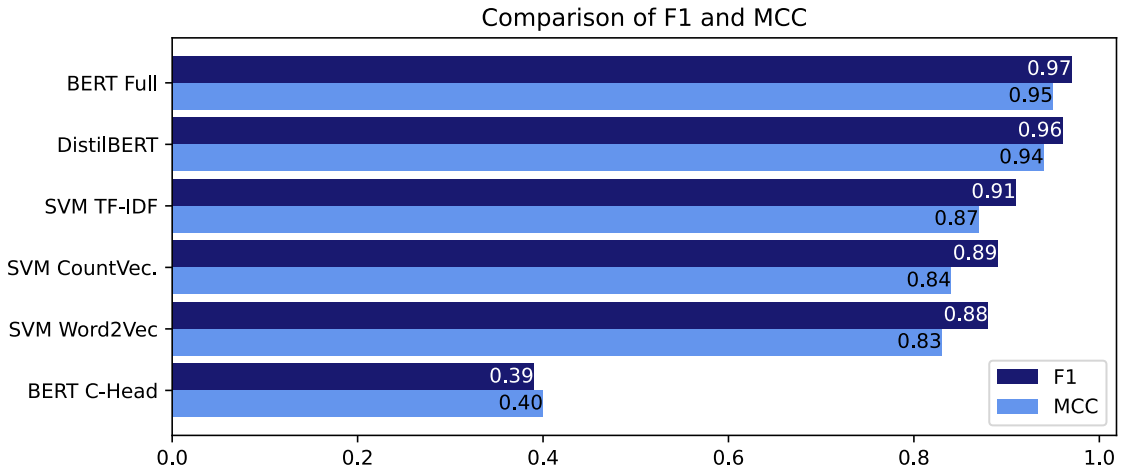

Figure 11: Comparison of the F1 and MCC scores for every model's top performing configuration.

7.2 Training Time

By looking at the performance alone, one can already get a good overview of the models and their performance. But not every model took the same time to train. Yet, this is an important factor when evaluating a model.

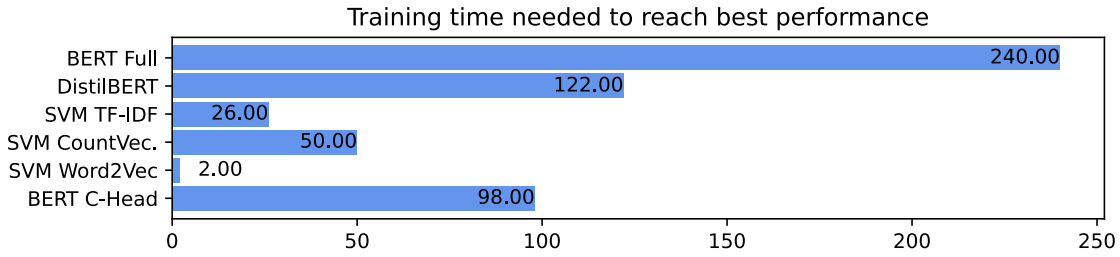

Figure 12: Comparison of the training times needed to reach the best performance. All values are in minutes.

Figure 12 compares all models and the time they took to train to reach their best performance. All values are given in minutes. As expected, the BERT model took the longest to train to reach its top performance and trained for 4 hours. DistilBERT reaches its top performance in about half the time, 2 hours. The SVM combined with the embedding algorithms takes significantly less time to reach its top performance, reaching from 50 minutes for the CountVectorizer embeddings to 2 minutes for the Word2Vec embeddings. These results become especially interesting when factoring in the performance of the models. While BERT reaches the best performance of all models, it takes by far the longest to train and is computationally expensive. DistilBERT, on the other hand, takes only half the time to train but still reaches almost the same performance as the full-size BERT model. The divergence between training time and performance becomes even larger when looking at the embedding algorithms. While the Word2Vec embeddings only take about 2 minutes to train, it reaches an accuracy of 92.88. This result is very impressive when compared to the full-size BERT model, which reaches an accuracy of 98.21 after 4 hours of training.

These findings can greatly impact the choice of which model to implement a fake news detection model. If one's goal is the absolute highest possible performance

and the most accurate predictions, and computational power is abundant, then the full-size BERT model should be chosen. If training cost and efficient training are of higher priority instead, the DistilBERT model offers almost similar performance while reducing the training duration by over half of the time. And finally, if the computational resources are strongly restricted, one can still get usable results by training a SVM model with a word embedding algorithm.

