FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Classification of Fake News in the Media/Social media Ecosystem
Student:	Jan Flajžík
Supervisor:	Ing. Mgr. Ladislava Smítková Janků, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 3 December 2022 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Instructions

This thesis is focused on the design and implementation of machine learning methods for the detection of selected fake news.
1) Prepare a survey of fake news classification. Describe several existing fake news datasets.
2) Prepare an experimental dataset containing examples of fake news. Extract it from the existing fake news databases.
3) Design and perform experiments using existing tools or libraries for fake news classification.
4) Evaluate classification performance and discuss results.

Datasets:
1. Eldariel – public version of the disinformation chain email database, https://eldariel.cesti-elfove.cz/
2. EU vs DiSiNFO database of fake news, https://euvsdisinfo.eu/disinformation-cases/
3. ISOT Fake News Dataset, University of Victoria https://www.uvic.ca/ecs/ece/isot/datasets/fake-news/index.php

Literature:
1. D'Ulizia A, Caschera MC, Ferri F, Grifoni P: Fake news detection: a survey of evaluation datasets. PeerJ Comput Sci. 2021
2. Zhou, X & Zafarani: A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities, ACM Computing Surveys, vol. 53. 2020

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 3 December 2022 in Prague.

Bachelor's thesis

CLASSIFICATION OF FAKE NEWS IN THE MEDIA/SOCIAL MEDIA ECOSYSTEM

Jan Flajžík

Faculty of Information Technology
Katedra aplikované matematiky
Supervisor: Ing. Mgr. Ladislava Smítková Janků, Ph.D.
May 16, 2024

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Flajžík Jan. *Classification of Fake News in the Media/Social media Ecosystem.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

List of Figures

List of Tables

I would like to express my gratitude to my supervisor, Ing. Mgr. Ladislava Smítková Janků, Ph.D., for providing me with constructive feedback throughout the writing of this thesis. Furthermore, I would like to extend my thanks to the faculty and staff of CTU in Prague, for providing a rich learning environment. Finally, I would like to thank my family and friends for their endless patience and support.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 16, 2024 .....

Abstract

The rapid development of online technologies allows people to spread information faster than ever before. This carries the risk of spreading manipulative information that can negatively impact society. The focus of this thesis is the classification of fake news in the media ecosystem from the perspective of natural language processing (NLP). Two brand-new fake news datasets were created using the online fake news databases, with the English one made up of short summaries of fake news articles and the Czech one consisting mainly of chain letters. These two datasets and two more existing datasets were used for performing experiments with many text preprocessing methods. Two machine learning classifiers were used for experiments: Naive Bayes and random forest, and two neural network architectures: convolutional neural network (CNN) and LSTM. The CNN attained the highest classification accuracy of 97% on an already-made dataset, whereas the best results on a self-obtained dataset attained the LSTM with 95% accuracy.

Keywords fake news, disinformation, social networks, chain letters, natural language processing, text classification, machine learning, neural networks

Abstrakt

Rychlý vývoj v oblasti online technologií umožňuje lidem sdílet informace rychleji než kdykoli před tím. To s sebou nese riziko šíření manipulativních zpráv, které mohou negativně ovlivnit společnost. Tato bakalářská práce se zabývá klasifikací fake news v online prostoru z pohledu zpracování přirozeného jazyka (NLP). V rámci této práce byly vytvořeny dva nové datasety za použití online databází fake news, přičemž jeden obsahuje shrnutí fake news článků v angličtině a druhý je tvořen především řetězovými emaily v češtině. Tyto datasety byly doplněny o dva již existující datasety a všechny byly použity k provedení expermerimentů s mnoha různými metodami předzpracování textu. Ke klasifikaci byly využity dva modely strojového učení, Naivní Bayes a náhodný les a dvě architektury neuronové sítě, konvoluční neurovová sít (CNN) a LSTM. Nejvyšších výsledků na již existujícím datasetu dosáhla CNN architektura s klasifikační přesnosti 97 %, zatímco nejvyšších výsledků na nově vytvořeném dosáhla LSTM architektura s přesností 95 %.

Klíčová slova fake news, dezinformace, sociální sítě, řetězové emaily, zpracování přirozeného jazyka, klasifikace textu, strojové učení, neuronové sítě

List of abbreviations

BoW	Bag-of-words
BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-words Model
CNN	Convolutional neural network
ELECTRA	Encoder that Classifies Token Re-placements Accurate
ELMO	embeddings from language model
EU	European Union
GloVe	Global vectors for word representation
IDF	Inverse document frequency
k-NN	k-nearest neighbors
LSTM	Long short-term memory
MLP	Mulitlayer perceptron
NLP	Natural language processing
NN	Neural network
no	Number
SGD	Stochastic gradient descent
TF	Term frequency
TF-IDF	Term frequency-inverse document frequency

Chapter 1

Introduction

False, misleading, and manipulative information has been around since the dawn of time, but with the rise of the internet, especially social media platforms, the spread of information has become easier than ever before.

The term fake news started to circulate during the 2016 US presidential election. In that year, BuzzFeed[1] carried out a survey of 3015 US adults regarding fake news stories related to the election. In that survey, social media, especially Facebook, turned out to be the third most popular major source of news. More than 80 % of respondents who cited Facebook as a major source of information claimed that they were familiar with the fake news headline and found the presented fabricated story "somewhat" or "very" accurate [1].

Since then, social media platforms, in cooperation with governments, have started to employ strategies to combat this phenomenon. Governments tend to spread awareness of the issue, whereas social media platforms tag and delete posts that are regarded as fake news. Due to the large amount of such news, the process must involve machine learning methods for automatic text classification. Different outlets use different methods, while the effort is to be as precise as possible.

1.1 Goals

The main goal of this thesis is to analyse, implement and compare several existing machine learning approaches for fake news classification using natural language processing or NLP. This includes obtaining a brand-new fake news dataset using public fake news databases in both the Czech and English. Since spreading fake news has become easier than ever before due to internet accessibility, the areas of the fake news datasets used in this thesis will vary. This goal must be divided into several following subtasks:

- research the existing literature related to fake news,
- research state-of-the-art of fake news classification,
- study machine learning classifiers and implement the selected ones,
- provide and describe several existing fake news datasets,
- obtain a brand-new dataset from online fake news databases,
- perform and evaluate classification experiments on the selected datasets and discuss the results.

[1] `https://www.buzzfeed.com/`

1.2 Bachelor thesis structure

This thesis is organised as follows.

In *Chapter 2*, we present a survey of related academic literature regarding the fake news classification. We start by defining the terms relevant to the issue, which we will use throughout the text. Then we survey state-of-the-art fake news classification in the academic sphere. We continue with a description of multiple machine learning models used for the text classification task, including approaches for natural language processing. We conclude the chapter with a theoretical overview of selected classification models and a description of related evaluation metrics.

In *Chapter 3*, we provide a description of multiple existing fake news datasets. This is followed by an overview of selected fake news databases and tools used for obtaining the brand new ones. Finally, we present the self-obtained ones.

In *Chapter 4*, we describe the existing tools and libraries necessary for performing experiments on the datasets described in the previous chapter. This chapter concludes with a description of the experiments and discussion of the results.

Chapter 2
Related work

In this section, we survey academic research regarding fake news automatic detection. First, we define topic-related terms for a better understanding of the context of this thesis. Then we look at the current state of research in this area and describe several existing approaches regarding fake news detection. Finally, we look at the tools for text processing and describe selected classification models and evaluation metrics.

2.1 Definition of basic terms

Firstly, we must define frequently used terms in this thesis. There is no globally agreed definition of any of the terms, so other scientific literature might work with different ones. However, in the following text, they will be used in thus-defined meanings.

Fake news can be defined as a news article that is *"intentionally and verifiably false"* [2]. The European Union (EU) uses a similar but broader definition for the term **disinformation**. According to the EU, it is a *"false or misleading content that is spread to deceive or secure economic or political gain, and which may cause public harm"* [3]. These two terms are often used interchangeably, although they are not entirely identical. The term fake news often refers to a single piece of content, whereas disinformation is a broader term for the phenomenon. The latter term is often confused with the term **misinformation**, which is incorrect because this noun refers to spreading fabricated stories unconsciously without any harmful intentions [4, 5]. Misinformation classification is not a topic of this thesis.

The methods of spreading fake news have changed throughout history. The invention of letterpress printing in the 15th century was a significant event for this phenomenon, but it cannot be compared to the development of internet services in the past 30 years. For a better understanding of how fake news spreads in the online ecosystem, we must also take a look at the two most popular environments of online communication.

Social media refers to *"platforms such as Twitter and Facebook that help individuals from around the globe build networks and share information and/or sentiments in real-time"* [6]. The posts on social media regarded as fake news do not only include articles from disinformation outlets but also visual files such as images of videos that can also bear the manipulative narrative [5], but we are not going to work with such additional media in this thesis.

A **chain letter** is a letter usually sent via email or social media platform whose purpose is to capture the recipient's interest by providing some shocking piece of information and make the reader pass it on to as many people as possible by threat or promise. The reader's attention is usually captured by some kind of hook phrase, such as "THIS INFORMATION IS VITAL." Although chain letters are typically associated with advertisements, they are a widely used platform for spreading fake news [7].

2.2 Current state of research in fake news detection

Fake news is not spread in only one medium. It is usually a bigger system that contains text, images, and sometimes even videos or audio files. There are many possible approaches to classifying those pieces of content, and all of them can be due to the development of technology done automatically. Analysing the content itself is one of many existing perspectives. One viable alternative is, for example, analysing the social context of the news (see section 2.2.2).

In this thesis, we employ the content-based approach by examining the linguistic features of fake news. In other words, we address this task as a text classification problem.

2.2.1 Language processing approaches

Even though the research into text classification and language processing has been around for a long period of time, the interest in research on fake news detection has increased with the rise of social media and especially with the pandemic of COVID-19 [6]. The fake news detection problem is aimed to count the probability that a news article is intentionally fake which is desired to be done automatically due to economic factors [8].

