Should You Care about Tools in Fake News Detection? A Comparative Study of Cutting-Edge Embedding Methods and Classification Algorithms

Szymon Socha, Piotr Wójcik

Abstract

Recent events such as the COVID-19 pandemic and the conflict in Ukraine have demonstrated that disinformation has reached unprecedented levels. The proliferation of disinformation methods and their increasingly sophisticated concealment techniques have highlighted the need for more powerful tools for detecting it. The main aim of this study is to examine how various natural language processing methods influence the accuracy of fake news detection. We apply a wide range of models to the LIAR dataset introduced by Wang (2017) that consists of 12,800 manually labeled texts collected from POLITIFACT.COM. We investigate how the use of various embedding methods (GloVe, Word2Vec, BERT, RoBERTa, GPT-2) and classification algorithms (Logistic Regression, KNN, SVM, Random Forest, XGBoost, Neural Network) impacts the performance of the models. It appears that the combination of GloVe embeddings with a neural network is the best-performing model. Employing statistical tests, we demonstrate that the choice of embedding method does not exert a significant influence on the performance of the model, while confirming that the selection of the classification method significantly affects its efficacy.

Keywords: fake news detection, NLP, embeddings, classification, machine learning, neural networks

Introduction

In the present era, marked by the rapid digitization and migration of media to the Internet, information accessibility has become increasingly pervasive, offering manifold benefits. With unobstructed information flow, individuals can stay abreast of new events in near real-time, as the distribution of information no longer entails verification hurdles and participation is open to all. Consequently, news agencies have ceded their information-sharing monopoly to social networks and independent news portals, resulting in a decentralized information landscape that is less susceptible to lobbying biases. The elimination of verification processes further enables the instant dissemination of information, providing further advantages.

The widespread availability of information in the digital age brings certain advantages, such as the unhindered flow of information and immediate access to news. However, this phenomenon also entails some drawbacks. The lack of verification measures in conjunction with the unrestricted access to information heightens the risk of disseminating misinformation. Consequently, the sources of the new information or those sharing it need not rely on dependable sources or possess the necessary expertise on the subject matter. Furthermore, since every user is capable of contributing to the creation and transmission

of new information, traditional methods of quality control become ineffectual when the volume of content is immense.

It is worth defining what false information is. In this paper, the term *fake news* is defined as news that contains false or manipulative information, formulated in such a way as to resemble true information as much as possible. They can be spread through various platforms such as television, print media, online news portals, and social media platforms. Fake news usually aims to exert pressure and implant a specific view on a particular phenomenon among a certain group of people. A well-directed campaign of fake news can have a significant impact on social or political events.

At the receiving end of the information transmission network, lies the responsibility of verifying the authenticity and accuracy of the information received. The sheer volume of information available poses a challenge to the recipient, who must filter out false information to remain well-informed. Besides the desire for knowledge, there are other motivations, such as public health and safety concerns, for the detection and filtering of false information. The COVID-19 pandemic is a pertinent example where the detection of fake news was crucial, given the potential impact on public health. Misinformation, including fake cures and conspiracy theories, rapidly proliferated during the pandemic, leading to confusion and real-life consequences. The spread of such news impeded efforts to control the pandemic and its impact. Therefore, methods to identify and label misinformation can aid in the fight against fake news, thereby enhancing public safety. The search for the optimal approach for predicting fake news is a trending research area, with potential significant implications for shaping the future of humanity.

This paper focuses on examining how different approaches to embedding methods and classification methods impact the performance of models detecting fake news. The following models will be investigated: GloVe, Word2Vec, and language models such as Bidirectional Encoder Representation from Transformers (BERT) developed by Google, Robustly Optimized BERT Approach (RoBERTa), and Generative Pre-trained Transformer 2 (GPT-2).

Among the classification methods we compare a logistic regression, k-nearest neighbors, support vector machine classifier, random forest, extreme gradient boosting, and a neural network

This paper, in an unprecedented manner in the literature, analyzes the performance of various combinations of embedding methods with classification methods, examining the combinations between different types of embedding methods and representatives of different types of classification algorithms.

The first part of the paper provides a literature review, including the latest research and various approaches to fake news detection. Based on the literature review we formulate research hypotheses. In the second section of the article we propose the model architecture and describe the dataset, including its source and characteristics. In addition, we explain the architecture of the embedding models and the classification algorithms – their principles of operation, and methods of tuning. The third section presents the results of empirical analysis and discussess the findings in relation with our research hypotheses.

The article ends with the summary and conclusions drawn from the analysis. We also make some suggestions regarding the aspects of this research that could be improved in future studies.

1 Related Work

The detection of fake news using machine learning techniques is a widely researched problem in the literature (Pathak et al. 2020). Researchers have proposed various approaches to address this problem, which can be categorized into three distinct types: traditional methods, deep learning methods, and extensions with language models. The primary objective of these approaches is to enhance the performance of the models to detect fake news as accurately as possible. The classical approach to detecting fake news involves analyzing text length, word frequency, and the use of n-grams (Shu et al. 2017). Additionally, researchers have proposed using sentiment analysis to improve the performance of the model. It has been found that the sentiment of a text is associated with the type of information conveyed, indicating whether the information is misinformation or not (V. Rubin et al. 2016).

The classical approach to detecting fake news using machine learning techniques is gradually being replaced by deep learning methods. Recent research papers demonstrate that deep learning techniques outperform classical approaches in identifying fake news. For example, Wang et al. achieved superior performance by employing a model based on convolutional neural networks (CNN) as opposed to classical text classification methods (W. Y. Wang 2017). Similarly, Rashkin et al. reported comparable results by utilizing the Long Short-Term Memory (LSTM) model (Rashkin et al. 2017). Furthermore, various machine learning algorithms were evaluated on the same dataset to compare their performance in detecting fake news. The findings indicate that deep learning approaches, such as Deep CNN, LSTM, or RNN, exhibit superior performance over classical machine learning methods. These results are consistent across different datasets, demonstrating the overall superiority of deep learning methods in identifying fake news (Mridha et al. 2021).

Moreover, there is a growing body of research indicating that deep learning models can be further enhanced through the incorporation of language models (Conroy, V. L. Rubin, and Y. Chen 2015). The authors of this paper address the topic of fake news detection, achieving the best result with Multimodal CNN, with an F1 Score of 0.87. The second best model turned out to be BERT, with an F1 Score of 0.74. BiLSTM, CNN, and SVM followed in subsequent positions, with scores of 0.70, 0.69, and 0.67, respectively. Therefore, BERT proved to be the best unimodal model, while SVM was outperformed by all deep learning models. Notably, one of the most widely used language models is the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2018). BERT is a pre-trained model designed to learn the contextual relations between words in unlabeled text. There are several variants of BERT available, with RoBERTa (Robustly optimized BERT approach) being one of the most effective in terms of performance. However, RoBERTa is computationally expensive and time-consuming (Liu et al. 2019). Another BERT variant is DistilBERT, a lighter version that requires fewer resources and is faster, but has weaker performance (Sanh et al. 2019). This relationship between language models and deep learning also holds true for the detection of fake news. The incorporation of language models has been shown to improve the performance of existing state-of-the-art models. In comparison to traditional machine learning methods such as SVM, Logistic Regression, Decision Tree, and AdaBoost, language model-based models (DistilBERT, BERT, RoBERTa) have achieved better results on the LIAR dataset. Among the language models, BERT and RoBERTa performed the best with an F1 Score of 0.62, followed by deep learning Conv-HAN model with an F1 Score of 0.59.

In this study, we use the same dataset. In order to compare the obtained results, we also categorize the levels of fake news in the same manner as (J. Y. Khan et al.

2021). This will allow for a comparative analysis of the findings from both studies. (Joy et al. 2022) compared the performance of BERT and RoBERTa. They concluded that RoBERTa, with an F1 Score of 0.98, outperforms BERT, which achieved an F1 Score of 0.89. Furthermore, they compared their results to those of other papers conducted on the same dataset, where DistilBERT achieved a score of 0.94 and SVM scored 0.93.

In this paper, we will focus on two types of BERT models, Vanilla and RoBERTa, and examine how their utilization impacts the improvement of fake news prediction. Given the relatively small dataset at hand, we have decided to employ RoBERTa as it should not pose computational challenges. The investigation omits the examination of the DistilBERT model, which is lighter and faster but exhibits weaker performance.

As previously discussed, language models have been shown to be effective for text classification, including the specific case of detecting fake news. BERT, a widely used language model, has also been confirmed to be useful in this domain by previous research, as highlighted in the paper authored by Gundapu and Mamidi (Gundapu and Mamidi 2021). In the study, the authors demonstrated that BERT, with an F1 Score of 0.9813, outperforms traditional methods such as SVM with an F1 Score of 0.9640. Furthermore, BERT proved to be superior to BiLSTM with an Attention Mechanism, which achieved an F1 Score of 0.9785. Furthermore, a multitude of research works (Alonso-Bartolome and Segura-Bedmar 2021; Y. Wang et al. 2021; Singhal et al. 2019; Aljawarneh and Swedat 2022; Jwa et al. 2019; Yang, Niven, and Kao 2019) have corroborated the applicability of BERT in various approaches to detecting fake news.

In order to enhance its performance, the Bidirectional Encoder Representations from Transformers (BERT) model can be fine-tuned to a particular task or dataset. This approach allows users to tailor the pre-trained model, which incurs significant costs in terms of time and resources, to suit their specific requirements, such as a specific linguistic structure or subject matter of texts. Fine-tuning the BERT model is a cost-effective alternative to training the base model from scratch, although it still demands a substantial amount of computational resources. Technically, the fine-tuning process entails adding an additional layer to the final BERT layer and training the entire model for several epochs (Devlin et al. 2018). As a result, the state-of-the-art performance achieved via fine-tuning has made it widely adopted for various Natural Language Processing (NLP) tasks.