7.3 Split Dataset

Unexpectedly, the models trained on split articles perform worse than their counterparts trained only on the first 512 tokens of the articles. The initial idea behind splitting the articles was to give the models more text data to learn and, thus, be able to separate the two classes better from each other. But when comparing the results in Table 12 with the ones in Table 9, it can be observed that the performance of the split models is lower than their regular counterparts. This could either indicate that the first 512 tokens already have enough information over the articles, that the rest of the articles is not needed to better differentiate between the two classes, or that the splitting method was too rudimentary and didn't help the model understand the differences between real and fake any better. Future research could test and evaluate additional splitting variants, i.e. splitting at every paragraph or splitting at a certain number of tokens. Another option to improve the results could be using more complex systems like Hierarchical Transformers by Pappagari *et al.* [41].

7.4 Dataset Bias

When comparing the models' performance on the raw and sanitized datasets, the differences in performance are only marginal. For the BERT models, this result was expected since the tokenizer cuts the input off after 512 tokens. To further explore the effects of these trivial features on transformer models, follow-up experiments testing models like the Longformer [42] should be conducted. At the time of writing this thesis, no models that were pre-trained on German datasets were available.

When looking at the performance of the SVM models, the differences in performance are only marginal, too. Contrary to the BERT tokenizer, the embedding algorithms took the full articles into account when creating the embeddings. This indicates that the models paid only little attention to the trivial features in the first place. But there is still a difference in performance between the two datasets. Therefore, the articles should be further examined to find even more trivial features in the data that could be removed.

7.5 Dataset

While the dataset itself is already a good starting point for training fake news detection models, there are a few improvements that could be made in future iterations of the dataset. Currently, the articles are divided into real and fake, depending on the publisher. The problem with this approach is that what is considered truth changes over time as new information is discovered. This can lead to a situation where a news article labeled as real contains the same claim as a fake news article that was published at a later point in time. This could have negative effects on the training.

To counter these negative effects, one approach could be the actual fact-checking of the content in the news article. Instead of labeling articles as fake or real, the articles could be divided into single claims. These claims could then be fact-checked, and a fact database could be created, which is then kept up to date. And when classifying articles, the content of an article is classified as real or fake on a claim basis instead of the whole article. This would make the distinction between real and fake much finer, thus improving the dataset and the performance of the models. An approach similar to the proposed idea would be the FEVER dataset [43]. The dataset consists of claims that are annotated with a label whether they are supported claims, refuted claims or if there's not enough info to decide. A similar dataset could be created for German fake news detection and be kept up to date.

7.6 Problems & Possible Improvements

The full BERT model delivered the best performance. This performance could be further improved by training the model for more epochs and testing different batch sizes. As already mentioned, the training on the GPU was limited by restrictions from the GPU provider and the GPU itself. The GPU possessed 8GB of VRAM. This was enough to load the full BERT model and data batches with a batch size of 8. While this was sufficient for training the model and achieving good performance, the performance could be further improved by utilizing a better GPU with more GB of VRAM. This would make it possible to increase the batch size to 16 or 32 and train the model for 2 to 4 epochs, as recommended by Devlin *et al.* [1, p. 4183-4184].

7.7 Limitations and Outlook

The proposed models are restricted by some limitations. The first limitation is the static nature of the trained models. For topics like covid, facts can change over time as new medical discoveries are made and the fact base changes. To mirror these changes in knowledge, the models would have to be retrained on the updated dataset.

News articles not only consist of written text but also of images and other media. Therefore, it could be of great use to create multimodal systems which do not only factor in the written content of a news article but also the other media aspects presented with it. To achieve this, additional models would have to be trained to recognize artificially generated images.

Finally, one last limitation of the models is the restriction to only one topic. In the case of the FANG dataset, the topic is the coronavirus. News articles often touch on various different topics which are connected, like the coronavirus and the economy, for example. To identify if such articles should be classified as fake news, the models need to have a broader understanding of the whole news landscape.