Since fake news has become a trending topic in recent years, the academic community has started to study this phenomenon and ways of combating it more closely. This resulted in hundreds of articles published every year. In their meta-analysis of studies from 2022, Thompson et al. [6] claim that over 2000 articles were published between 2014 and 2022 regarding the topic of fake news classification. In the following text, we only mention a couple of examples of published articles related to this topic.

Shu et al. in paper [9] from 2017 defined fake news classification as a problem of distortion bias on information, which is usually modelled as a binary classification problem. In this article, the authors also argued for using linguistic-based features for fake news detection as one of the methods of extracting features of fake news. For example, lexical features such as total words or characters per word or syntactic features, for instance, frequency of function words.

Modelling this task as a binary classification problem is not the only viable option. Since every text can always be at least somewhat true, the authors of other articles, such as a proposal of a special neural network architecture for fake news detection [10] by Islam et al., worked with three or more classes based on how much fake news the concrete articles are.

As an example of an article where the authors utilised machine learning approaches, we present a paper by Gilda [11] in which the author performed an evaluation of multiple classification algorithms for fake news detection using linguistic features. They used many algorithms, such as Support Vector Machines, Stochastic Gradient Descent, Gradient Boosting, Bounded Decision Trees and Random Forests on a cleaned dataset of 11,000 news articles from both verified credible and verified non-credible sources originally which was analysed by Corney et al. [12]. The experiments reached the best performance with the TF-IDF (term frequency-inverse document frequency) of bi-grams fed into the Stochastic Gradient Descent algorithm with 77.2 % accuracy, which was a surprisingly poor result.

An example of an article with better results that employed machine learning models is a paper by Ahmed et al. in which the authors analysed the impact of sizes of n-grams on classification performance [13]. In this article, authors attained up to 92 % accuracy using a Linear Support vector machine with TF-IDF feature extraction on their own dataset based on articles from Reuters[1] and Kaggle[2].

Deep learning methods and masked language models started to be employed around the year 2020. As an example of this solution, we present the benchmark study by Khan et al. [14] published in 2021, where authors took a closer look at comparing traditional machine learning

[1] www.reuters.com
[2] www.kaggle.com

approaches with deep learning classifiers based on neural networks and masked language models. The proposed methods were presented by other authors between the years 2015 and 2020. The authors of this benchmark study compared the presented models using three datasets: "Liar", "Fake or real news" and "Combined corpus". Liar is a benchmark dataset presented originally in an article by Wang [15]. This dataset is among the most popular in fake news detection research and will be described in Chapter 3. The second dataset "Fake or real news" of 6335 articles which were partly obtained from Kaggle and serious media outlets including The Wall Street Journal[3], Bloomberg[4], The Guardian[5], etc. The third one called "Combined corpus" was obtained from different sources using empirical analysis of inner-topic distances. Authors divided the utilised classification models into 3 categories:

- traditional machine learning including k-nearest neighbours (k-NN), decision tree, Naive Bayes, AdaBoost, support vector machine (SVM),
- deep learning models including convolutional neural networks, long short-term memory networks, bidirectional-long short-term memory, C-Long short term memory (C-LSTM), hierarchical attention network (HAN), and Convolutional hierarchical attention network,
- advanced masked language models including Bidirectional Encoder Representations from Transformers (BERT) and its variations RoBERTa, DistilBERT, efficiently Learning an Encoder that Classifies Token Re-placements Accurate (ELECTRA) and Embeddings from Language Model (ELMo).

They then created a performance measurement where the advanced masked language model RoBERT (Robustly optimized BERT) outperformed all other models on a dataset of "Combined corpus", with both accuracy and F1 score reaching 96 %. On the other hand, the performance of the other two categories of models did not lag. The best results from the machine learning group reached Naive Bayes with bigram TF-IDF feature extraction with both accuracy and F1 score of 93 % and Bi-LSTM and C-LSTM from the deep learning category reaching within the same metrics up to 95 %.

According to the earlier mentioned meta-analysis of studies [6], today's most widely used approach is based on deep learning using all types of neural networks, with the architecture based on the convolutional neural network being the most common. Also, there is no one specific architecture that is recommended for this task. Instead, many different architectures have been proposed that are capable of reaching promising results. On the other hand, traditional machine learning approaches are still commonly used for this problem, with random forest being the most popular one.

2.2.2 Other possible approaches

Examining linguistic features is not the only existing approach for fake news classification [2]. One of the possible different approaches to fake news classification is so-called stance detection. This approach also employs natural language processing techniques, but instead of viewing the problem as a binary classification task, it finds the relationship between two fragments of text (headline and article). The stance between those two contents is then described by these four labels: Agree, Disagree, Discuss, and Unrelated, which define their relationship [16].

As an example of a response-based approach for fake news detection, we present a paper by Kidu et al. [17] published in 2021 in which the authors present an approach for fake news detection by *"investigating the reaction of users to a post composed by malicious authors."* The authors apply traditional machine learning techniques such as random forest and logistic regression, but also recurrent neural networks on comments on social media in combination with

[3] www.wsj.com
[4] www.bloomberg.com
[5] www.theguardian.com

encoding emotional responses as categorical data, which *"increased the performance of all six algorithms."* Interestingly, this approach attains an accuracy of 97 % and F1 score of 98 %. However, focusing on this approach or any other approach is not the goal of this thesis, and it would be impossible to examine all of them properly.

2.3 Text classification methods

As we described in the section *Language processing approaches*, in the current state-of-the-art of fake news detection using linguistic analysis, both traditional machine learning and deep learning models are widely applied. Before anything else, we must take a look at what sort of tasks we are solving during the fake news detection.

Classification task is a problem in which a model tries to predict one of the labels of a given data point based on additional data. The difference between classification and regression tasks is that the classification works only with a limited and usually very small number of possible labels. In this thesis, we will only work with two-label classification. Both deep and machine learning models can be employed in the classification task which falls into a field of artificial intelligence called supervised learning.

Supervised learning is a broader term for tasks in which models find the relations between the desired solution (dependent variables), such as a label or numeric value, and other given data (independent variables). The models learn (shape their parameters) based on those relations and afterwards predict the outcome for newly presented data with as little error as possible [18, 19].

However, most classification models are designed to work with numerical and categorical data, but in this case, the input data are in the form of unstructured text, which causes them multiple problems [20]. In the next section, we are going to look at possible ways of preprocessing raw text and extracting its key features into a form processable by a classifier.

2.3.1 Natural Language Processing

Natural language processing can be defined as a system that analyses, attempts to understand, or produces human languages [21]. In the context of this thesis, it is a set of fundamental operations whose main goal is to preprocess raw unstructured text (news article) into a clean and normalised form, which should lead to an improvement in the feature extraction phase [20] that is described further. Before anything else, we are going to look at commonly used text preprocessing techniques.

2.3.1.1 Special characters removal

News articles, both real and fake, contain multiple punctuation, numerals, and special characters that do not have any effect on whether the news is real or fake. These characters can have an influence on how humans understand the text, but in most circumstances, they have no bearing on the text categorization [10]. Apart from the special characters, fake news articles may contain links to either other fake articles or even cite credible sources [22]. The links are, in this process, usually removed as well.

2.3.1.2 Capitalisation

Natural languages in general have predefined rules for capitalising parts of sentences, e.g., English sentences must always start with the first capital letter. Even though the machine learning models could work with the texts in their original form, it is preferable to convert all texts to single letter case [10]. It is worth noting that changing the letter case of certain words and abbreviations could tweak their original meanings. For example, "US" could refer to the United States of

America or a first-person plural pronoun [23]. In the experimental part of this thesis, we apply lowercase to the whole text except for some specific words.

2.3.1.3 Tokenization

Tokenization can be defined as a process of *"breaking a stream of text up into phrases, words, symbols, or other meaningful elements called tokens"* [24]. The most common way to tokenize a sentence is to create a list of single words using a space character as delimiter [25]. However, the text can also be processed into n-grams which are sequences of items of length n. These items can be anything from single characters to full words. The most popular forms of n-grams are word-based and character-based [13]. In this thesis, we work with the word-based type to generate features.

2.3.1.4 Stop words removal

Stop words are the parts of sentences that do not add any additional meaning to the sentence and thus can be easily dismissed. This often includes the most common words, such as "the," "and," "you," etc. Stop words can be obtained from the text itself by finding the words with the highest occurrence in the whole dataset or from already-made lists called corpora, which are available for most world languages [26, 24].

2.3.1.5 Stemming

Words with identical meanings might vary in form. The goal of stemming is to map various morphological versions of a word to its base forms. Stemming reduces the number of words in the corpus and can help match the ones with the same meaning. This is accomplished by reducing the original word to the stem word. For example, the stem word for the terms "change" and "changing" would be "chang." There are multiple approaches to executing stemming, but they are not effective every time [27].

2.3.1.6 Lemmatization

Lemmatization is the process of adjusting the inflected parts of the words and returning them to their vocabulary form, the so-called lemma. The lemma is the base of all inflected parts. This allows matching the words that are in different forms as well as synonyms using thesaurus, vocabulary, and recognising parts of speech. The process is similar to stemming but adds meaning to the word, thus making the algorithm more complex and harder to implement[27].

2.3.2 Feature extraction

When the original text is preprocessed, we need to convert the result into a suitable form for classification models—vectors of numbers. There are multiple ways to do that, and the most popular ones are described in the following sections. The result of these operations is then used for training the classifiers and their evaluation.

2.3.2.1 Bag-of-words

Bag-of-words (BoW) is one of the most fundamental methods to transform words into a set of features. The BoW technique goes through all the tokens in a sentence or article and calculates the word occurrences for each token, disregarding the order of the words or grammar. The result is a structure where each word matches its number of appearances. For example, the BoW representation of the sentence "Dara likes to go to cinema." looks like this: $\{Dara : 1, likes : 1, to : 2, go : 1, cinema : 1\}$. The final BoW representation of the dataset is a document-term

matrix, where each ij cell represents the number of occurrences of word j in document i. One of the problems with this method is that it computes the number of occurrences for each word, so frequent terms like "a" or "the" usually have the maximum number of occurrences. Thus, it is important to remove stop words before creating a BoW. Another problem with the BoW model is that the document-term matrix is very sparse and most of the elements are zeros [28, 29].