However, Devlin et al. (2018) have contended that state-of-the-art results can also be obtained without the need for fine-tuning. They suggest utilizing pre-trained BERT embeddings as inputs to a Bidirectional Long Short-Term Memory (BiLSTM) network prior to the final classification layer. This technique produces an F1 Score that is only 0.3 percentage points lower than the score achieved via fine-tuning. Peters et al. (2018) have arrived at similar findings. They have conducted a comparative analysis of fine-tuned and feature-based models on various NLP tasks, such as semantic text similarity, named entity recognition, sentiment analysis, and natural language inference. Their results indicate that, except for semantic text similarity, both types of models, fine-tuned and feature-based approaches, exhibit comparable performance on the aforementioned NLP tasks (Peters, Ruder, and Smith 2019).

According to the existing literature, it can be inferred that the incorporation of BERT leads to improved prediction results, irrespective of the model applied to it (Kaliyar, Goswami, and Narang 2021). Kaliyar et al. (2021) proposed a framework in which they employed BERT for word embedding, and subsequently applied diverse machine learning models such as Multinomial Naïve Bayes, Decision Tree, Random Forest, and KNN. The authors noted that the utilization of BERT led to enhanced prediction results. Alternatively, one can aim to enhance the performance of Vanilla BERT by combining it with LSTM (BERT + LSTM). In their study, Rai et al. investigated the performance of BERT

with a modification involving the addition of LSTM on two imbalanced datasets, namely FakeNewsNet (PolitiFact and GossipCop). The incorporation of LSTM in one dataset demonstrated an improvement in the F1 Score from 0.88 to 0.90. In the second dataset, the F1 Score remained unchanged at 0.89 (Rai et al. 2022). Considering the susceptibility of neural networks to overfitting, Naïve Bayes has been observed to outperform deep learning and language models on small datasets. However, as the size of the dataset increases, deep learning models begin to surpass Naïve Bayes in fake news prediction. The utilization of language models is expected to yield improved and state-of-the-art performance (J. Y. Khan et al. 2021).

In addition to the extensively described models for predicting fake news based on the BERT architecture found in the literature, there is another family of models that has gained popularity recently. The language models Generative Pre-trained Transformer (GPT), created by OpenAI, have gained significant popularity in recent months due to their unprecedented ability to generate human-like text. These models can be used to extract embeddings that are useful for predicting fake news. The latest GPT model released in open-source mode is GPT-2, which will be utilized in this study.

There are very few references in the literature that discuss the idea of using GPT-2 for fake news prediction. However, the results of one study indicate that RoBERTa outperforms GPT-2 in terms of accuracy, achieving 0.979 compared to GPT-2's accuracy of 0.974 (in the same study, BERT achieved 0.971) (Shifath, M. F. Khan, and Islam 2021). Evidence of RoBERTa's superiority over GPT-2 can also be found in other studies (Cruz, Tan, and Cheng 2019). Nevertheless, the literature also provides confirmation that using GPT-2 for fake news prediction may perform better than BERT. Although the authors do not extract the pre-trained embedding layer, this is the closest approach found in the literature regarding the use of GPT-2 for fake news prediction (Goel et al. 2021).

When comparing the results of the models from the presented studies, it can be observed that they significantly differ in terms of performance. This is due to the fact that the quality of predictions is largely dependent on the dataset used. Data sets, or rather news texts, are entirely dependent on the data collection methodology. Some datasets (e.g., Fake News[1]) are characterized by easily detectable patterns. Typically, in such datasets, both fake and true news contain specific words that allow for easy classification. For studies that used these types of datasets, the performance will be nominally very high.

On the other hand, there are also datasets such as LIAR[2], used in this paper, where the performance of the same model, set up as for other datasets and measured with the same statistics, is nominally weaker than the others. Its structure does not exhibit easily detectable dependencies between fake news and true news.

Therefore, it is important to note that due to the differences in the datasets, it is not possible to compare the performance of models between different studies if they used different datasets. It is possible to compare models within a specific paper and then verify whether a particular model performs similarly compared to the other models in other studies.

In each of the aforementioned sources, the authors state that they achieved state-of-the-art performance using language models. We will verify this in our research. Therefore, the first research hypothesis of our study assumes that:

H1: Language model-based models (BERT, RoBERTa, GPT-2) outperform the other embedding models (Word2Vec, GloVe) in fake news detection.

[1] `https://www.kaggle.com/mrisdal/fake-news`

[2] `https://www.cs.ucsb.edu/~william/data/liar_dataset.zip`

During the literature review, we did not find any study that comprehensively examined how different types of embeddings, in combination with various classification methods, affect the model's performance. Additionally, due to differences in datasets among the papers, it is not possible to resolve this issue by analyzing the results of different studies. Therefore, we have decided to investigate whether embedding methods and classification methods affect the model's performance. To examine this phenomenon separately we formulate a second research hypothesis:

H2: The type of classification method used significantly influences the accuracy of a fake news detection model.

To address these hypotheses, we employ statistical tests to compare the obtained results of all model combinations.

2 Methodology

2.1 Architecture

One of the important aspects of this study is to construct such an architecture that allows for the comparison of results between different models. Since the quality of predictions of models composed of various embedding methods and classification algorithms is being compared, it is necessary to ensure that the entire process is designed correctly. The individual variants of embeddings should be trained on the same data, and the classification algorithms on the same embeddings. Figure 1 illustrates the architecture designed for the purpose of comparing the results of the models in this study.

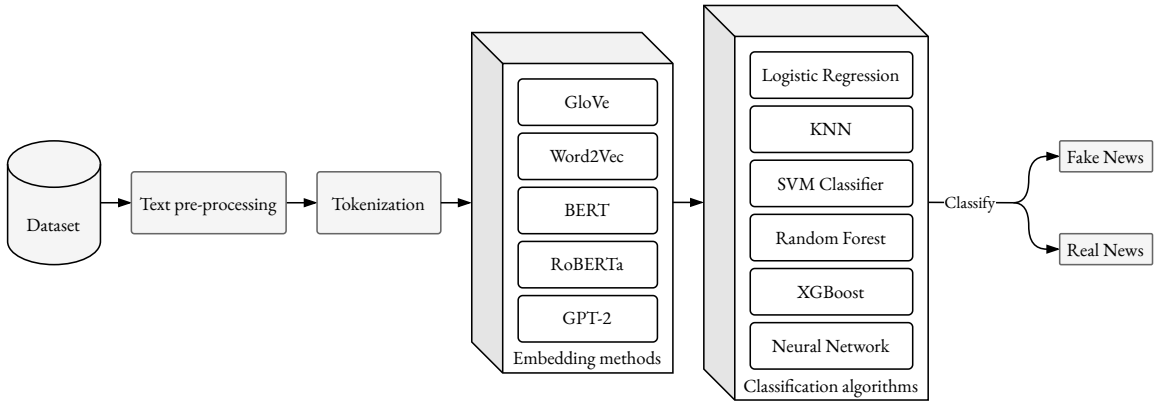

Figure 1: Architecture of each embedding-classifier model combination

The prepared dataset, previously subjected to cleansing and tokenization, has been divided into a training and testing set (further details regarding the dataset's characteristics are provided in the subsequent section). Subsequently, the data underwent an embedding procedure. The same combination of data was employed for all embedding methods utilized.

The Word Embeddings obtained in the previous step served as an input for the classification algorithms in the subsequent step. After each word embedding procedure, each algorithm received exactly the same set of data. Hyperparameter tuning on the training set was performed using cross-validation with random search.

Finally, all combinations of 5 embedding methods and 6 classification methods resulted in 30 different fake news detection models. For each embedding-classifier combination,

text classification into fake news and real news was performed on the test set created in the first step. More details about the obtained results will be presented in the Results section.

2.2 Dataset

Access to high-quality data is an important challenge faced in data modeling and machine learning. In the absence of high-quality data, it is impossible to construct accurate machine learning models. Poor quality data, characterized by unreliable data collection methods, a large number of errors, incompleteness, or mislabeling, can compromise the effectiveness of any machine learning algorithm. This issue is particularly pertinent in the context of datasets containing fake news, where labeling is not as straightforward as with other datasets. The task of manual labeling, which is required to obtain ready-made texts labeled as fake news or real information, can be both time-consuming and challenging, requiring substantial trust in the person performing the labeling. As such, the labeling process is prone to significant human error, which can impact the selection of an appropriate dataset.

This research article aims to investigate how the use of various natural language processing (NLP) methods influences the accuracy of fake news prediction, rather than building a model to accurately predict fake news. Therefore, when selecting a dataset, the most crucial considerations are the quality of the labeled observations and the consistency of the labeled texts across the entire dataset. Additionally, the text should closely align with the research area, as the focus of this study is on detecting fake news. To construct a universal tool for detecting fake news, the dataset should include varied topics and text characteristics, reflecting the real-world distribution as accurately as possible. In this way, the model can learn from natural data, picking up the nuances present in different texts, such as writing style, formality, and topics. However, as noted earlier, the primary objective of this article is not to construct a ready-made tool but rather to examine and impact the predictive quality of different NLP methods.

The dataset selected for the research that meets the above criteria is the LIAR dataset, a publicly available dataset for detecting fake news presented by Wang (2017). The dataset consists of 12,800 manually labeled texts collected from POLITIFACT.COM. These texts are characterized by different contexts, and each of them has been verified by POLITIFACT.COM editors. The texts have been labeled into 6 fine-grained labels evaluating the degree of truthfulness of the text: *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true* (W. Y. Wang 2017).

In the context of this paper, we do not deal with multi-class classification. Instead, we reduce the number of classes to two, classifying *true* and *mostly-true* and *half-true* as true news, and the remaining as fake news. As a result, we obtain a binary classification of text into true news and fake news. In the training set, 5,104 observations are classified as fake news compared to 6,420 pieces of information classified as true. In the test set, there are 553 fake news and 714 true news. Both the training and test sets have a similar ratio of the two classes. We arbitrarily acknowledge that such a ratio of two classes is sufficient for conducting the study, and we do not undertake any further steps to improve data balance.