8 Conclusion

This work has shown that it is possible to robustly identify fake news in the German language using word embedding techniques and deep learning approaches for detection. Interestingly, not only the deep learning approach using various forms of BERT models did bring good results, but the naive approach using classical embedding algorithms combined with the SVM made robust predictions on the dataset.

Due to the computing power-hungry nature of the BERT finetuning approach, it would be exciting to see if a naive model, i.e., the SVM, could be finetuned to a point where it matches the performance of BERT. Therefore, more advanced embedding approaches should be tested and evaluated to increase the naive model's performance. But also the trade-off between performance and efficiency could be further optimized by experimenting with the numbers of frozen and unfrozen BERT-layers. In this work, only the extremes, completely frozen or completely unfrozen, were considered due to limitations in time and resources.

Also, further developments in the dataset and their effects on the performance of the models would be interesting to see. A closer examination of the articles could reveal more trivial features in the data, which could be removed. Another aspect would be the number of fake news articles in the data. By scrapping articles from even more fake news sources, their percentage in the dataset could be further increased. This, in turn, could improve the performance of every model trained on the dataset, making German fake news detection models perform even better. Additionally, to further improve the performance of fake news detection models, the use of social media context should be considered since it "has been shown to be a valuable source of information" [11, p. 83], [44], [45].

Identifying fake news will become more and more complex in the future, with unforeseeable challenges. Therefore, as many options as possible should be tested and evaluated to effectively fight the spread of fake news. This thesis gave an overview of various options to choose from, improve, and build on.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[2] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[4] T. Jiang, J. P. Li, A. U. Haq, A. Saboor, and A. Ali, "A novel stacking approach for accurate detection of fake news," *IEEE Access*, vol. 9, pp. 22 626–22 639, 2021.

[5] K. Wagner, "Three major ways social media is changing journalism," https://www.scu.edu/illuminate/thought-leaders/kurt-wagner-12/three-major-ways-social-media-is-changing-journalism.html, Mar 2017, accessed: 30.07.2023.

[6] M. Cantarella, N. Fraccaroli, and R. Volpe, "Does fake news affect voting behaviour?" *Research Policy*, vol. 52, no. 1, p. 104628, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048733322001494

[7] D. G. Nyilasy, "Fake news in the age of covid-19," https://fbe.unimelb.edu.au/newsroom/fake-news-in-the-age-of-covid-19, accessed: 30.07.2023.

[8] P. Gensing, "Angstmache, falschmeldungen und gerüchte," https://www.tagesschau.de/faktenfinder/impfen-fakenews-101.html, Jan 2021, accessed: 30.07.2023.

[9] I. Skafle, A. Nordahl-Hansen, D. S. Quintana, R. Wynn, and E. Gabarron, "Misinformation about covid-19 vaccines on social media: Rapid review," *J Med Internet Res*, vol. 24, no. 8, p. e37367, Aug 2022. [Online]. Available: https://www.jmir.org/2022/8/e37367

[10] A. Mosseri, "Working to stop misinformation and false news," https://www.facebook.com/formedia/blog/working-to-stop-misinformation-and-false-news, April 2017, accessed: 30.07.2023.

[11] J. Mattern, Y. Qiao, E. Kerz, D. Wiechmann, and M. Strohmaier, "FANG-COVID: A new large-scale benchmark dataset for fake news detection in German," in *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*. Dominican Republic: Association for Computational Linguistics, nov 2021, pp. 78–91. [Online]. Available: https://aclanthology.org/2021.fever-1.9

[12] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[13] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Machine Learning: ECML-98*, C. Nédellec and C. Rouveirol, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142.

[14] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] scikit-learn developers, "sklearn.feature_extraction.text.countvectorizer," https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html, accessed: 29.07.2023.

[17] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. [Online]. Available: http://nlp.stanford.edu/IR-book/information-retrieval-book.html

[18] T. Mikolov, W.-t. Yih, and G. Zweig, "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*. Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 746–751. [Online]. Available: https://aclanthology.org/N13-1090

[19] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning based text classification: A comprehensive review," *CoRR*, vol. abs/2004.03705, 2020. [Online]. Available: https://arxiv.org/abs/2004.03705

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[21] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune BERT for text classification?" *CoRR*, vol. abs/1905.05583, 2019. [Online]. Available: http://arxiv.org/abs/1905.05583