2.3.2.2 TF-IDF

Term frequency-inverse document frequency (TF-IDF) is another fundamental method for text feature extraction using numerical statistics that show the relevance of keywords to a document. TF-IDF, as we can see from its name, can be broken down into two terms:

- Term frequency (TF) refers to the frequency of a term t in some concrete document d. The frequency might be computed as a raw count or by other possible techniques, such as relative to other terms in the document.
- Inverse document frequency (IDF), on the other hand, works with the importance of each word for a set of documents. Stop words such as "the" would always reach the highest significance regarding TF. IDF, on the contrary, assigns higher significance to infrequent words and lowers the significance of stop words. It measures the amount of information provided in the word. If D signifies all of the documents and d is one concrete document, then the IDF of term t is usually calculated with the formula:

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}. \tag{2.1}$$

Both metrics disregard the order and grammar of the words in the document. The final TF-IDF of each term is computed as a product of its TF and IDF. The dataset is finally represented as a document-term matrix, where each cell represents the TF-IDF of each word in each document, similarly to BoW. [29, 30].

2.3.2.3 Word2Vec

Word2Vec is one of many available word embedding algorithms. These algorithms can be defined as projections of words into a vector space that preserve semantic and syntactic similarities [31]. The difference between word embedding and feature extraction algorithms described earlier is that word embedding algorithms convert each word to its high-dimensional (usually tens or hundreds) vector representation instead of the whole dataset. Word embeddings work on the hypothesis that words with similar meanings tend to be used in the same context [32].

Word2Vec is a group of models designed to create word embeddings based on semantic and syntactic context. This model consists of two neural network sub-models: Continuous bag-of-words model (CBOW) and Skip-gram. Both models are modelled as neural networks and trained using stochastic gradient descent and backpropagation algorithms [33].

The purpose of CBOW is to predict a target word based on its context. It takes one-hot encoded vectors of words from word context as an input. One-hot encoded means that all terms from the corpus are represented as vectors of zeros with an exceptional 1 at the position of the word in the corpus. The aim of the prediction is to compute the maximal probability that word w occurs in the context of a few preceding and following words. It is called a bag of words since the order of the words does not matter [34].

Skip-gram is the opposite of the CBOW model. As an input, it takes the one-hot encoded central word and predicts its context. In the training process, this model creates a contextual window that consists of multiple surrounding words, and it predicts the probability of each contextual word based on the input word. Similarly to CBOW, this model does not rely on the order of the words in the document but rather works with the relationship of the words [34].

In order to feed the output of this model to the classifier, we must transform the whole article into a numerical vector. There are several ways to do this, but the most common ones are averaging the vectors [35] of all the words in the article or, in the case of neural networks, using the Embedding layer, which maps each word to its embedding vector [26, 20].

2.3.2.4 GloVe

Global vectors for word representation (GloVe) is another technique for obtaining the vector representation of the words. GloVe utilises the word-word co-occurrence matrix in which stores the occurrence of concrete words in the context of others. The ratio of co-occurrence is used for capturing the relationship between concrete words [35].

GloVe, originally proposed in 2014, comes with two main highlights: nearest neighbours and linear substructures. The first one means that GloVe uses Euclidean distance or cosine similarity for measuring the linguistic distance and similarity between words, which are effective metrics.

However, this approach can lead to an interesting phenomenon. The nearest neighbours of concrete terms can lie outside of the average human's vocabulary. For example, the seven closest words to the target word frog are:

1. Frogs
2. Toad
3. Litoria
4. Leptodactylidae
5. Rana
6. Lizard
7. Eleutherodactylus

These are not typical synonyms, even though they are factually correct.

The second highlight is the linear substructures, which are strongly related to metric operations since the relationship between two words is described with a single number. This simple approach can lead to problems since all pairs of words have more complex relationships that cannot be described with only one number. GloVe is designed to capture the relationship between two words using vector differences. The vector difference of two words is comparable to the difference between their synonyms or other related words, which leads to creating linear substructures. These substructures are demonstrated by the pair of words "man" and "women" as portrayed in Figure 2.1.

The distances between those two words are similar to other terms describing family ("brother"-"sister") or rulers ("duke"-"duchess", "king"-"queen"). Also as described, the gender-specific terms are grouped together [36].

2.3.2.5 Pretrained word embeddings

Word embedding is considered a powerful NLP technique. Because of this, many pretrained approaches that can be easily distributed and applied have been published in the past few years. Numerous studies show that applying pretrained embedding can enhance the performance of various NLP tasks [37].

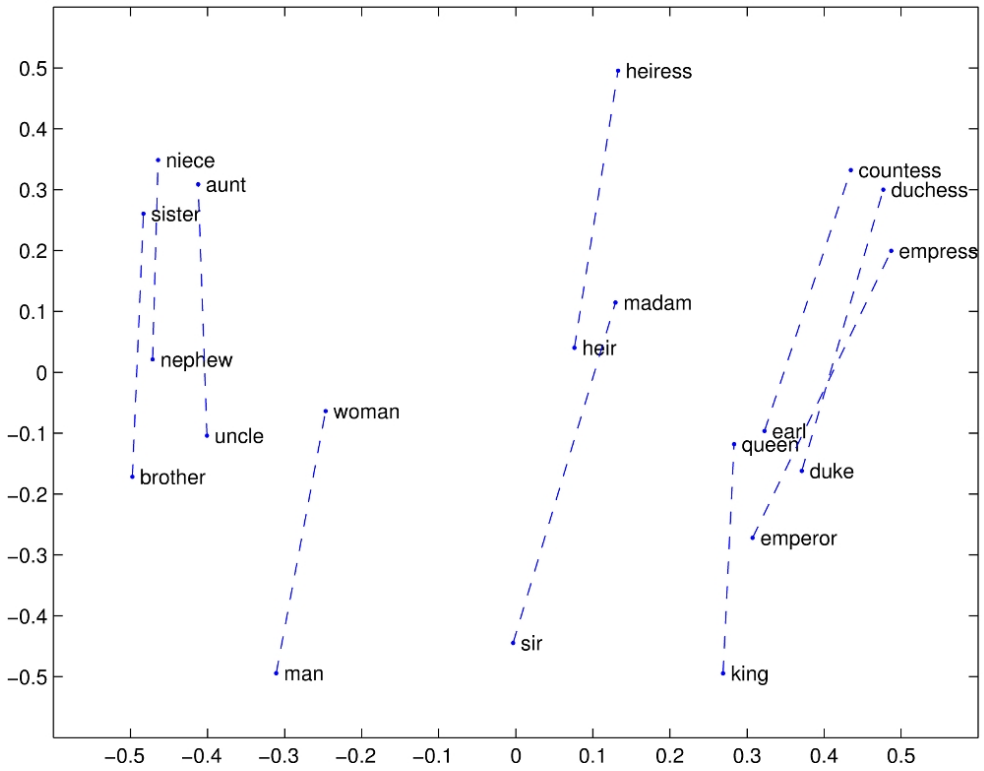


Figure 2.1 Linear substructures of man and woman [36].

2.3.3 Machine learning classification models

As we mentioned in section 2.3, the classification of fake news based on linguistic features falls into the category of supervised learning. This means that during training, the model is trying to find the most accurate relationship between the provided features and the given class [29].

The models are usually complex structures with hyperparameters that define the complexity of each model. Hyperparameters are the variables that control the training process of the model. Finding the best hyperparameter setting can enhance the performance of the model. This is done by tuning. To tune the hyperparameters, we split the original dataset into three parts: training, validation, and test data. The hyperparameter tuning is done by finding the best combination using the validation data. The purpose of the test set is to measure the performance of the model, and thus it shall be entirely separated from these two groups. There is no defined ratio between those subsets, but usually the train set contains 60% of the original dataset and the others carry 20% each [29, 38].

One possible alternative to creating the validation set is to evaluate concrete hyperparameter settings using cross-validation. Cross-validation is a set of techniques where we do not create a validation set. Instead, we split the training data into smaller subsets. The basic type of cross-validation is k-fold cross-validation, during which we split the training set into k roughly same-sized subsets. Then the model is trained for each hyperparameter setting using $k - 1$ subsets. The last one is left for evaluating the performance and computing the classification error. The final error of the model with a concrete set of hyperparameters is then calculated as an average of all the previously computed errors. This method is an efficient way to deal with the lack of training data, but it is also computationally intensive [38].

2.3.3.1 Random forest

According to the meta-analysis [6] mentioned earlier, random forests were the most frequently used machine learning model by fake news classification researchers in the past few years. Random forest is an example of the so-called ensemble method, which utilises multiple smaller machine learning models. The final prediction of those models is then a combination of the predictions of the smaller models. Random forest, as the name suggests, utilises decision trees usually trained via bagging.

Bootstrap aggregation, or bagging, is a method of training that splits the original training set into the same number of subsets as the number of created decision trees. The size of all subsets should be roughly the same. This is done by performing random sampling with a replacement on the dataset, which is called a bootstrap. After the dataset is split, each tree is trained on its own subset. To make the final prediction, the predictions of all submodels are aggregated, typically by statistical mode, i.e., the most frequent class among the predictions is selected.

Due to its complexity, this model has numerous hyperparameters. Many of them, such as the maximal depth of trees, are derived from the decision tree classifier. Another important hyperparameter is the number of trees in the random forest. With an adequate number of diverse trees, this model can achieve great results [39, 40].

2.3.3.2 Naive Bayes

Naive Bayes is a classification model based on Bayes' theorem, which can transform any conditional probability of $P(X|Y)$ into three separate probabilities. It is called naive Bayes due to its simplistic assumption about the features. It considers all features as independent for a given class.