Next, we subjected the raw dataset to the process of data cleaning and feature engineering. The topics of the news articles were concatenated with their respective content, resulting in a longer text that retains all pertinent information. The cleaned texts were further processed by removing stopwords and punctuation marks. The topic information was omitted from the analysis as it unnecessarily complicates the dataset, potentially

introduces uncertainty regarding its correctness, and is difficult to obtain for new data. Importantly, introducing additional variables would add noise to the prediction results, thereby complicating the analysis of the impact of different embedding methods on prediction accuracy. Similarly, the time variable of article publication was also omitted. The prepared dataset is now primed for subsequent analysis and the application of advanced embedding techniques to effectively process and represent the news content.

2.3 Embeddings

The next step after cleaning the data is to convert it in a way that is understandable to the computer. Textual data have the characteristic of forming a coherent whole. Individual letters form words, words form sentences, and sentences form longer pieces of text with a logical structure. Natural language is full of nuances. The same words can have different meanings and emotional connotations depending on the context. Understanding language comes naturally to humans and does not pose major difficulties. The human brain easily captures the appropriate context of words and their meaning, combining them into a logical and coherent text.

However, computers operate in a completely different way than the human brain. Computers are only capable of analyzing sequences of bits, which when arranged in a specific order, yield a result in the form of an element of a larger whole. This way, a sequence of bits can be presented as individual letters. Further, a sequence of letters can be represented as words. Then, in the tokenization process, individual words can be assigned a specific number. This way, the computer is able to decode the data represented by these words into something it can understand.

Nevertheless, the problem of conveying a deeper understanding of the text, the meaning of individual words, and the semantic relationships between them as understood in terms of similarity and analogies, still remains to be solved. The answer to this problem is Word Embedding. Word Embedding is used to represent words using numerical vectors, in such a way that the distance calculated between two words that are semantically similar is small, and between two semantically different words is large.

The Figure 2 illustrates the graphical interpretation of Word Embedding, often encountered in the literature. Similar words, such as King and Queen, Man and Woman, are closer to each other in the vector space. Two dimensions can be interpreted. The first dimension determines the fact of being a member of the family, while the second dimension indicates gender.

Trained unsupervised machine learning algorithms handle the conversion of words into vectors from large sets of textual data. These algorithms learn to predict the context of a given word by analyzing the words with which it is associated. In this process, they learn the context of words and the semantic relationships between them. Examples of such algorithms include BERT, Glove, Word2Vec, and GPT-2, which has not yet been investigated in the literature for the purpose of fake news detection.

2.3.1 GloVe

GloVe (Global Vectors) is an unsupervised machine learning algorithm aimed at creating word embeddings by aggregating co-occurrence matrices of words that contain information about how often individual word pairs occur in a given text corpus.

The starting point is to create a co-occurrence matrix. The values in this matrix indicate how often a given word pair occurs together. The next step is to calculate

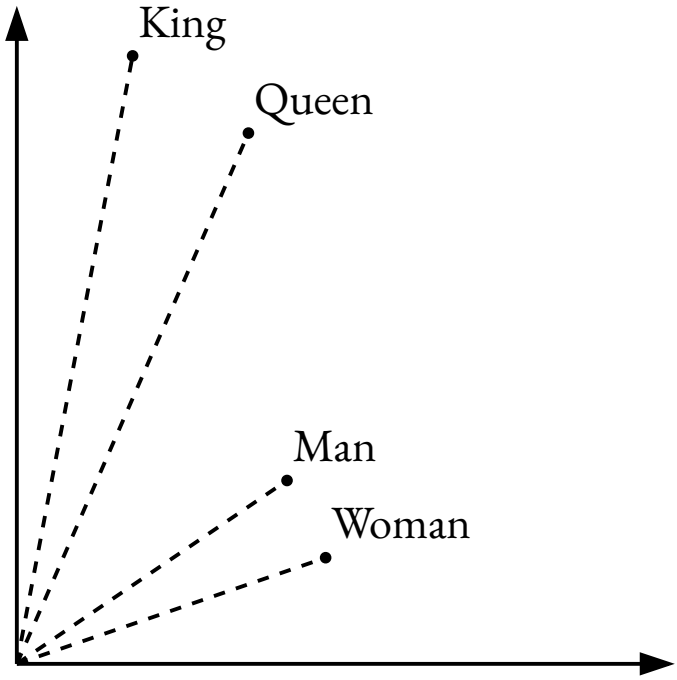

Figure 2: Similar words are closer together

the probability of one word occurring in the presence of another word, or rather the ratio between successive probabilities (Pennington, Socher, and C. D. Manning 2014). Without delving into mathematical details, word embeddings are obtained using a matrix subjected to factorization in a process similar to dimensionality reduction (Albrecht, Ramachandran, and Winkler 2020).

It is possible to use pre-trained word vectors trained on large text datasets. In this paper, a model trained on Common Crawl was used, consisting of 840 billion tokens and 2.2 million words, which allowed for obtaining 300-dimensional vectors for each word.

2.3.2 Word2Vec

The Word2Vec algorithm is yet another method for obtaining word embeddings. Unlike GloVe method, it is not a method that consists of a single, homogeneous algorithm. Word2Vec can be based on two distinct models: the CBOW and Skip-Gram Model.

The functioning of these two algorithms can be intuitively interpreted as two opposites. CBOW works as a model that predicts a given word based on its context. On the other hand, the Skip-Gram Model is a model that predicts the context of a given word, i.e., the words preceding and following it. Both models are built on a 3-layer neural network with an input layer, a hidden layer, and an output layer.

The originator of the Word2Vec method recommends using the Skip-Gram Model with negative sampling, which outperforms other variants in research. In this study, pre-trained vectors were used, trained on a portion of the Google News dataset consisting of about 100 billion words. The model consists of 300-dimensional vectors for 3 million words and phrases. It was not specified which Word2Vec method was used to obtain these vectors (Mikolov et al. 2013).

2.3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning Large Language Model founded on the transformer self-attention mechanism. The transformer architecture incorporates encoders responsible for assimilating input data and assigning importance weights to individual words via the self-attention mechanism. This allows BERT to acquire an understanding of contextual dependencies among words within a sentence. Unlike sequential directional models, which process textual input data in a linear fashion, transformer encoders simultaneously process all words, resulting in an inherently bidirectional nature (Vaswani et al. 2017). The bidirectional sentence-level classification architecture of BERT is depicted in the provided Figure 3.

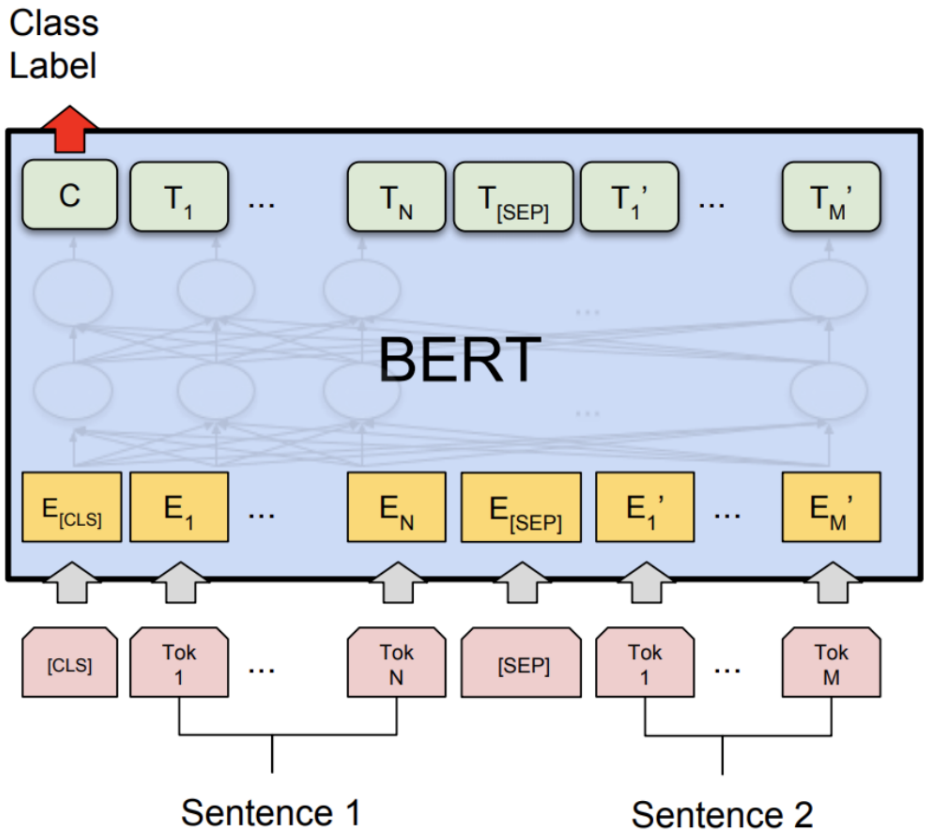

Figure 3: BERT model architecture (Devlin et al. 2018)

BERT belongs to the class of masked language models. This implies that the learning process of BERT can be interpreted in such a way that the model randomly masks a selected word and, based on the surrounding words (both preceding and succeeding), predicts the masked word using the mechanisms described earlier. Next Sentence Prediction (NSP) serves as another objective for the model during the pre-training process. In this procedure, BERT combines two randomly masked sentences as inputs. Subsequently, the model generates predictions regarding whether the given pair of sentences logically follows each other or not.

A noteworthy characteristic of BERT is its pre-training capability. During the pre-training phase, the BERT model acquires contextual understanding of a given word by analyzing the surrounding words. This is achieved through the application of a masked language model (MLM), where tokens are randomly masked, and the model attempts to predict the missing word's semantics. BERT offers a variety of pre-trained models that can be utilized for diverse purposes. Specifically, the BERT-base model, employed in this

study, encompasses 12 encoder stack layers, 768 hidden units, and 12 multi-head attention heads, comprising a total of 110 million parameters. The pre-training of this model was conducted on a dataset containing 11,038 books and the English edition of Wikipedia.