[22] Q. T. Nguyen, T. L. Nguyen, N. H. Luong, and Q. H. Ngo, "Fine-tuning BERT for sentiment analysis of vietnamese reviews," *CoRR*, vol. abs/2011.10426, 2020. [Online]. Available: https://arxiv.org/abs/2011.10426

[23] H. Darji, J. Mitrović , and M. Granitzer, "German BERT model for legal named entity recognition," in *Proceedings of the 15th International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and Technology Publications, 2023. [Online]. Available: https://doi.org/10.5220%2F0011749400003393

[24] J. Lee, R. Tang, and J. Lin, "What would elsa do? freezing layers during transformer fine-tuning," *CoRR*, vol. abs/1911.03090, 2019. [Online]. Available: http://arxiv.org/abs/1911.03090

[25] C. Wardle and H. Derakhshan, "Information disorder: Toward an interdisciplinary framework for research and policy making." Strasbourg Cedex, France: Council Of Europe, Oct 2017.

[26] B. Collins, D. T. Hoang, N. T. Nguyen, and D. Hwang, "Trends in combating fake news on social media – a survey," *Journal of Information and Telecommunication*, vol. 5, no. 2, pp. 247–266, 2021. [Online]. Available: https://doi.org/10.1080/24751839.2020.1847379

[27] F. D. Medeiros and R. B. Braga, "Fake news detection in social media: A systematic review," in *XVI Brazilian Symposium on Information Systems*, ser. SBSI'20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3411564.3411648

[28] J. Dirkes, "AI-Fakes erkennen: Twitter-user scheitern in Massen am Daunen-Papst," https://www.ingame.de/news/ai-fakes-erkennen-papst-franziskus-daunenjacke-twitter-midjourney-dalle-chat-gpt-hamburg-92173754.html, April 2023, accessed: 26.07.2023.

[29] B. für Sicherheit in der Informationstechnik, "Deepfakes - gefahren und gegenmaßnahmen," https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kuenstliche-Intelligenz/Deepfakes/deepfakes_node.html, accessed: 30.07.2023.

[30] H. F. Villela, F. Corrêa, J. S. d. A. N. Ribeiro, A. Rabelo, and D. B. F. Carvalho, "Fake news detection: a systematic literature review of machine learning algorithms and datasets," *Journal on Interactive Systems*, vol. 14, no. 1, p. 47–58, Mar. 2023. [Online]. Available: https://sol.sbc.org.br/journals/index.php/jis/article/view/3020

[31] S. S. Ahmed H, Traore I, "Detecting opinion spams and fake news using text classification," *Journal of Security and Privacy*, vol. 1, Jan/Feb 2018.

[32] ——, "Detection of online fake news using n-gram analysis and machine learning techniques," in *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017*, ser. Lecture Notes in Computer Science (LNCS), vol. 10618. Springer, Cham, 2017, pp. 127–138.

[33] T. Jiang, J. P. Li, A. U. Haq, and A. Saboor, "Fake news detection using deep recurrent neural networks," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2020, pp. 205–208.

[34] I. Vogel and P. Jiang, "Fake news detection with the new german dataset "germanfakenc"," in *Digital Libraries for Open Knowledge*, A. Doucet, A. Isaac, K. Golub, T. Aalberg, and A. Jatowt, Eds. Cham: Springer International Publishing, 2019, pp. 288–295.

[35] C. McCormick and N. Ryan, "Bert fine-tuning tutorial with pytorch," https://mccormickml.com/2019/07/22/BERT-fine-tuning/, Jul 2019, accessed: 12.07.2023.

[36] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *CoRR*, vol. abs/1910.03771, 2019. [Online]. Available: http://arxiv.org/abs/1910.03771

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang,

Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019. [Online]. Available: http://arxiv.org/abs/1912.01703

[38] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[39] T. pandas development team, "pandas-dev/pandas: Pandas," Jun. 2023, if you use this software, please cite it as below. [Online]. Available: https://doi.org/10.5281/zenodo.8092754

[40] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[41] R. Pappagari, P. Zelasko, J. Villalba, Y. Carmiel, and N. Dehak, "Hierarchical transformers for long document classification," *CoRR*, vol. abs/1910.10781, 2019. [Online]. Available: http://arxiv.org/abs/1910.10781

[42] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *CoRR*, vol. abs/2004.05150, 2020. [Online]. Available: https://arxiv.org/abs/2004.05150