This model classifies data using maximum a posteriori (MAP) estimation. It finds the highest probability of $P(Y = y|X = x)$ for each class y from all classes Y and feature x from all features X. This can be described as follows:

$$\hat{Y} = \underset{y \in Y}{argmax} P(Y = y|X = x) \tag{2.2}$$

where $\hat{Y}$ is the estimated class.

Since this is difficult to compute directly, Bayes' theorem is applied. All of the n features are considered independent for all classes y, the final MAP estimation of the predicted class is:

$$\hat{Y} = \underset{y \in Y}{argmax}(\prod_{i=1}^{n}(P(X = x_i|Y = y)P(Y = y))). \tag{2.3}$$

Although the presumption that features are conditionally independent is often wrong, it prevents the model from suffering the curse of dimensionality. Therefore, it is capable of achieving good results on a relatively small amount of data [19, 41].

2.3.4 Artificial neural networks

Although traditional machine learning models are still popular in the field of fake news detection, more advanced concepts such as deep neural networks attracted a lot of interest and were widely applied in the past few years [6].

Deep learning, which is considered a field of machine learning, employs multiple layers to extract important information from both huge amounts of data and data collected from different sources. This approach is an application of neural networks, which stack multiple layers of nonlinear processing units that are capable of feature extraction and transformation [42, 43].

There are many existing architectures of the network, but in the following text, we describe only those used in the experimental part of this thesis. For example, we disregard those based on transformers, mentioned in section 2.2.1.

2.3.4.1 Perceptron

The perceptron is the most fundamental unit of neural networks. Originally proposed back in 1943, it was inspired by the structures of neural structures. It is worth noting that in academic literature, the perceptron is sometimes referred to as a *neuron*.

The structure of a perceptron can be observed in Figure 2.2. We can see that the unit has multiple inputs $x_1 ... x_n$ associated with their weights $w_1 ... w_n$ and bias w_0. The unit calculates the weighted sum of all inputs, adds the bias, and plugs the result into an activation function. The most popular activation function is the sigmoid function, which is usually computed as the following:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

and the heaviside function that is calculated by the following formula:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.5}$$

The output of the perceptron can be described with the following formula:

$$\hat{Y} = f(\sum_{i=1}^{n} x_i w_i + w_o) \tag{2.6}$$

where f is the non-linear activation function and n is the number of inputs.

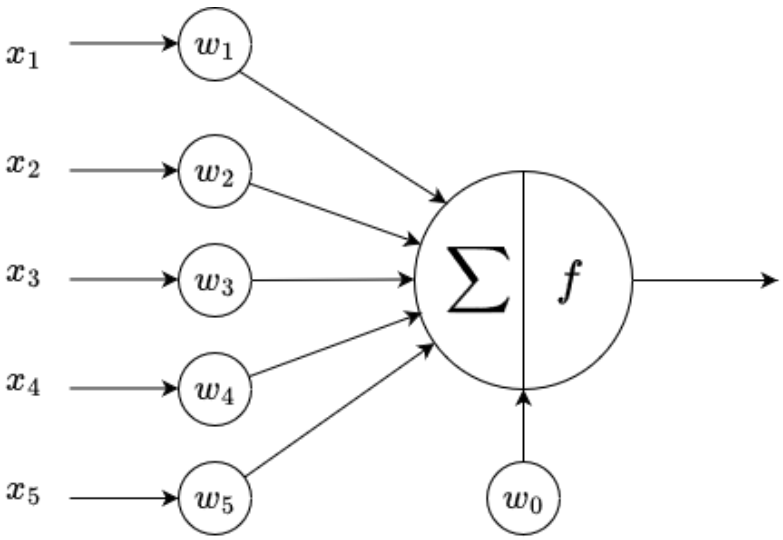


Figure 2.2 Structure of a perceptron inspired by [40].

Perceptron itself is capable of binary classification. If the output exceeds the given threshold, a positive class is returned. To train a single perceptron, the bias and weights are adjusted to minimise the error. During the training process, the perceptron is fed one training instance at a time, and a prediction is made. Afterwards, the weights, including bias, are updated according to the classification error. If no change occurs for each training instance, the training process is done [44, 40].

2.3.4.2 Multilayer perceptron

Even though a single perceptron is capable of binary classification, it is incapable of solving some trivial tasks, for example, XOR prediction. This shortcoming can be solved by stacking multiple perceptrons into a coherent structure. A multilayer perceptron (MLP), as seen in Figure 2.3, is composed of an input layer, one or more layers of stacked perceptrons (so-called hidden layers) and the output layer. The number of perceptrons in the output layer is based on the given task. For a binary classification, the output layer consists of one perceptron. For a multi-class problem, the number of perceptrons equals the number of distinguished classes.

The purpose of an input layer is to feed all units of the first hidden layer with all features from the input vector, so the number of units is the same as the number of input features. The number of hidden layers is based on the given task, and the number of units in these layers may also vary. As an activation function, typically a rectified linear unit activation function (ReLU) is chosen instead of the sigmoid function due to problems described in the following text. The function is usually defined by the following formula:

$$f(x) = max(0, x). \tag{2.7}$$

Figure 2.4 displays graphs of the mentioned activation functions.

The output layer of neurons for the binary classification task utilises the sigmoid function, which returns real numbers between 0 and 1 and can distinguish the predicted classes. The sum of returned values from all units then does not have to be added to one [45, 44, 46].

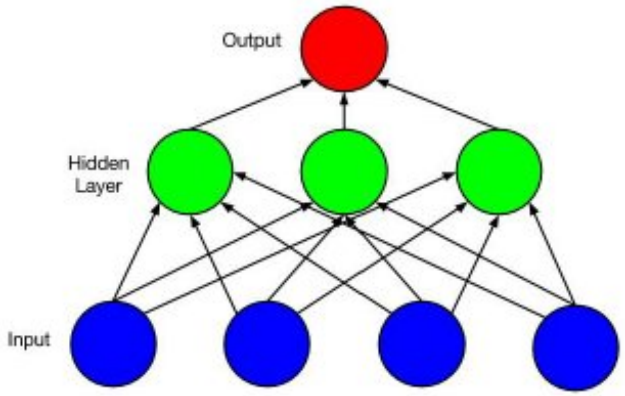


Figure 2.3 A multilayer perceptron according to [45]

2.3.4.3 Training process of MLP

The ability to predict is based on the weights and biases of all perceptrons in the model. They are assigned randomly and then adjusted by the backpropagation algorithm described further.

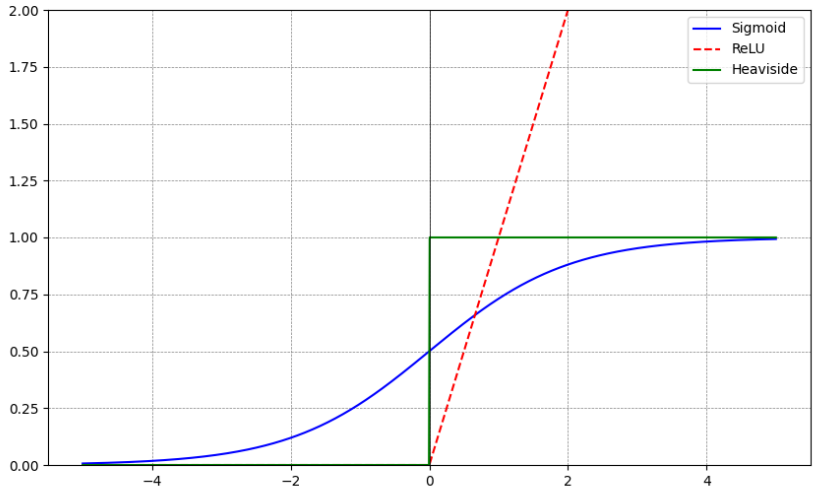


Figure 2.4 ReLU, sigmoid and heaviside activation functions

The parameters are tuned to find the minimum of the loss function that measures the error. In binary classification, binary cross entropy serves this purpose and can be described with the following formula:

$$L(Y, \hat{Y}) = -\frac{1}{N}\sum_{i=1}^{N}[Y_i \log(\hat{Y}_i) + (1 - Y_i)\log(1 - \hat{Y}_i)] \tag{2.8}$$

where Y is the real value, $\hat{Y}$ is the predicted value and N is the size of the dataset. For multi-class classification, categorical cross entropy is usually used. This function is defined by the following formula:

$$L(Y, \hat{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} Y_{ij} \log(\hat{Y}_{ij}) \tag{2.9}$$

where Y is the real value, $\hat{Y}$ is the predicted value and N is the number of instances and C is the number of classes.

Training of the model is done by a Backpropagation algorithm that consists of two distinct phases. A forward pass followed by a backward pass.

In the first one, the model is fed with each instance from the training data set. The outputs of all units are computed starting with the first hidden layer and moving on to the next layers using the outputs from the previous layers as inputs to the following ones until the output layer is reached. The intermediate results are preserved for the second phase. The model is not filled with all the training data at once. Instead, to speed up the whole process, the dataset is split into batches of typically smaller sizes, e.g., 32 instances. A full cycle of all training data being exposed to the model is called an epoch.

When the last layer is reached and the output of the network is calculated, the second phase begins. The loss function is employed to measure the output error of the network. Then the contribution of each connection to the error is computed, starting with the final layer and then moving backward until the input layer is reached. This is done by applying the chain rule, which enables us to calculate the gradient of the loss function for each layer.

Finally, the weights and biases are updated in the opposite direction of the just-computed error gradients. Finding the optimal learning rate that determines the size of the step to take is a difficult task. The regular method for this is by using the gradient descent algorithm. This is not the only possible approach since it is usually optimised to attain better results faster. Optimizers that are capable of both minimizing the loss function and speeding up the training process are described in the following section.

The problem with the backpropagation algorithm described in the previous paragraph is that as the backward phase progresses to the lower layers of the model, the gradients often get smaller and smaller. This can lead to the so-called vanishing gradient problem. One of the reasons for this is the usage of the sigmoid activation function which was later in the research swapped for ReLU or another function that does not saturate at zero or one with a derivative close to zero.