BERT requires input data to be specially converted to the required format before being used in the pre-trained model. This enables the use of its key component, word embedding. During word embedding, BERT uses a technique called WordPiece. This technique is based on breaking words down into smaller subwords. This is done to capture the meanings of words that may have different interpretations or different meanings depending on the context. The WordPiece technique thus creates all combinations that can occur in a given text, assigning each extracted part a pre-trained vocabulary vector. These vectors then serve as input data for the previously trained transformer architecture.

An important feature of BERT is the fact that it can be fine-tuned for use in a specific task or adapted to the characteristics of a given dataset. Fine-tuning involves running the pre-trained model training on a new dataset for several epochs. According to the literature, such a trained model achieves state-of-the-art performance. However, in this paper, we will not apply fine-tuning. Instead, we focus on comparing the impact of using different embedding methods on changes in model performance. Additionally, as mentioned earlier in the literature review, fine-tuning can be replaced by using a classifier on the pre-trained BERT architecture.

In this research, we use BERT as a feature extractor. We extract embeddings for the first token ([CLS] token) from the last hidden layer of the model.

2.3.4 RoBERTa

The Robustly Optimized BERT Approach (RoBERTa) is a modification of the BERT model. The creators of the model have modified the way in which the input data is masked. In the case of BERT, masked words were inputted to the model as input data. This is a weakness because during the training of the model in successive epochs, the model learned from the same duplicated data. RoBERTa introduces a change in the form of dynamic token masking during training in successive epochs. Additionally, another change introduced in RoBERTa is the removal of Next Sentence Prediction (NSP) during pre-training. The remainder of the architecture is identical to that of the BERT model. Furthermore, RoBERTa is trained on a larger amount of data (160GB corpus text, with 16GB of the same data used for training BERT). Due to the improvements made, RoBERTa is expected to demonstrate better performance than BERT (Liu et al. 2019).

2.3.5 GPT-2

Another model using the transformer architecture is the GPT-2 (Generative Pre-trained Transformer 2) model. Unlike BERT, it does not process input data in two directions but rather employs a unidirectional architecture that reads data from left to right. Another distinction is that GPT-2 utilizes decoder-only transformers (while BERT employed encoder-only transformers). This difference arises from the distinct purposes of the models. BERT was designed as a tool to create various models for different applications at a low cost by adding an additional layer. Consequently, BERT employs encoders whose output comprises contextualized embeddings that can serve as inputs for further models. On the other hand, GPT-2 serves an entirely different purpose, which is text generation. Since the model's output should be comprehensible to humans, GPT-2 relies solely on decoder-only transformers.

Decoders in GPT-2, similar to BERT, employ self-attention mechanisms. However, unlike BERT, self-attention in GPT-2 only takes into account preceding words. Otherwise, if considering the subsequent words in the sequence, the model would incorrectly learn to predict the next word by simply returning the next word in the sequence with the highest probability of occurrence. Another difference between GPT-2 and BERT lies in the fact that GPT-2 processes data token by token (while BERT processes the entire text simultaneously). GPT-2 generates predictions for the next word based on the preceding text. The output of one iteration simultaneously serves as the input for the next iteration.

2.4 Classification algorithms

The final component of the fake news detection model architecture is a classifier, which outputs only one of two values, namely whether a given sentence is a fake news or not. Classifiers are models whose results belong to previously defined classes, namely binary classification when there are two possible return values, or multiclass classification when there are more possible classes. There are many types of classifiers, which differ in structure and purpose depending on the characteristics of the data to which they are applied. Some classifiers perform better with one type of data than others. In selecting a group of classifiers for the study, we aimed to cover the widest range of popular classification models. Thus, logistic regression, which is a representative of econometric approach, K-Nearest Neighbours (KNN) and Support Vector Machine Classifier (SVC), which are representatives of non-parametric models, Random Forest as a representative of ensemble methods that use decision trees, eXtreme Gradient Boosting (XGBoost) as a representative of ensemble methods that use boosting, and neural networks, which are a separate class of models also widely used in classification, were chosen for the study.

2.4.1 Logistic Regression

Logistic Regression is a model used for binary classification. This model is based on the utilization of the logistic function to model the relationship between the variables in the model and the probability of the occurrence of a true event. The output of the model ranges from 0 to 1, which can be interpreted as the probability that a given observation is positive.

The logistic regression model is fitted by optimizing a loss function (in this case, the cross-entropy loss). In the optimization process, the aim is to obtain weights for the input variables (or parameters) that minimize the difference between the estimated probabilities and the actual values in the training set.

The logistic regression model can also be tuned. By having a set of variables, one can make choices about which variables are most useful in modeling a particular phenomenon, thereby minimizing model overfitting and improving its generalization. This is achieved through a technique called regularization. It introduces a "penalty" for the model for assigning larger weights to variables, thus "encouraging" the simplification of the model by eliminating the excessive influence of unnecessary variables. Two types of regularization are distinguished: L1 (Lasso Regularization) and L2 (Ridge Regularization). The key difference between them is that L1 regularization allows for the possibility of reducing weights to zero, completely eliminating variables from the model. On the other hand, L2 regularization does not allow for such a possibility, as variables always retain some nonzero weight. Additionally, it is possible to adjust the strength of the regularization. By selecting appropriate parameters, it is possible to find an optimal trade-off between

the complexity of the model and its generalization (Tibshirani 1996).

Logistic regression is a parametric model that is based on certain a priori assumptions about the input data. The model assumes a linear relationship between the variables and the log-odds of positive classes (this assumption can be relaxed by introducing new variables that are combinations of the remaining ones).

2.4.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is an algorithm that can be used for both classification and regression tasks. It is non-parametric, meaning that no assumptions about the data are made during the fitting process. Instead, the data is directly used for making predictions. KNN operates on the principle that similar observations belong to the same classes. Upon receiving a new instance, the algorithm calculates the distance to the remaining observations and selects the K closest neighbors (hence the name of the algorithm). The new instance is then assigned to the cluster with which it shares the most common points, or in the case of regression, the mean is calculated.

Model tuning involves adjusting the number of neighbors (K) considered by the algorithm. Selecting an appropriate number of clusters is crucial because a too low value of K can lead to overfitting and high noise in the data. On the other hand, a large number of neighbors can make it difficult to capture local variations in the data.

To improve the model's performance, one can also customize the method used to calculate distances between instances. The most popular distance calculation methods include Euclidean distance, Manhattan distance, and Minkowski distance. It is important for accurate distance calculation that the variable ranges are similar. Therefore, it is important to normalize the variables prior to training KNN models (Fix and Hodges 1989).

2.4.3 Support Vector Machine Classifier

Another popular algorithm used for both regression and classification is Support Vector Machine (SVM). The algorithm is based on finding the best possible hyperplane that separates the data into distinct classes. The constructed hyperplane aims to be as far as possible from two different data points belonging to different classes (these points are called support vectors), while minimizing the classification error. To achieve this, the hinge loss function is minimized, which penalizes the algorithm for misclassified observations.

By applying the so called kernel trick and mapping the data to a higher-dimensional space, SVM gains the ability to capture nonlinear relationships between variables. The most popular kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.

During SVM tuning, the parameter C can be modified, which influences the trade-off between maximizing the distance of the hyperplane from two different points and minimizing the classification error. A small value of the C parameter flattens the hyperplane, causing the model to potentially misclassify more points. A large value of the C parameter makes the hyperplane more prone to adjusting its shape to individual observations, while also exposing the model to overfitting (Vapnik 2000).

2.4.4 Random Forest

Random Forest is a non-parametric algorithm that can successfully be used in both classification and regression tasks. It represents a family of ensemble methods, where the final predictions are the result of combining predictions from multiple decision trees. By leveraging the results of individual decision trees, Random Forest predictions are more accurate and robust. Each decision tree is estimated on randomly selected variables and observations selected with replacement. This technique, called bootstrap sampling, allows each decision tree to be trained on a slightly different dataset.

For each randomly selected subset of data, a decision tree is created, which iteratively divides the given dataset into two or more subsets. These divisions are made in such a way that the resulting groups differ as much as possible from each other. The degree of dissimilarity between subgroups can be measured, for example, using Gini impurity or information gain. Subsequent splits are generated until a stopping criterion is reached, such as maximum depth or minimum number of observations in a leaf node.

After creating all the trees, their predictions are combined into one. In the case of classification, this is done through majority voting, where the most frequently indicated value is selected.

In comparison to the previously described methods, Random Forest offers a larger number of hyperparameters for tunning. The most important ones include the number of trees, maximum depth, minimum samples split, and feature subset size. Modifying the number of trees in the forest usually involves a trade-off between model training cost and accuracy. The more trees, the more accurate the predictions, but at a higher prediction cost. The maximum depth of a tree determines its complexity level. Deeper trees capture variable dependencies better, but also increase the risk of overfitting. The minimum samples split parameter determines the minimum number of observations required to make a split. A smaller value results in a more complex tree. Modifying the feature subset size parameter affects the level of randomness and diversity among individual trees (Breiman 2001).

Due to its ability to handle multidimensional data, handle missing data, and provide feature importance, Random Forest is a common choice for creating machine learning models.

2.4.5 eXtreme Gradient Boosting

The eXtreme Gradient Boosting (XGBoost) algorithm is another non-parametric algorithm based on the concept of decision trees. It is known for its high performance and effectiveness across a wide range of tasks. Similar to Random Forest, it represents a family of ensemble methods, specifically gradient boosting.