[43] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: a large-scale dataset for fact extraction and VERification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long*

Papers). New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 809–819. [Online]. Available: https://aclanthology.org/N18-1074

[44] K. Shu, S. Wang, and H. Liu, "Beyond news contents: The role of social context for fake news detection," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 312–320. [Online]. Available: https://doi.org/10.1145/3289600.3290994

[45] X. Zhou and R. Zafarani, "Network-based fake news detection: A pattern-driven approach," *CoRR*, vol. abs/1906.04210, 2019. [Online]. Available: http://arxiv.org/abs/1906.04210

A Appendix

A.1 Sanitized Phrases

The following phrases were removed from the raw dataset to create the sanitized dataset:

- "Wenn Sie mehr über die heimlichen Machenschaften der Elite erfahren wollen, dann lesen Sie das brisante Enthüllungsbuch „Illuminatenblut: Die okkulten Rituale der Elite“ von Nikolas Pravda, mit einigen Artikeln die bereits von Suchmaschinen zensiert werden."
- "Ein handsigniertes Buch erhalten Sie für Euro 30,- (alle drei Bücher für Euro 90,-) inkl. Versand bei Zusendung einer Bestellung an: info@pravda-tv.com."
- "Ein handsigniertes Buch erhalten Sie für Euro 30,- (beide Bücher für Euro 60,-) inkl. Versand bei Zusendung einer Bestellung an: info@pravda-tv.com"
- "Ein handsigniertes Buch erhalten Sie für Euro 30,- (beide Bücher für Euro 60,-) inkl. Versand bei Zusendung einer Bestellung an: info@pravda-tv.com."
- "Ein handsigniertes Buch erhalten Sie für Euro 30,- inkl. Versand bei Zusendung einer Bestellung an: info@pravda-tv.com"
- "„Der Hollywood-Code: Kult, Satanismus und Symbolik – Wie Filme und Stars die Menschheit manipulieren“ (auch bei Amazon verfügbar), mit einem spannenden Kapitel: „Die Rache der 12 Monkeys, Contagion und das Coronavirus, oder wie aus Fiktion Realität wird“."
- "Am 28. April erschien „Der Hollywood-Code: Kult, Satanismus und Symbolik – Wie Filme und Stars die Menschheit manipulieren“ (auch bei Amazon verfügbar), mit einem spannenden Kapitel: „Die Rache der 12 Monkeys, Contagion und das Coronavirus, oder wie aus Fiktion Realität wird„."
- "Am 28. April erschien „Der Hollywood-Code: Kult, Satanismus und Symbolik – Wie Filme und Stars die Menschheit manipulieren“ (auch bei Ama-

zon verfügbar), mit einem spannenden Kapitel: „Die Rache der 12 Monkeys, Contagion und das Coronavirus, oder wie aus Fiktion Realität wird„, warum Stanley Kubrick wirklich sterben musste!"

- "Am 28. April erschien „Der Hollywood-Code: Kult, Satanismus und Symbolik – Wie Filme und Stars die Menschheit manipulieren“ (auch bei Amazon verfügbar), mit einem spannenden Kapitel: „Eyes Wide Shut„, warum Stanley Kubrick wirklich sterben musste!"
- "Am 28. April erscheint das zweite Buch, „Der Hollywood-Code: Kult, Satanismus und Symbolik – Wie Filme und Stars die Menschheit manipulieren„, mit einem spannenden Kapitel: „Die Rache der 12 Monkeys, Contagion und das Coronavirus, oder wie aus Fiktion Realität wird„."
- "Am 15. Dezember 2020 erschien „Der Musik-Code: Frequenzen, Agenden und Geheimdienste: Zwischen Bewusstsein und Sex, Drugs & Mind Control“ (auch bei Amazon verfügbar), mit einem spannenden Kapitel: „Popstars als Elite-Marionetten im Dienste der Neuen Corona-Weltordnung“."

Declaration of Authorship

Specimen for declaration in accordance with § 18 Para. 4 Nr. 7 APO THI

I hereby declare that this thesis is my own work, that I have not presented it elsewhere for examination purposes and that I have not used any sources or aids other than those stated. I have marked verbatim and indirect quotations as such.

Ingolstadt, 03.08.2023

Julian Willweber