On the other hand, the ReLU function might also come to problems. Because it returns zero for every non-positive number, some perceptrons could "die" during training. This means that those units start returning only zeros no matter what [40, 47].

2.3.4.4 Gradient descent optimization

As we mentioned earlier, the essential part of training the MLP is to find the minimum of the loss function by tuning the parameters and biases in the network. The most basic method is gradient descent, which has multiple variants described further [48, 40].

Stochastic gradient descent (SGD) is a modification of gradient descent that does not update the weights in the model for the whole training set. Instead, it picks a random instance, computes the gradients based on this one instance, and updates the weights. This makes the algorithm faster and more memory efficient since it only stores one instance at a time. On the other hand, this can lead to problems with convergence to the exact minimum because the algorithm might overshoot. One of the possible solutions to this might be to decrease the learning rate over time. The weights are updated by the following formula:

$$\theta = \theta - \eta \nabla_\theta L(\theta; x; y) \tag{2.10}$$

where θ denotes the weights, η is the learning rate, and $\nabla_\theta L(\theta; x; y)$ are the gradients of the loss function regarding training instance x and its label y [48, 40].

Momentum is an extension to the previously described SGD. This extension allows the algorithm to accelerate in the right direction, leading to faster convergence and reduced oscillation. Momentum utilises the previously computed gradients to update the weights. At each iteration, it multiplies the previous gradients v_{t-1} by a "friction" parameter γ which regulates the speed of the acceleration of the process. The weight update can be described using the following formulas:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta L(\theta), \\ \theta &= \theta - v_t \end{aligned} \tag{2.11}$$

where v_t is the current update vector, η is the learning rate, θ are the weights and $\nabla_\theta L(\theta)$ are the gradients of loss function based on weights [48, 40].

AdaGrad or adaptive gradient algorithm, is different from the earlier-described optimisation technique because it adapts the training rate for each parameter. It accumulates the square gradients of the loss function regarding the parameters θ. Usually, the tweaks to parameters that are frequently updated are smaller, whereas those that get updated less often tend to get larger adjustments. The update of each parameter θ_i is defined by the following formula:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_t + \epsilon}} . \nabla_{\theta_t} L(\theta_{t,i}) \tag{2.12}$$

where G_t stands for a diagonal matrix where each diagonal element is the sum of the squares of the gradient and ϵ is a small constant number that prevents dividing by zero [48, 40].

The biggest problem with AdaGrad is that the learning rate might shrink over time, and the model stops obtaining any new knowledge. That can be fixed by the following model.

RMSprop was developed to fix the AdaGrad problem of not converging into the global optimum. This is done by accumulating only gradients from the recent history of iterations [48, 40].

Adam of Adaptive Moment Estimation is an optimisation technique that employs ideas of RMSprop and Momentum algorithms. It keeps track of both exponentially decaying averages of past squared gradients v_t like RMSprop and an exponentially decaying average of past gradients similar to Momentum, which can be observed in the following formulas:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L(\theta) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta L(\theta))^2 \end{aligned} \tag{2.13}$$

where m_t and v_t are the estimates of mean and variance and β_1 and β_2 are the "friction" parameters that are typically initialised very close to one, e.g., 0.9 for β_1 and 0.999 for β_2. Since the parameters m_t and v_t are usually initialised as vectors of zeros, they tend to be biased towards zero at the beginning of the training therefore, they must be corrected as follows:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_1^t} \end{aligned} \tag{2.14}$$

Finally, the parameters are tweaked according to the update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.15}$$

where η is the learning rate and ϵ is small constant very close to zero that prevents the denominator from reaching zero [48, 40].

2.3.4.5 Dropout

Dropout is one of the existing regularisation techniques that prevents the MLP from overfitting. It also allows us to combine results from multiple network architectures. The term dropout means that some of the perceptrons are randomly switched off during every training step with their input and output connections. The units are ignored randomly. The probability p of being dropped out is set by a fixed hyperparameter, and its common value is between 10 and 50% [49].

2.3.4.6 Recurrent neural networks

The model of multilayer perceptron is a feedforward structure. This means that the outputs of the units can flow only in one direction. Recurrent neural networks (RNNs) are a class of neural networks that allow the creation of directed cycles in their architecture. That implies that the previous output of a perceptron can be used as an input to the same one in a hidden state.

The core of the RNN is the recurrent neuron. The architecture of this structure is similar to the already described perceptron. However, this one comes with two sets of weights: w_x for the inputs x_t and w_y for the outputs from the previous step y_{t-1}. The output $\hat{Y}$ of this structure is then computed by the following equation:

$$\hat{Y} = f(\sum_{i=1}^{N}[w_{x,i}x_{t,i} + w_{y,i}y_{t-1,i}] + w_0)$$

where w_0 is the bias and f is the activation function.

Since the output of the recurrent neuron is defined by the previous states, we can claim that it has "memory" which allows the structure to find patterns in the input data. This makes RNNs suitable for working with sequential data such as audio, video, or text.

The training algorithm of RNNs is called backpropagation through time (BPTT). It is similar to the already-described backpropagation algorithm, but with one key difference, which is that the network is unfolded in time, as depicted in Figure 2.2.

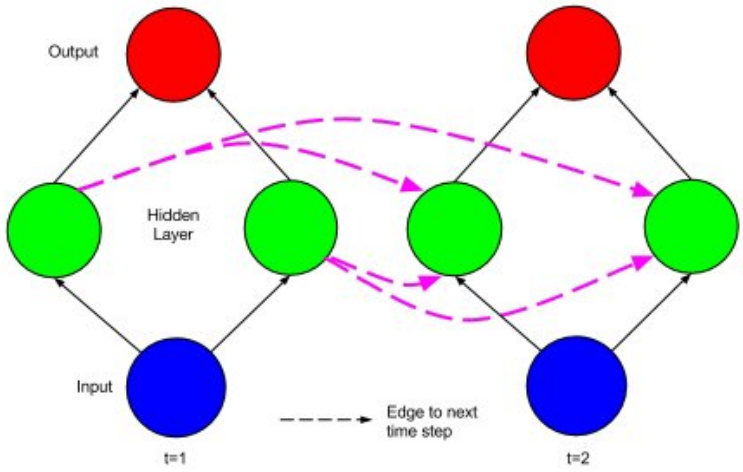


Figure 2.5 Simple RNN unfolded through time according to [45].

After the network is unfolded, the forward feeding phase begins, and the output is evaluated by the loss function. It is followed by the backward phase when the gradients flow backwards through the unfolded network. Finally, the parameters are updated. This often leads to the problem of either vanishing or exploding gradients since the multiplication of gradients is repeated many times. In the following section, we describe a structure that can deal with this problem [45, 40, 50].

2.3.4.7 Long short term memory

Long short-term memory (LSTM) architecture, originally introduced in 1997, was designed to address the exploding/vanishing gradient problem. The architecture of the cell is a modification of the recurrent neuron, which can store multiple pieces of information from previous time steps and thus learn which information will be preserved and which will be forgotten. The architecture is depicted in Figure

The architecture of these cells consists of three gates, in the picture noted as "X" in pink circles, regulating the amount of information passed to the next state and to the output.

The **forget gate** (intersection of f_t and C_{t-1}) controls which parts of the long-term state will be forgotten.

The **input gate** (intersection of i_t and $\tilde{C}_t$) adds some parts of the input to the long-term state.

The **output gate** (intersection of o_t and C_{t-1}) passes some parts of the long-term state to both the output of this cell and its short-term state.

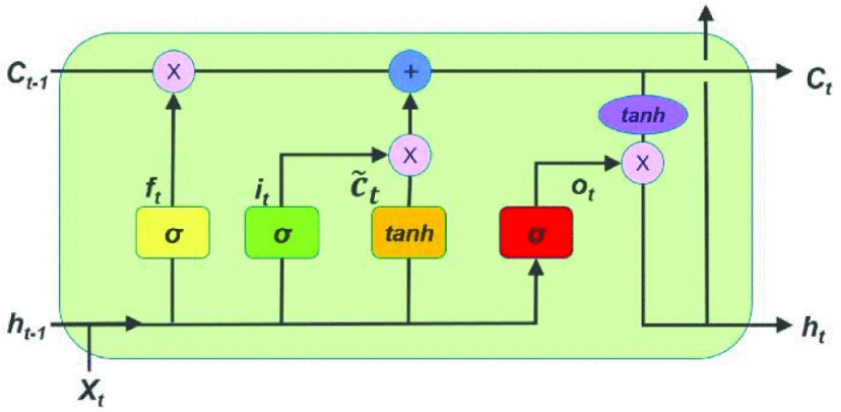


Figure 2.6 Architecture of LSTM cell according to [51]

When the input X_t is passed to the cell, it is sent to the four different functions combined with the previous short-term state h_{t-1}. The outputs of these functions are then passed to the gates, and the cell state (long-term state) C_t is passed to the next state as well as the new short-term state h_t. Finally, the output is returned. The process can be described with the following computations:

$$\begin{aligned} f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \\ i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\ \tilde{C}_t &= tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \\ C_t &= f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t, \\ h_t &= o_t \otimes tanh(C_t) \end{aligned} \tag{2.16}$$

where W_x represents the weights of the input, W_h denotes the weights of the previous state h_{t-1}, b stands for biases and $\otimes$ is the element-wise multiplication. We must also mention that the output equals the current short-term state h_t [40, 51].

2.3.4.8 Convolutional neural networks

Although convolutional neural networks (CNN) are typically used for image processing, they are still capable of achieving promising results in text classification. The architecture of the CNN is similar to that of the classical MLP, but as one or more hidden layers, it incorporates convolutional hidden layers and pooling layers.

As the name of this structure suggests, it utilises convolution in the classification process. To perform convolution over the input, firstly, the input text (sentence or an article) is converted into an embedding matrix of sizes $n \times k$, where n is the maximal number of words in the article and k is the dimension of the word embedding. The final article matrix W is created by simple concatenation, which can be described with the following formula:

$$W = w_1 \oplus w_2 \oplus \ldots \oplus w_n \tag{2.17}$$

where w_n is the embedding vector of nth word and $\oplus$ is a concatenation operator.