XGBoost predictions are generated by combining the predictions of multiple weak-performing models (typically decision trees) into a single strong model. The decision trees are created sequentially, and each subsequent tree is estimated based on the error of the previous tree, aiming to improve it by minimizing the loss function. As the name suggests, XGBoost utilizes the gradient boosting algorithm, which employs gradient descent optimization to minimize the loss function. In each iteration step, the algorithm computes the gradient of the loss function based on the predictions and updates the model by training the next tree on the residuals from the previous tree.

The idea of hyperparameter tuning in XGBoost is very similar to that of Random Forest. Just like in Random Forest, users can adjust the maximum depth of the trees, the number of trees in the ensemble, regularization parameters, and subsampling ratio.

Changing the hyperparameters yields similar results as in Random Forest. Additionally, one can modify the learning rate, which refers to the speed at which subsequent models follow the gradient, effectively reducing the error of the previous model. A too high learning rate may cause the model to skip the global minimum of the gradient and settle for a local minimum, leading to overfitting. Conversely, a too low learning rate results in slower convergence, prolonging the model's learning process. By modifying the loss function and applying a sigmoid transformation to the final predictions, the model can successfully be used for binary classification (T. Chen and Guestrin 2016).

2.4.6 Neural Network

The neural network model is inspired by the functioning of the human brain in its architecture. Neural networks operate based on interconnected nodes called neurons, which collectively participate in data processing and prediction generation. Among all the models discussed above, Neural Networks have the most diverse range of applications in various fields. They can be applied to all tasks that were previously discussed for other models. Additionally, they can be used for image recognition and natural language processing, which were not possible with the previously mentioned models. Standard neural networks have a fixed number of parameters, representing a family of parametric models.

The process of training neural networks consists of two main components: forward propagation and backpropagation. During forward propagation, the input data pass through the network and undergo computations layer by layer. Each neuron in the network transforms the data using adjustable activation functions and learnable parameters such as weights and biases. This process continues until the data traverse the entire network, reaching the output layer and obtaining predictions.

After obtaining predictions, the next step is to compare them with the true values and calculate the prediction error (or loss function). Subsequently, in the backpropagation process, this error is propagated in the reverse direction, from the output layer to the input layer. During backpropagation, the weight and bias of each neuron are adjusted using the gradient of the loss function, aiming to minimize the final error. The entire process can be repeated multiple times to optimize the model's performance. One such iteration (a complete pass of forward propagation and backpropagation) is called an epoch.

Neural networks can be tuned in various ways. One of them is modifying the network architecture, such as adding or removing layers and adjusting the number of neurons in each layer. Dropout is a commonly used technique, where randomly selected neurons are omitted during the learning process, improving the model's generalization. Other regularization techniques, such as L1 and L2 regularization, aim to mitigate overfitting. Similar to XGBoost, learning rate can be adjusted in neural networks. It functions analogously to XGBoost, determining how quickly a given model converges (Wager, S. Wang, and Liang 2013; S. Wang and C. Manning 2013).

After appropriately adjusting the network architecture, it can successfully be utilized for classification tasks. The output layer can use activation functions like softmax to convert the output into probability values for different classes (Raschka and Mirjalili 2019).

3 Results

The raw dataset was already divided into a training set and a testing set. We subjected it to text preprocessing and applied tokenization.

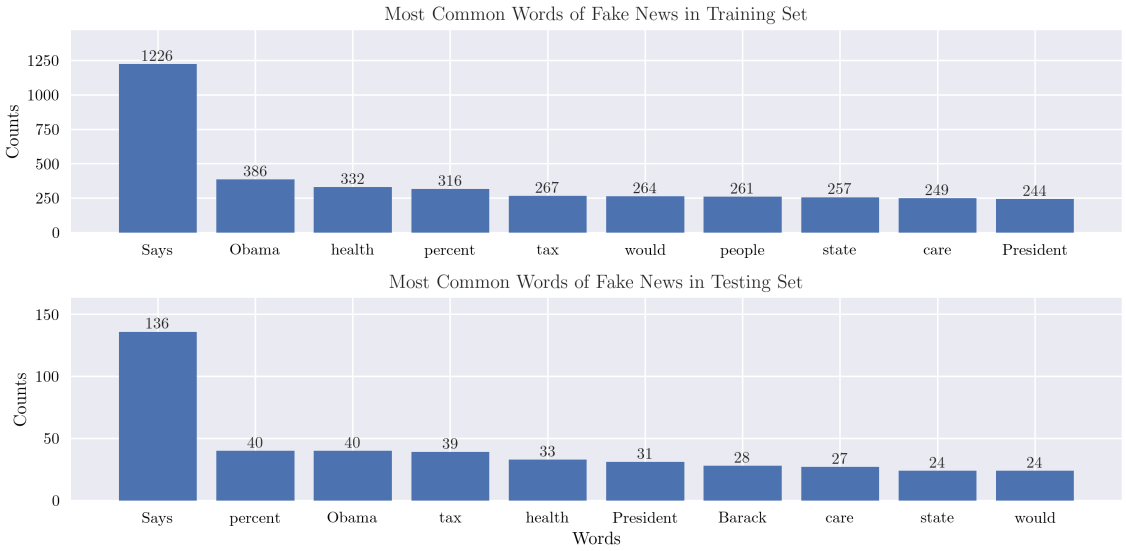

Figure 4: Top 10 most common words for fake news

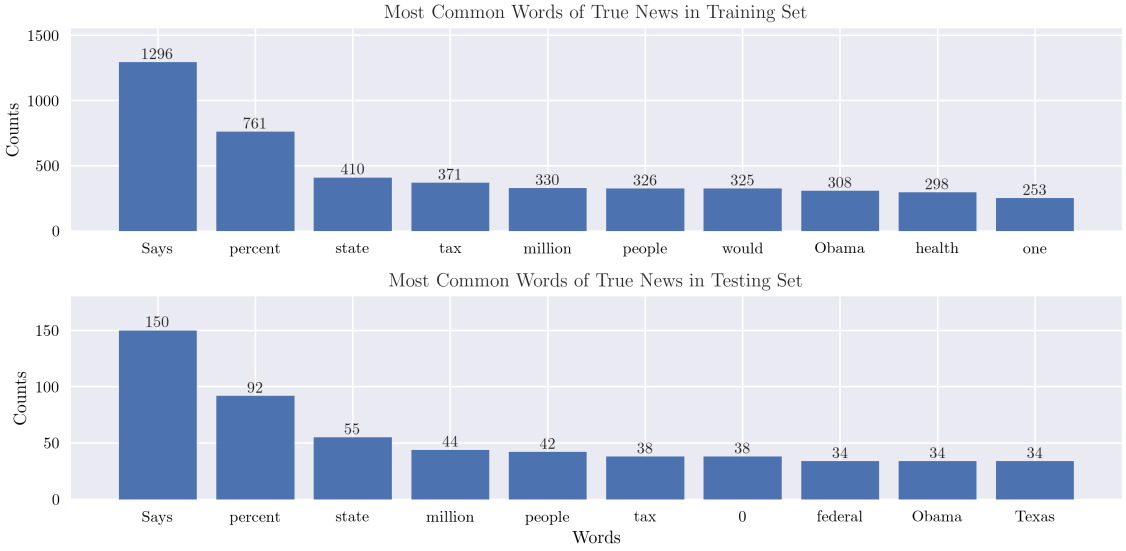

Figure 5: Top 10 most common words for true news

The prepared dataset was examined with regard to the most common words. The analysis was divided based on the training and testing sets, as well as the type of news, whether it is a fake news (Figure 4) or true news (Figure 5). The analysis of the following charts allows us to conclude that no distinct pattern of the most popular words stands out among the identified groups, which could disrupt the study.

Using the architecture described in the previous section, we built separate models for each combination of embedding and classification algorithms. To verify the research hypotheses, it was necessary to examine the performance of each model thoroughly.

Since the study operated on a slightly imbalanced dataset, closer attention had to be paid to the issue of selecting appropriate metrics. Not all metrics are suitable for use on imbalanced data. In this case, an example of an incorrect metric would be accuracy. Accuracy is calculated as the ratio of correctly classified samples to the total number of samples, regardless of the class distribution. Therefore, when one class predominates

numerically over the other, the model can achieve high accuracy simply by assigning all predictions to the majority class, thus making poor predictions on the less numerous class.

In the case of imbalanced data, a much better choice would be to use metrics such as Balanced Accuracy, F1 Score, or AUC-ROC, which were employed in this study.

Balanced Accuracy is a metric that deals with imbalanced classes by introducing a balanced evaluation of the model's performance on each class. It is the average percentage of true positive results that have been correctly classified (in other words, recall or sensitivity) for each class. However, Balanced Accuracy does not take into account precision, which focuses on the ratio of correct positive classifications. Depending on the goal we want to achieve when building a model, we may be interested in different metrics. For example, precision is crucial when the priority is to minimize false positives.

Model		Score			
Embedding	Classification	Accuracy	B. Accuracy	F1 Score	ROC AUC
GloVe	Logistic Regression	60.1	58.4	49.3	58.4
	KNN	55.3	53.9	45.5	53.9
	SVM Classifier	59.2	57.9	50.3	57.9
	Random Forest	60.0	58.4	49.9	58.4
	XGBoost	61.2	59.3	49.7	59.3
	Neural Network	60.6	60.2	**55.7**	60.2
Word2Vec	Logistic Regression	61.7	59.7	49.9	59.7
	KNN	54.0	53.4	47.9	53.4
	SVM Classifier	60.5	59.4	53.0	59.4
	Random Forest	57.9	56.1	46.6	56.1
	XGBoost	57.1	56.0	48.7	56.0
	Neural Network	60.6	59.9	**54.6**	59.9
BERT	Logistic Regression	60.5	58.9	50.7	58.9
	KNN	55.9	54.9	48.3	54.9
	SVM Classifier	58.6	57.4	50.3	57.4
	Random Forest	57.6	45.7	45.7	55.7
	XGBoost	59.4	58.0	50.1	58.0
	Neural Network	58.8	57.8	**53.7**	57.8
RoBERTa	Logistic Regression	60.1	58.9	52.0	58.9
	KNN	56.9	55.9	49.2	55.9
	SVM Classifier	61.5	60.2	53.1	60.2
	Random Forest	59.1	57.0	46.5	57.0
	XGBoost	58.3	57.1	49.7	57.1
	Neural Network	62.0	61.0	**54.8**	61.0
GPT-2	Logistic Regression	62.0	60.6	53.6	60.6
	KNN	55.2	53.4	43.4	53.4
	SVM Classifier	61.6	60.1	52.2	60.1
	Random Forest	58.8	57.0	47.8	57.0
	XGBoost	60.1	58.6	50.6	58.6
	Neural Network	60.3	59.7	**54.6**	59.7

Table 1: Performance metrics for all combinations of embeddings and classification algorithms on a test set

Another metric that can be successfully used to measure the performance of a model on an imbalanced dataset is the F1 Score. The F1 Score is the harmonic mean of precision and recall. It introduces a balance between precision and recall, which is useful when we

want to consider both false positives and false negatives. The F1 Score takes values in the range of 0 to 1, with values closer to 1 indicating a better model. This metric is useful when assigning equal weight to the consequences of false positives and false negatives. Since we are looking for a metric that best compares performance while disregarding the business aspects of the model (such as modifying cutoff points to achieve a specific outcome), we consider it an appropriate metric for comparing models in this study.

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) measures the performance of a classification model. The ROC curve is a graphical representation of the tradeoff between true positive rate and false positive rate for different cutoff points. Similar to the F1 Score, it takes values between 0 and 1, where 1 denotes a perfect model, and 0.5 denotes a model that is no better than random. It is worth noting, however, that the AUC-ROC result is an average value across all threshold levels. Since this study focuses only on the default threshold level of 0.5, we choose the F1 Score as the leading and decisive metric for selecting the best model.

Analyzing the results, we conclude that none of the models proved to be better than the best model presented by Khan et al. (55.7 vs. 62.0 F1 Score). However, the goal of this study was not to build the best-performing model but to compare how different tools affect the model's performance. When analyzing the embedding methods, it is difficult to determine if any of the methods turned out to be the best. To verify the hypothesis that language models outperform the other embedding methods used in this study, we employed the Kruskal-Wallis test. This test compares the distributions of a particular variable in multiple independent groups and determines whether the distributions do not differ between groups. This test is a non-parametric alternative to one-way ANOVA – does not assume data normality and can be used to compare groups of different sizes.

We employ the Kruskal-Wallis test to verify the initial research hypothesis by comparing the F1 Score results for each embedding method. In this manner, we compare 30 observations, with six in each of the five groups. The results of the test indicate that the performance of different embedding models does not differ significantly from each other. The p-value of the Kruskal-Wallis test was 0.98. Therefore, at any significance level there is no basis for rejecting the null hypothesis that the distribution of performance scores of individual embedding methods differ from each other.

The research hypothesis that language models (BERT, RoBERTa, GPT-2) outperform the other embedding methods (Word2Vec, GloVe) was not confirmed.

To verify the second null hypothesis that some classification algorithms perform better than others, we conducted a similar test among different classification methods. We compare the performance, measured by F1 Score, of individual groups of classification methods, which differ from one another. Within each group, we test the F1 Score results for the same classification methods, but using different embedding methods. In total, we examine 30 observations, divided into six groups, each consisting of five instances. The p-value of the Kruskal-Wallis test was <0.001 ($=0.0002$). Therefore, at any sensible significance level the null hypothesis that the distribution of performance scores of individual classification methods is the same has to be rejected.

The research hypothesis that the choice of classification algorithm significantly affects the model's results has been confirmed statistically. However, the Kruskal-Wallis test does not indicate which of the examined groups differs from the others. To determine which group differs from the others, it would be necessary to perform pairwise comparisons between the groups.

In order to achieve this, I perform a post-hoc test for the Kruskal-Wallis test – the Dunn test. This test compares individual groups in pairs and returns p-values for each comparison of means. The Dunn test examines the null hypothesis of equal means. If the

p-value is smaller than the significance level, there is evidence to reject the null hypothesis of the test and conclude that the means in the examined groups differ.

	LR	KNN	SVM	RF	XGB	NN
LR	1.000	0.337	1.000	0.510	1.000	1.000
KNN	0.337	1.000	0.095	1.000	1.000	**0.001**
SVM	1.000	0.095	1.000	0.153	1.000	1.000
RF	0.510	1.000	0.153	1.000	1.000	**0.002**
XGB	1.000	1.000	1.000	1.000	1.000	0.161
NN	1.000	**0.001**	1.000	**0.002**	0.161	1.000

Table 2: Dunn post-hoc pairwise test results

Although the Dunn test results did not indicate the superiority of Neural Network over all classification methods, they suggest that this classification method tends to dominate over the other classification methods.

Indeed, upon analyzing the results table (Table 1), it can be observed that for each embedding method, considering the F1 Score, Neural Network turned out to be the best. The best overall model was the one with GloVe embeddings and Neural Network classification (F1 Score = 55.7).

	Predicted negative	Predicted positive
Actual negative	454 (TN)	260 (FP)
Actual positive	239 (FN)	314 (TP)

Table 3: Confussion matrix for the GloVe + Neural Network model

Upon analyzing the confusion matrix (Table 3), it can be concluded that the best model successfully identified the majority of fake news and true news, exhibiting a relatively high number of true positives (TP) and true negatives (TN), while having a relatively low number of false positives (FP) and false negatives (FN).

Conclusion

The aim of the study was to examine how the application of various tools affects the performance of fake news detection. For this purpose, the LIAR dataset was employed, as it demonstrated greater difficulty for prediction execution compared to other datasets. The fake news detection process consists of two crucial phases. The first one involves obtaining embeddings, while the second phase focuses on applying an appropriate algorithm for classifying fake news. The data, after undergoing prior cleansing and preparation, undergo the embedding generation process. Subsequently, these embeddings serve as inputs to the fake news classification algorithms, distinguishing between fake and genuine information.

In the literature, various approaches to obtaining embeddings and utilizing classification algorithms can be found. However, no comprehensive review has been conducted on how different combinations of embedding methods and classification algorithms impact the model's performance. This study focuses on the most popular and high-performing

models. Regarding the embedding methods, GloVe, Word2Vec, BERT, RoBERTa, and GPT-2 were examined. For the classification algorithms, Logistic Regression, KNN, SVM, Random Forest, XGBoost, and Neural Network were selected. The model utilizing GloVe embeddings and a neural network classifier proved to be the best-performing model in terms of F1 Score (F1 Score = 55.7), outperforming the second-best model that employed RoBERTa embeddings and a neural network classifier (F1 Score = 54.8). The best-performing model achieved a lower performance compared to the benchmark model developed by Khan et al. (F1 Score = 60.2).

The research hypothesis stating that language model-based models outperform other embedding methods included in this study was verified. Additionally, it was investigated whether the choice of embedding method and classification algorithm significantly affects the model's performance.

The research hypotheses were verified using Kruskal-Wallis statistical tests, which demonstrated that the selection of an embedding method does not significantly impact the performance of the fake news detection model (p-value = $0.98 > 0.05$, thereby failing to reject the null hypothesis that the distributions of the model results do not differ significantly from each other). Conversely, the same test revealed that the selection of a classification method does have a significant impact on the model's performance (p-value = $0.0002 < 0.05$). Furthermore, in order to identify which of the examined classification methods exhibit different performance compared to the others, the pairwise Dunn test was employed. It was demonstrated that the average F1 Score results obtained from the Neural Network classification differ significantly from those obtained from KNN (p-value = 0.001) and Random Forest (p-value = 0.002).

In response to the titular question of whether GPT-2 is the best tool for fake news detection, the study did not provide sufficient evidence to indicate the existence of embedding methods demonstrating superior performance over others among those examined.

As a future research direction, to obtain more robust results, a similar study could be conducted on a larger number of datasets simultaneously. This would minimize the risk of results being influenced by the specific characteristics of a given dataset. Furthermore, improvements could be made to the classification methods. In this study, a classical neural network was employed. In addition to deep learning methods, another frequently mentioned classification method yielding highly accurate results is the Naive Bayes method. Given that the hypothesis that classification methods significantly impact model outcomes has been confirmed in this study, examining additional classification algorithms can greatly influence the results of future extensions of this research.

References

Albrecht, J., S. Ramachandran, and C. Winkler (2020). *Blueprints for Text Analytics Using Python: Machine Learning-based Solutions for Common Real World (NLP) Applications*. O'Reilly Media, Incorporated. ISBN: 9781492074083. URL: https://books.google.pl/books?id=M8KLzQEACAAJ.

Aljawarneh, Shadi A. and Safa Ahmad Swedat (2022). "Fake News Detection Using Enhanced BERT". In: *IEEE Transactions on Computational Social Systems*, pp. 1–8. DOI: 10.1109/TCSS.2022.3223786.

Alonso-Bartolome, Santiago and Isabel Segura-Bedmar (2021). *Multimodal Fake News Detection*. DOI: 10.48550/ARXIV.2112.04831. URL: https://arxiv.org/abs/2112.04831.

Breiman, L (Oct. 2001). "Random Forests". In: *Machine Learning* 45, pp. 5–32. DOI: 10.1023/A:1010950718922.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

Conroy, Nadia K., Victoria L. Rubin, and Yimin Chen (2015). "Automatic deception detection: Methods for finding fake news". In: *Proceedings of the Association for Information Science and Technology* 52.1, pp. 1–4. DOI: https://doi.org/10.1002/pra2.2015.145052010082. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/pra2.2015.145052010082. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/pra2.2015.145052010082.

Cruz, Jan Christian Blaise, Julianne Agatha Tan, and Charibeth Cheng (2019). "Localization of Fake News Detection via Multitask Transfer Learning". In: *CoRR* abs/1910.09295. arXiv: 1910.09295. URL: http://arxiv.org/abs/1910.09295.