After the matrix is created, a convolutional layer is employed. Its purpose is to apply filters to find patterns and features in the text. The filters, also called kernels, slide over the matrix,

perform the convolution operation, and return a new feature. For example, with a filter ω of dimension of $l \cdot k$, where l is the number of words in the sliding window, a feature c is generated by

$$c = f(\omega.W_{n:n+l-1} + b) \tag{2.18}$$

where b denotes the bias and f is a non-linear activation function. In order to extract different features at different scales, filters with different sizes are applied. The convolutional layer produces a feature map which consists of features extracted from the input matrix.

This is typically followed by the pooling layer. The purpose of this layer is to aggregate data from produced feature maps. The most popular way of pooling is max-pooling, which selects the highest value from each window of a given feature map and hence captures the most important features [52, 53].

2.3.5 Evaluation metrics

The performance of the classifiers is typically evaluated in two stages of the model. In the training stage, evaluation is used to optimise the classifier to enhance its performance. In the tasting stage, the evaluation gives us relevant information about the effectiveness of the final classifier on unseen data.

The metrics used for measuring quality vary, but the most popular ones are those related to the confusion matrix. If we consider fake news classification as a binary problem with two possible classes: 0, which means that the article is fake news, and 1, which means that the text is reliable. The main idea behind the confusion matrix is to display how the instances were classified by the model compared to their original class. In our case of those two classes, the following values describe the possible outcomes of predictions:

- True positive (TP) is the number of instances from the class 1 correctly classified as 1,
- True negative (TN) is the number of instances from the class 0 correctly classified as 0,
- False positive (FP) is the number of instances from the class 0 misclassified as 1,
- False negative (FN) is the number of instances from the class 1 misclassified as 0.

We use these values to compute the following metrics:

Accuracy measures the ratio between correct predictions and the total number of instances. It is computed with the following formula:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.19}$$

Precision, on the other hand, is used for measuring the ratio of correctly classified positive instances to all positive instances. The following formula is used:

$$PREC = \frac{TP}{TP + FP}. \tag{2.20}$$

Recall measures the ratio of positive instances that are correctly classified to all correctly classified. The way its calculated is following:

$$REC = \frac{TP}{TP + FN}. \tag{2.21}$$

F1 score is a harmonic mean of precision and recall. It is highly used for imbalanced datasets in which one class predominates the others.

$$F1 = \frac{2.PREC.REC}{PREC + REC}. \tag{2.22}$$

When we perform classification into more than two classes, there are multiple options to adjust the confusion matrix. We can either create a separate confusion matrix for all of the classes independently and consider being part of that class as a binary problem, or we can create one big confusion matrix that includes all of the classes. The evaluation metrics must, in this case, be slightly modified [54, 40, 19].

2.3.6 Chapter summary

In this chapter, we defined the basic terms that we use in this thesis. Then we took a look at the current state-of-the-art of fake news detection and described several existing approaches that can be utilised for this task. This included an overview of examples of academic literature regarding the explored approaches. Finally, a description of the theoretical background of natural language processing and classification algorithms was presented.

Chapter 3
Datasets

In this chapter, we present a survey of existing state-of-the-art fake news datasets that are used in current research. This includes datasets that will be used in the last chapter for performing experiments. This chapter is divided into two sections. Firstly, we give a brief description of several existing datasets and then we describe those that are self-obtained. The second part also contains an outline of the tools used for obtaining the new datasets.

In the final part of this thesis, we experiment with two already-made fake news datasets in the English language proposed in the past five years for research purposes. This is followed by performing experiments on two self-obtained datasets, which are combinations of texts from online fake news databases in both English and Czech. Their description is provided in the following text. The text also includes word clouds from selected datasets.

Wordcloud is a Python library that allows users to create a graphical representation of the most used tokens in any dataset [55]. The word clouds were generated for all datasets that are going to be used for the experiments.

3.1 Existing datasets

Thanks to an ongoing interest in the fake news detection area, researchers and news outlets have been working on creating benchmark datasets that can be used in fake news detection research. In the survey of fake news evaluation datasets [56] from 2021, the authors examined 27 publicly available datasets published between 2009 and 2019, mostly in English and Spanish. However, three of the datasets were multilingual. A majority of the datasets were related to politics (55.6 %) and society (44.4 %). It is also worth mentioning that most of the surveyed datasets are smaller than 10,000 data points and distinguish only two classes (true or fake). The list of surveyed datasets also includes different rumours, hoaxes, reviews, clickbaits, and satirical texts. Although the definition of fake news provided at the beginning of this thesis can be applied to all of these categories, they will not be the subjects of our experiments.

3.1.1 ISOT fake news dataset

The ISOT fake news dataset consists of over 44,000 real and fake articles. The truthful articles were obtained from the Reuters agency[1]. The fake ones were obtained from various unreliable

[1] `https://www.reuters.com/`

websites and fact-checked by US fact-checking organisation Politifact, and Wikipedia. The articles were mainly collected from the news published in 2016 and 2017 and contain various articles regarding different topics, but the majority focus on politics. [57].

The dataset can be downloaded from the original website of the University of Victoria[2]. It comes within 2 files: *True.csv* of 21,417 articles and *Fake.csv* of 23,481. The files also include metadata about the texts, such as the title of the text, subject, and date of publication. We preprocessed the dataset by merging these two files, adding proper labels, and merging the title into the article itself. Then we remove the title attribute. Since the subjects of real news comprise *World-News* and *Politics-News* while fake news subjects include *Government-News*, *Middle-east*, *US News*, *left-news*, *politics*,*News*, there is no overlap, and therefore this feature was also removed. Finally, we reduced the size of the dataset by excluding all articles from 2016 because we are interested in newer data. Finally, we shrank the dataset to contain only 10,000 articles that will be used to perform the experiments. This was mainly done to lower the size difference between this dataset and the others. The final ratio between real and fake news ended up being 53% to 47 %.

The most frequent tokens in the ISOT dataset are depicted in Figure 3.1. As the word cloud shows, articles in this dataset predominately talk about former US President Trump and his policies.

Figure 3.1 Most frequent tokens in ISOT dataset.

3.1.2 ReCOVery

After the outbreak of COVID-19, the media ecosystem experienced an "infodemic" of fake news. The authors of the paper [58] examined 2,029 English-language articles published between January and May 2020 with both low and high credibility related to the COVID-19 epidemic. The articles were crawled from 55 different media outlets and then checked for their credibility by two resources: NewsGuard[3]and Media Bias/Fact Check[4]. NewsGuard is a transparent website created to review and rate news websites run by journalists and editors. Media Bias/Fact Check checks the factual accuracy and political bias of media outlets. In this paper, the authors do not

[2] https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/
[3] www.newsguardtech.com/
[4] www.mediabiasfactcheck.com/

focus only on news articles but also on tweets and social media posts, which are not used in this thesis.

The dataset is available online on GitHub[5] and it is the most imbalanced dataset of those described, with the reliable to unreliable ratio 67% to 33%. All of the articles will be used in the experimental part. The original file contains numerous features such as *url*, *publisher*, *political bias*, etc. Although this metadata might be useful for some deeper analysis, we removed all of the columns except the title of the article, which we merged into the text to be considered one entity. As we can see in the word cloud, the articles speak as much about the pandemic as its political consequences.

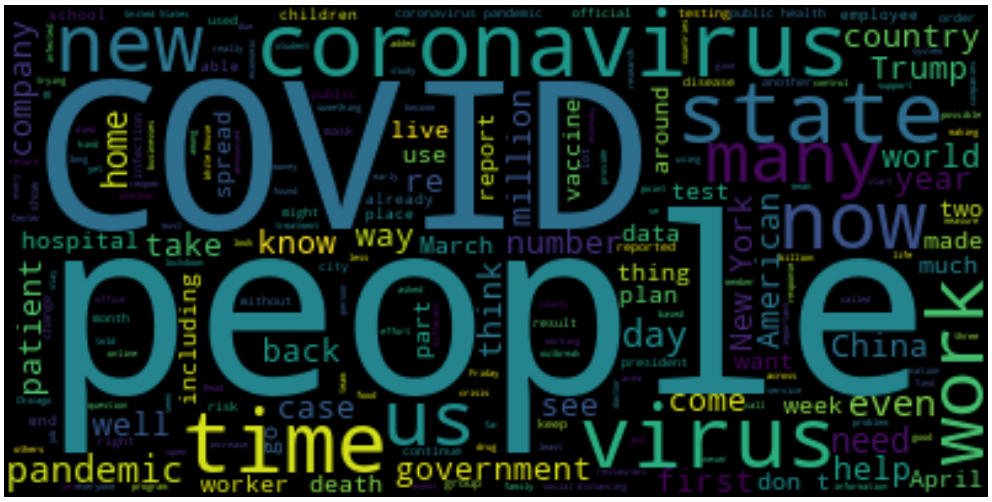


Figure 3.2 Most frequent tokens in ReCOVery dataset.

3.1.3 Unused datasets

As we mentioned earlier, fake news articles are ubiquitous. So the research into their detection has developed significantly. This means that more and more datasets are created every year. In this section, we present some publicly available datasets in English that were initially considered for this thesis but are left for future work.

Liar, Liar Pants on Fire is a publicly available benchmark dataset for fake news detection originally presented in 2016. It consists of 12,788 short statements collected between 2007 and 2016. This is not a typical fake news dataset because it does include only short, manually labelled statements in various contexts from the website of US fact-checking site Politifact[6]. This website provides a detailed analysis report and links to source documents for each case. Each statement has been evaluated by the Politifact editor. The statements fall into one of six possible categories of truthfulness. For each statement, the creators of the dataset provide the other 12 metadata columns, including the subject of the statement, the name of the speaker, their job, party affiliation, and the context of the statement [15]. The dataset is available online on the website of the University of California, Santa Barbara[7].

BuzzFeedNews is one of the datasets related to the 2016 U.S. presidential election. It contains news published on Facebook by nine news agencies over the course of a week, two

[5] https://github.com/apurvamulay/ReCOVery
[6] www.politifact.com/
[7] https://sites.cs.ucsb.edu/~william/data/

months before the election. The dataset consists of 1,627 factchecked articles (claim by claim) from both mainstream and political-biased media. It is available on GitHub[8] [59].

FacebookHoax is an example of a dataset that includes data from Facebook pages related to scientific news and conspiracy pages. The dataset consists of 15,500 posts with information about likes and user interaction and is also available on GitHub[9] [60].

CLIMATE-FEVER is an example of a dataset related to one particular topic. This dataset, which is available online[10], consists of 1,535 real-world claims regarding climate change. For each claim, the authors retrieved the top five relevant evidence sentences from Wikipedia. Evidence was checked by humans, and the claims were labelled as supported, refuted, or not giving enough information depending on the evidence [61].

3.2 Self-obtained datasets

Governments and researchers are not the only ones who store articles that can be regarded as fake news. Due to the ubiquity of this phenomenon, many individuals and groups have decided to fact-check non-credible news on the internet. There are several projects and organisations all around the world that store fake news articles in publicly available databases. These projects are often run by journalists and experts in this area. However, not all of them are credible because of their political bias. In this thesis, we selected two of them that have not run into any remarkable controversy in the past few years.

3.2.1 Crawling news

news-please is an open-source news crawler[11] that can extract structured information from almost any news website. It combines the power of multiple state-of-the-art libraries and tools for web crawling. It can extract multiple features of the article, such as the headline, main text, publication date and inserted images [62].

3.2.2 EUvsDisinfo

EUvsDisinfo is a flagship project of External Action Service's East StratCom Task Force that was established in 2015 to forecast, address and respond to the disinformation campaign run by the Russian Federation against the European Union.

This website monitors and analyses different disinformation media in 15 languages and compiles them in the EUvsDisinfo database[12]. In this database, each disinformation case is saved with meta-information about the date of detection, title of the news, language of the article, and countries discussed in the text. This organisation also releases disinformation reviews and analyses of up-to-date fake news [63].

However, to create a brand-new fake news dataset, we did not obtain texts exposed on the website. There are two reasons for this. Firstly, this database obtains data related to the Kremlin and its narrative, so the articles are predominantly written in Russian, and secondly, many of the links that are related to fake news written in English are not working anymore.

Instead, we used the provided summary of the disinformation, which is written in English for each article. We obtained these data from an already existing dataset[13] created for a case study addressing disinformation on the web [64]. Working with the summary of an article instead of

[8] www.github.com/BuzzFeedNews/2016-10-facebook-fact-check/tree/master/data
[9] www.github.com/gabll/some-like-it-hoax/tree/master/dataset
[10] www.huggingface.co/datasets/climate_fever
[11] www.github.com/fhamborg/news-please
[12] www.euvsdisinfo.eu/disinformation-cases/
[13] www.github.com/FloFloB/Euvsdisinfo-dataset

the original article should not lead to any problems because of the following reasons: Firstly, the summaries contain the important points of the articles and are long enough to be meaningfully classified, and secondly, the way we consume the media in today's world includes consuming shorter statements and articles due to our attention span getting generally shorter [65]. This phenomenon should be taken into consideration in further research on fake news detection.

The cleaning process was similar to the preceding ones. We only chose articles that were published in 2022. We used titles and texts of summaries of fake news. Finally, we concatenated these two columns together to create only one text feature.

To create a corpus of reliable news, we crawled summaries of war days published by The Guardian[14] since the beginning of the Russian invasion of Ukraine in February of 2022. The final dataset consists of 2,139 texts with a ratio of 1,066 reliable news to 1,073 unreliable.

As we can see in the presented word cloud Figure 3.3 the articles of this dataset relate mainly to Russia, Ukraine, and the war.

Figure 3.3 Most frequent tokens in EUvsDisinfo dataset.

3.2.3 Czech dataset

Čeští elfové is a Czech citizen initiative that actively monitors, analyses, and debunks fake news in the Czech media ecosystem [66]. This organisation created a publicly available database Eldariel[15] of chain letters, which is a popular way of sharing disinformation content in the Czech environment. This database stores the texts of chain emails as well as their areas of interest and people (usually politicians) mentioned in the letters. The presented text content is shortened, and the chain mails consist of just pictures or links to manipulative content on YouTube[16]. Therefore, the data had to be properly filtered before creating the dataset. We ended up with a dataset of 405 fake news texts in the Czech language regarding the topics of Russia, Ukraine, and international politics. Chain emails that are available in mail form only make up about 30% of the dataset. The rest consists of articles linked to the emails downloaded from the linked media outlets. We combined this set with 590 articles from the Czech public television news website[17] related to the same topics. Both reliable and unreliable parts of the dataset only consist of

[14] www.theguardian.com
[15] www.eldariel.cesti-elfove.cz
[16] https://www.youtube.com/
[17] https://ct24.ceskatelevize.cz/

content published in 2022 and in the first quarter of 2023. Figure 3.1 depicts the most frequent tokens in the Czech dataset.

Figure 3.4 Most frequent tokens in the Czech dataset.

3.3 Chapter summary

In this chapter, we provide a survey of existing datasets of fake news texts. We presented those used in the experimental part as well as those that we did not use. The sizes of each dataset and the ratio between reliable and unreliable content are depicted in Table 3.1. The sizes and ratios vary as well as the topics these datasets are related to. Also, the time periods the datasets come from differ. The first and oldest dataset that is used in the experimental part of this thesis is the modified ISOT dataset. The second dataset is ReCOVery, which is made up of articles related to the outbreak of COVID-19 in 2020. The third dataset consists of summaries of news based on the EUvsDisinfo database. The last dataset, and also the only one that is made up of articles in Czech, is based on the chain mails from the Eldariel database by Čeští Elfové.

Table 3.1 Comparison of datasets used in the experimental part of this thesis

dataset	no. of texts	ratio reliable:fake news (%)
ISOT	10,000	53:47
ReCOVery	2,029	68:32
EUvsDisinfo + The Gurdian	2,139	50:50
Eldariel + ČT24	995	59:41

Chapter 4

Implementation and experiments

In this chapter, we describe the existing tools and libraries for performing the experiments on the datasets presented in the previous chapter. Then we explain the classification experiments. Finally, we evaluate the performance of the presented models and discuss the results.

4.1 Tools and libraries

The entire implementation was done using the *Python* programming language. Python is suitable for this task because of the rich palette of existing tools and libraries for NLP and machine learning.

All of the coding was stored in multiple Jupyter notebooks[1]. Jupyter is a web application designed for creating and sharing computational documents [67].In the following text, we describe the Python libraries used for the experiments.

4.1.1 NLTK

Natural language toolkit (NLTK) is the most popular platform for text processing in Python. It provides numerous libraries for text processing purposes mainly in the English language [68]. From this library, we employed many functions related to text preprocessing.

4.1.2 Simplemma

Simplemma is a Python library that provides a simple and multilingual approach for lemmatization [69]. We used this library for the lemmatization of both the Czech and English texts.

4.1.3 czech_stemmer

Since the NLTK library does not provide the stemming algorithm for the Czech language, we use the Czech stemmer implemented by Luís Gomes in 2010. This is a Python script that enables users to perform both light and aggressive stemming [70].

[1] `www.jupyter.org/`

4.1.4 Pandas

Pandas is a Python library designed for data analysis and manipulation [71]. We used it for data preprocessing and analysis.

4.1.5 NumPy

NumPy is a Python library for scientific computing. It stores multiple data structures that make the mathematical operations efficient [72]. We used it mainly for some text preprocessing operations and model evaluation.

4.1.6 Scikit-learn

Scikit-learn, or Sklearn, is a Python library that provides a basic implementation of machine learning models and related utilities [73]. It was used for implementing the classifiers and tuning their hyperparameters.

4.1.7 TensorFlow

TensorFlow is an open-source library used for processing data and implementing machine learning and deep learning models [74]. It was mainly used for its API **Keras** which allows users to build neural networks [75].

4.1.8 Gensim

Gensim is a Python library for an efficient representation of documents as semantic vectors. It is mainly designed to process unstructured text using unsupervised learning algorithms [76]. It was used for the implementation of the Word2Vec algorithm.

4.1.9 Pretrained GloVe

The experiments included experiments on pretrained word embeddings. We used pretrained GloVe in both languages. The English module was published in 2014 at Stanford and is available online[2] under the Public Domain Dedication and License (PDDL). The vectors are available in 50, 100, 200, and 300 dimensions, and the corpus contains 6 billion tokens and a vocabulary of 400,000. The data on which the embedding was trained was crawled from Wikipedia and English Gigaword Fifth Edition. For our task, we chose the 100-dimensional module saved in file **glove.6B.100d** that contains 100-dimensional embedding vectors of over 400,000 English words [36].

The Czech pretrained GloVe embedding was published in 2016 under the name *cz_corpus*, and it is also available online[3]. The corpus was created using both Word2Vec and GloVe [77]. The GloVe module is available only in 300-dimension vectors and contains a vocabulary of 1.25 million. The model was trained only on the Czech Wikipedia. The utilised module is stored in file **vectors_cz_glove_dim300_25** and consists of over a million 300-dimensional vectors of Czech vocabulary.

[2] `www.nlp.stanford.edu/projects/glove/`
[3] `www.github.com/Svobikl/cz_corpus`

4.1.10 Matplotlib

Matplotlib is a Python library for creating visualisations [78]. We used this package for exporting graphical representations of experimental results.

4.2 Implementation

As we mentioned earlier, the implementation was stored in Jupyter notebooks. The final program consists of four notebooks and each is responsible for a different set of tasks. The structure of the program is as follows:

- **Text_Preprocessing.ipynb** stores all the functions related to the text preprocessing.
- **Models.ipynb** saves the implementation of classification models.
- **Experiments.ipynb** contains classes that control the flow of the program and save the results of the experiments to specified folders.
- **Perform_Experiments.ipynb** is a notebook from which the experimental program is run.