Devlin, Jacob et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

Fix, Evelyn and J. L. Hodges (1989). "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties". In: *International Statistical Review / Revue Internationale de Statistique* 57.3, pp. 238–247. ISSN: 03067734, 17515823. URL: http://www.jstor.org/stable/1403797 (visited on 06/09/2023).

Goel, Pratyush et al. (2021). "Multi Domain Fake News Analysis using Transfer Learning". In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1230–1237.

Gundapu, Sunil and Radhika Mamidi (2021). *Transformer based Automatic COVID-19 Fake News Detection System*. DOI: 10.48550/ARXIV.2101.00180. URL: https://arxiv.org/abs/2101.00180.

Joy, Sajib Kumar Saha et al. (2022). *A Comparative Study on COVID-19 Fake News Detection Using Different Transformer Based Models*. DOI: 10.48550/ARXIV.2208.01355. URL: https://arxiv.org/abs/2208.01355.

Jwa, Heejung et al. (2019). "exBAKE: Automatic Fake News Detection Model Based on Bidirectional Encoder Representations from Transformers (BERT)". In: *Applied Sciences* 9.19. ISSN: 2076-3417. DOI: 10.3390/app9194062. URL: https://www.mdpi.com/2076-3417/9/19/4062.

Kaliyar, Rohit, Anurag Goswami, and Pratik Narang (Mar. 2021). "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach". In: *Multimedia Tools and Applications* 80. DOI: 10.1007/s11042-020-10183-2.

Khan, Junaed Younus et al. (2021). "A benchmark study of machine learning models for online fake news detection". In: *Machine Learning with Applications* 4, p. 100032. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100032. URL: https://www.sciencedirect.com/science/article/pii/S266682702100013X.

Liu, Yinhan et al. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

Mikolov, Tomas et al. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. arXiv: 1310.4546 [cs.CL].

Mridha, M. F. et al. (2021). "A Comprehensive Review on Fake News Detection With Deep Learning". In: *IEEE Access* 9, pp. 156151–156170. DOI: 10.1109/ACCESS.2021.3129329.

Pathak, Ajeet et al. (Jan. 2020). "Analysis of Techniques for Rumor Detection in Social Media". In: *Procedia Computer Science* 167, pp. 2286–2296. DOI: 10.1016/j.procs.2020.03.281.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

Peters, Matthew E., Sebastian Ruder, and Noah A. Smith (2019). "To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks". In: *CoRR* abs/1903.05987. arXiv: 1903.05987. URL: http://arxiv.org/abs/1903.05987.

Rai, Nishant et al. (2022). "Fake News Classification using transformer based enhanced LSTM and BERT". In: *International Journal of Cognitive Computing in Engineering* 3, pp. 98–105. ISSN: 2666-3074. DOI: https://doi.org/10.1016/j.ijcce.2022.03.003. URL: https://www.sciencedirect.com/science/article/pii/S2666307422000092.

Raschka, Sebastian and Vahid Mirjalili (2019). *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd.

Rashkin, Hannah et al. (Sept. 2017). "Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 2931–2937. DOI: 10.18653/v1/D17-1317. URL: https://aclanthology.org/D17-1317.

Rubin, Victoria et al. (June 2016). "Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News". In: *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*. San Diego, California: Association for Computational Linguistics, pp. 7–17. DOI: 10.18653/v1/W16-0802. URL: https://aclanthology.org/W16-0802.

Sanh, Victor et al. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. DOI: 10.48550/ARXIV.1910.01108. URL: https://arxiv.org/abs/1910.01108.

Shifath, S. M. Sadiq-Ur-Rahman, Mohammad Faiyaz Khan, and Md. Saiful Islam (2021). *A transformer based approach for fighting COVID-19 fake news*. arXiv: 2101.12027 [cs.CL].

Shu, Kai et al. (Sept. 2017). "Fake News Detection on Social Media: A Data Mining Perspective". In: *SIGKDD Explor. Newsl.* 19.1, pp. 22–36. ISSN: 1931-0145. DOI: 10.1145/3137597.3137600. URL: https://doi.org/10.1145/3137597.3137600.

Singhal, Shivangi et al. (2019). "SpotFake: A Multi-modal Framework for Fake News Detection". In: *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pp. 39–47. DOI: 10.1109/BigMM.2019.00-44.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288. ISSN: 00359246. URL: http://www.jstor.org/stable/2346178 (visited on 06/09/2023).

Vapnik, Vladimir (2000). *The Nature of Statistical Learning Theory*. Springer: New York.

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Wager, Stefan, Sida Wang, and Percy Liang (2013). *Dropout Training as Adaptive Regularization*. arXiv: 1307.1493 [stat.ML].

Wang, Sida and Christopher Manning (June 2013). "Fast dropout training". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 2. Atlanta, Georgia, USA: PMLR, pp. 118–126. URL: https://proceedings.mlr.press/v28/wang13a.html.

Wang, William Yang (July 2017). ""Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 422–426. DOI: 10.18653/v1/P17-2067. URL: https://aclanthology.org/P17-2067.

Wang, Yuxiang et al. (2021). *COVID-19 Fake News Detection Using Bidirectional Encoder Representations from Transformers Based Models*. DOI: 10.48550/ARXIV.2109.14816. URL: https://arxiv.org/abs/2109.14816.

Yang, Kai-Chou, Timothy Niven, and Hung-Yu Kao (2019). *Fake News Detection as Natural Language Inference*. DOI: 10.48550/ARXIV.1907.07347. URL: https://arxiv.org/abs/1907.07347.

A Text pre-processing

I conducted an exploration and preprocessing of textual data, dividing it into a training set and a test set. I examined basic descriptive statistics regarding the length of the news articles. The average text length is approximately 80. Less than 25% of the texts are longer than 100. This is significant because the maximum text length that can be used as input for BERT is 512. Therefore, text lengths should not pose a problem. In the case of texts longer than 512, they are truncated to a length of 512, and the remaining words are disregarded. It is also worth noting that the training set and the test set exhibit very similar statistics.

Statistics	Train set	Test set
Count	11524	1267
Mean	81	83
Std	47	83
Min	8	9
25%	56	57
50%	75	75
75%	100	99
Max	2711	2560

Table 4: Text length exploration

Before merging multiple classes into two classes (for fake news and true news), I examined the cardinality of each individual class in order to combine them in a manner that would create a balanced binary class.

Class	Size
half-true	2362
false	2258
mostly-true	2213
barely-true	1891
true	1845
pants-fire	955

Table 5: Class size before combining into two classes

After the merger, the balance of both classes is represented as follows. The dataset balance is highly similar between the training and testing sets.

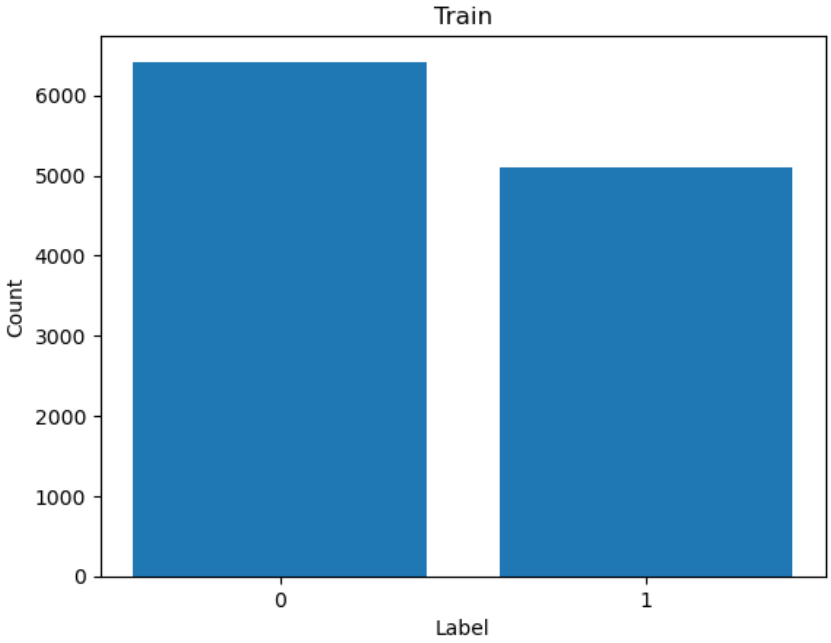

Figure 6: Train set class balance

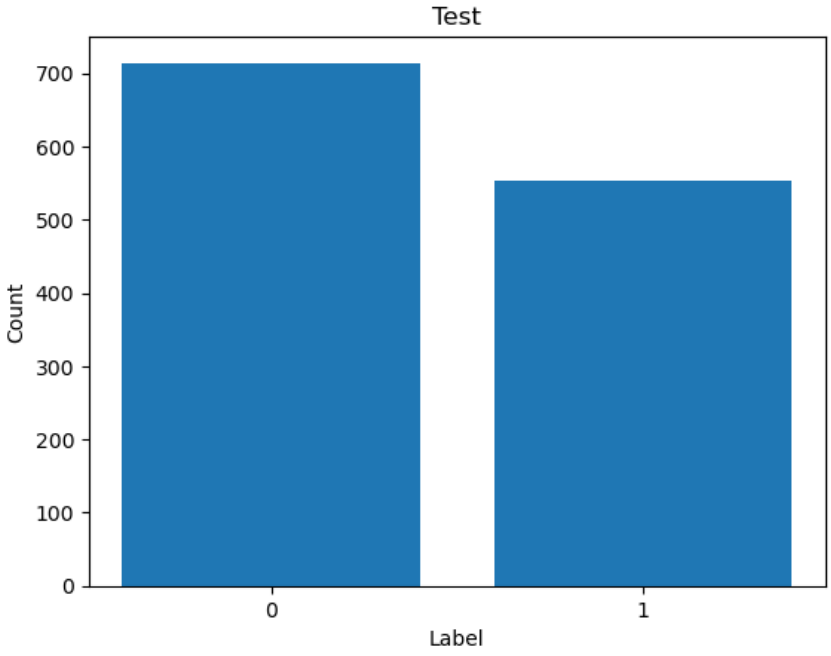

Figure 7: Test set class balance

Below are the wordclouds illustrating the frequency of words in the training and testing datasets.

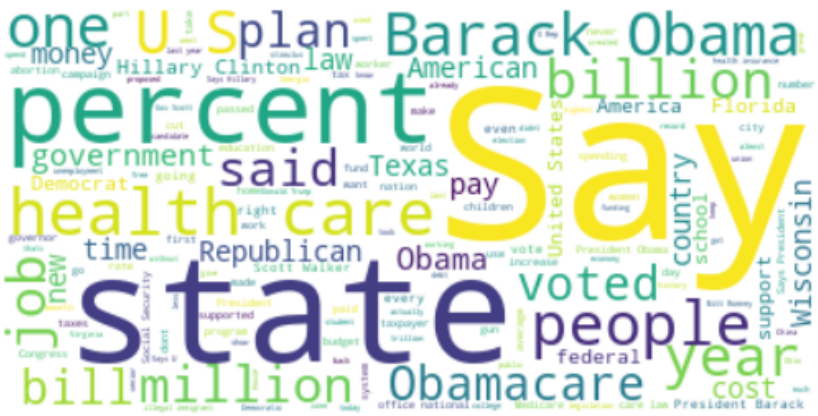

Figure 8: Cloud of words of fake news in training set

Figure 9: Cloud of words of true news for training set

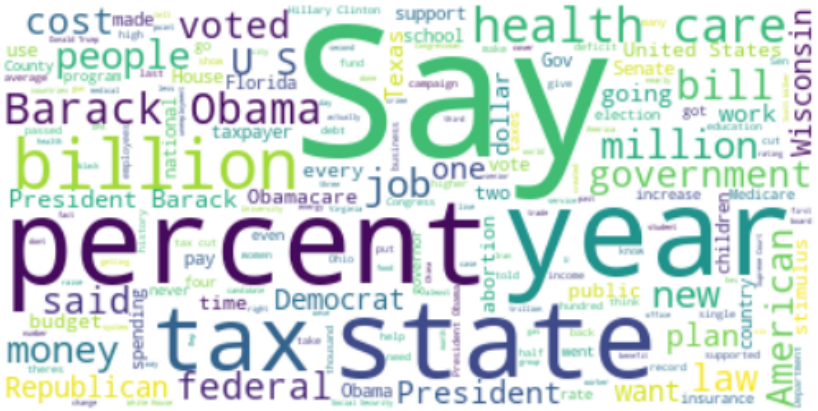

Figure 10: Cloud of words of fake news for testing set

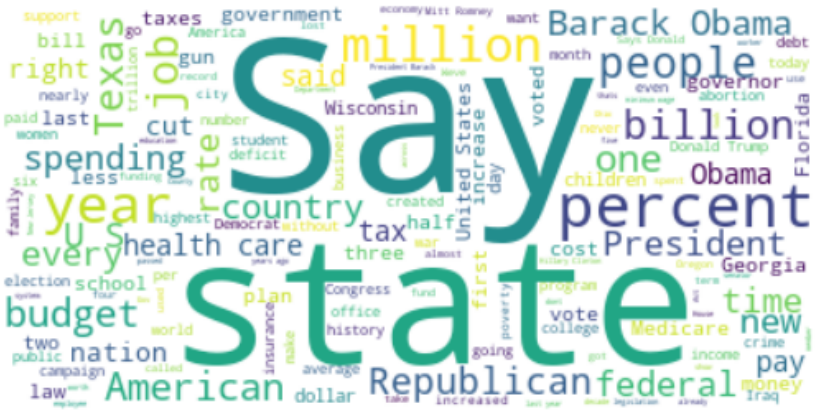

Figure 11: Cloud of words of true news for testing set

B Extended results

Below, I present the Confusion Matrices for all models.

	Predicted negative	Predicted positive
Actual negative	516 (TN)	198 (FP)
Actual positive	307 (FN)	246 (TP)

Table 6: Confussion matrix for the GloVe + Logistic Regression model

	Predicted negative	Predicted positive
Actual negative	465 (TN)	249 (FP)
Actual positive	317 (FN)	236 (TP)

Table 7: Confussion matrix for the GloVe + KNN model

	Predicted negative	Predicted positive
Actual negative	488 (TN)	226 (FP)
Actual positive	291 (FN)	262 (TP)

Table 8: Confussion matrix for the GloVe + SVM model

	Predicted negative	Predicted positive
Actual negative	508 (TN)	206 (FP)
Actual positive	301 (FN)	252 (TP)

Table 9: Confussion matrix for the GloVe + Random Forest model

	Predicted negative	Predicted positive
Actual negative	533 (TN)	181 (FP)
Actual positive	310 (FN)	243 (TP)

Table 10: Confussion matrix for the GloVe + XGBoost model

	Predicted negative	Predicted positive
Actual negative	540 (TN)	174 (FP)
Actual positive	311 (FN)	242 (TP)

Table 11: Confussion matrix for the Word2Vec + Logistic Regression model

	Predicted negative	Predicted positive
Actual negative	416 (TN)	298 (FP)
Actual positive	285 (FN)	268 (TP)

Table 12: Confussion matrix for the Word2Vec + KNN model

	Predicted negative	Predicted positive
Actual negative	483 (TN)	231 (FP)
Actual positive	270 (FN)	283 (TP)

Table 13: Confussion matrix for the Word2Vec + SVM model

	Predicted negative	Predicted positive
Actual negative	500 (TN)	214 (FP)
Actual positive	320 (FN)	233 (TP)

Table 14: Confussion matrix for the Word2Vec + Random Forest model

	Predicted negative	Predicted positive
Actual negative	466 (TN)	248 (FP)
Actual positive	295 (FN)	258 (TP)

Table 15: Confussion matrix for the Word2Vec + XGBoost model

	Predicted negative	Predicted positive
Actual negative	468 (TN)	246 (FP)
Actual positive	253 (FN)	300 (TP)

Table 16: Confussion matrix for the Word2Vec + Neural Network model

	Predicted negative	Predicted positive
Actual negative	508 (TN)	206 (FP)
Actual positive	295 (FN)	258 (TP)

Table 17: Confussion matrix for the BERT + Logistic Regression model

	Predicted negative	Predicted positive
Actual negative	447 (TN)	267 (FP)
Actual positive	292 (FN)	261 (TP)

Table 18: Confussion matrix for the BERT + KNN model

	Predicted negative	Predicted positive
Actual negative	476 (TN)	238 (FP)
Actual positive	287 (FN)	266 (TP)

Table 19: Confussion matrix for the BERT + SVM model

	Predicted negative	Predicted positive
Actual negative	504 (TN)	210 (FP)
Actual positive	327 (FN)	226 (TP)

Table 20: Confussion matrix for the BERT + Random Forest model

	Predicted negative	Predicted positive
Actual negative	495 (TN)	219 (FP)
Actual positive	295 (FN)	258 (TP)

Table 21: Confussion matrix for the BERT + XGBoost model

	Predicted negative	Predicted positive
Actual negative	426 (TN)	288 (FP)
Actual positive	244 (FN)	309 (TP)

Table 22: Confussion matrix for the BERT + Neural Network model

	Predicted negative	Predicted positive
Actual negative	487 (TN)	227 (FP)
Actual positive	279 (FN)	274 (TP)

Table 23: Confussion matrix for the RoBERTa + Logistic Regression model

	Predicted negative	Predicted positive
Actual negative	457 (TN)	257 (FP)
Actual positive	289 (FN)	264 (TP)

Table 24: Confussion matrix for the RoBERTa + KNN model

	Predicted negative	Predicted positive
Actual negative	503 (TN)	211 (FP)
Actual positive	277 (FN)	276 (TP)

Table 25: Confussion matrix for the RoBERTa + SVM model

	Predicted negative	Predicted positive
Actual negative	524 (TN)	190 (FP)
Actual positive	328 (FN)	225 (TP)

Table 26: Confussion matrix for the RoBERTa + Random Forest model

	Predicted negative	Predicted positive
Actual negative	478 (TN)	236 (FP)
Actual positive	292 (FN)	261 (TP)

Table 27: Confussion matrix for the RoBERTa + XGBoost model

	Predicted negative	Predicted positive
Actual negative	495 (TN)	219 (FP)
Actual positive	262 (FN)	291 (TP)

Table 28: Confussion matrix for the RoBERTa + Neural Network model

	Predicted negative	Predicted positive
Actual negative	507 (TN)	207 (FP)
Actual positive	275 (FN)	278 (TP)

Table 29: Confussion matrix for the GPT-2 + Logistic Regression model

	Predicted negative	Predicted positive
Actual negative	483 (TN)	231 (FP)
Actual positive	336 (FN)	217 (TP)

Table 30: Confussion matrix for the GPT-2 + KNN model

	Predicted negative	Predicted positive
Actual negative	516 (TN)	198 (FP)
Actual positive	288 (FN)	265 (TP)

Table 31: Confussion matrix for the GPT-2 + SVM model

	Predicted negative	Predicted positive
Actual negative	506 (TN)	208 (FP)
Actual positive	314 (FN)	239 (TP)

Table 32: Confussion matrix for the GPT-2 + Random Forest model

	Predicted negative	Predicted positive
Actual negative	503 (TN)	211 (FP)
Actual positive	294 (FN)	259 (TP)

Table 33: Confussion matrix for the GPT-2 + XGBoost model

	Predicted negative	Predicted positive
Actual negative	462 (TN)	252 (FP)
Actual positive	251 (FN)	302 (TP)

Table 34: Confussion matrix for the GPT-2 + Neural Network model

List of Figures

List of Tables