All experiments on each dataset began with an instance of class *CDataSet* from the notebook Experiments.ipynb that loads the dataset CSV file using the Pandas library. The datasets had already been preprocessed into a suitable form for the program i.e., unnecessary columns had been removed and the binary column that defines the reliability of the text had been added.

The instance passed the loaded data in the form of a Pandas DataFrame to a new instance of class *CExperiment*, which preprocessed the text data and performed the experiments. Each experiment was run three times, and the results were averaged to be presented in the text.

4.2.1 Text preprocessing

Before any text preprocessing began, the data had been split randomly into a training and test set using the Scikit function *train_test_split()* with a predefined random seed by a ratio of 80:20. Texts from both of the sets were preprocessed into a form suitable for the classification models.

First, the special characters, including numbers and links, were removed, and the tokens were lowercase, except for the words that do change meaning in lowercase, such as “US”. Then the stopwords were removed, and the maximal length of the articles was restricted to 250 words (if not specified differently) to unify the lengths of the articles. The words were then converted to their base form using either lemmatization or stemming. In the case of machine learning models, we also experiment with the effect of n-gram sizes on the classification results. Using the *word_tokenize()* function from NLTK, different-sized n-grams were created:

- 1-grams
- 2-grams
- 3-grams

This processed data was then passed to the feature extraction algorithms.

4.2.2 Feature extraction

The number of training features was restricted to 10,000 in each experiment in order to limit the computational intensity.

The experiments on machine learning models employed the bag-of-words and TF-IDF models in combination with n-grams. These models are implemented in the Sklearn library. Both of these

models come with numerous parameters. In our case, we specifically set the mutual parameter *max_tokens*, which restricts the number of features and the parameter *ngram_range* that defines the size of the n-grams. In the case of neural networks, these models were not employed.

Word2Vec was utilised in experiments with both machine learning models and neural networks. We used the implementation from the Gensim library, which comes with multiple parameters. The ones that we adjusted are described in the following list:

- **epochs** defines the number of training epochs of the model. We set it to 20, which was observed to be a good compromise between sufficient training and overfitting.
- **workers** allows us to use more CPU threads during computation. We set the value to -1, which means that all available threads are used.
- **vector_size** defines the dimension of the output embedding vector. It was set to 100 in the case of the English datasets and 300 in the case of the Czech one. The reason for this is to reflect the dimensions of the pretrained GloVe embeddings.

The usage of Word2Vec for extracting features was different for both machine learning and deep learning models. In the case of machine learning models, the embedding vectors were derived from the train set, and the final vector of each article was obtained as the mean of all of the embedding vectors in the article.

In the case of neural networks, the embedding matrix from vectors obtained from the training set was created, which was later passed to the embedding layer of the networks.

Pretrained GloVe word vectors were loaded from the given files. Each file was loaded before all experiments began, and dictionaries held all embedding vectors to corresponding words. The system of processing the articles into a suitable form for classification models was similar to Word2Vec. In the case of machine learning models, the articles were converted to a mean vector. The neural networks utilised the embedding layer.

4.2.3 Classification models

The classes and functions were stored in the notebook **Models.ipynb**. This notebook comes with two abstract classes, *CModel* and *CNeuralNetwork*, that predefine the interface for child classes that implement the models. The models and their hyperparameters are described in the following text.

All of the models that we utilised for the experiments were mentioned many times in the reviewed academic literature. Naive Bayes is a simple classification model that is employed for many text classification tasks. This model is fast and can perform well on smaller amounts of data. On the other hand, random forest is a less biased, robust technique that can achieve good results in the classification task.

The types of neural networks used for the experiments were also chosen based on their results in the related work. The convolutional neural networks as well as the LSTM neural network proved to be successful in working with the sequence data. Convolutional networks are good for finding patterns in the text, whereas LSTM networks can find long-term dependencies. The concrete architectures are described in the further text.

Naive Bayes was implemented using the *GaussianNB* class in Scikit-learn. This implementation of Naive Bayes assumes that the provided features have a Gaussian distribution. This model is capable of working with negative features, which is useful for processing word embeddings. It comes with a very small number of tunable parameters. So we tweaked only one.

- **var_smoothing** defines a small number that is added to the variance of the features for more calculation stability. The values were set to six numbers from the interval between 10^{-12} and 10^{-6}.

Random forest was implemented using the *RandomForestClassifier* class from the Scikit library. This implementation comes with many hyperparameters that can be adjusted. For our purposes, we tuned the following ones:

- **n_estimators** defines the number of trees in the random forest. In the training process, we set it to values between 150 and 200 with a step size of 10,
- **max_depth** is the maximal depth of the decision trees. This value was set between 5 and 8,
- **bootstrap** is a bool value that determines whether bootstraps are used for building the trees. If it is set to false, the whole dataset is used for building each tree. We allowed both options of this parameter.

The architecture of the neural networks was inspired by the paper [79] in which the authors compared many machine learning models and neural networks on the fake news classification problem. In this paper, many architectures were tested for fake news classification and two of them were chosen for our experiments. The **OPCNN-FAKE** architecture reached the highest performance in this paper and is based on the convolutional neural network. The second architecture is simple one-layer LSTM architecture. In all experiments, the neural networks were run for 15 training epochs with a batch size of 32 samples, utilising the binary cross-entropy loss function and the Adam optimiser.

In the mentioned paper, the OPCNN-FAKE architecture outperformed the employed models on all of the datasets, including the machine learning ones. Experiments on our datasets with this model also achieved promising results. The architecture is depicted in Figure 4.1 and can be described as follows:

- **Embedding layer** encodes the words into dense vectors. In our case, the embedding matrix with a vocabulary limited to 10,000 words created from the train set is used.
- **Dropout layer** applies dropout to the input from the embedding layer. The values of probability were set to 30, 40, and 50%.
- **Convolutional layer** performs a convolution operation over the input data. The *Conv1D* implementation from Keras was employed. The number of filters was tuned during the training process. The values were set to 32, 64, and 128. The size of the kernel was also tweaked, with values of 8, 10, and 12. The ReLU activation function was employed.
- **Pooling layer** performs a pooling operation over feature maps produced by the preceding layer. We chose the max pooling version that is implemented in Keras *MaxPooling1D*.
- **Flatten layer** converts the output of the previous layer to a one-dimensional vector.
- **Output layer** produces the final output of the model. In our case, it consisted of only one neuron with a sigmoid activation function, which determines if the article is fake or reliable.

  The LSTM architecture employs only one LSTM layer, which makes it a very simple one. The architecture is depicted in Figure 4.2 and can be described as follows:

  - **Embedding layer** translates the words to their vector representation. It uses the same matrix as the previous model.
  - **LSTM layer** consists of LSTM neurons. The number of units in this layer was set to 128 and 256. The default tanh activation function is used.
  - **Dropout layer** applies dropout to the input from the previous layer. The values of probability were set to 30, 40, and 50%.
  - **Flatten layer** converts the output of the previous layer to a one-dimensional vector.

- **Output layer** returns the output of the model. It consists of only one neuron with a sigmoid activation function, which tells if the article is fake or reliable.

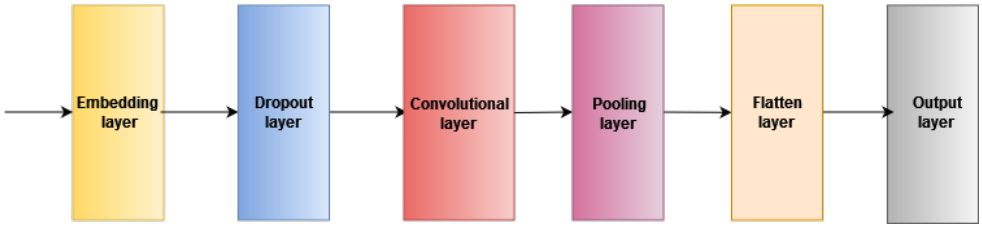


Figure 4.1 Schema of the CNN neural network architecture. Inspired by [79]

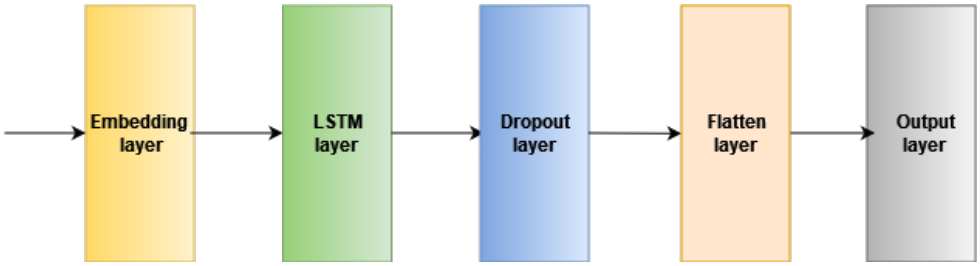


Figure 4.2 Schema of the LSTM neural network architecture.Inspired by [79]

Unlike the previous architecture, this one achieved poorer results on all datasets, as described in the mentioned paper. We managed to enhance the performance with some preprocessing methods on one of the datasets and outperformed the other classification models, which we will discuss later.

4.2.3.1 Tuning the hyperparameters

In each experiment, we created a new instance of each class that stores the classification model. The training and test data were passed to this instance, and the hyperparameter tuning began.

The hyperparameters were tuned using the *ParameterGrid* implemented in Scikit. Each model was trained with the defined parameters, and k-fold validation was employed to assess the performance of the model. In our case, the number of folds was set to 5. The average accuracy per fold was computed and stored.

Finally, the hyperparameter setting that reached the highest training accuracy was used for the classification of the test data. The result was stored, and the evaluation metrics were calculated. The performance was exported to text files in the form of latex tables at the end of all experiments per model.

4.3 Results

In each experiment, different text preprocessing techniques are used in combination with different feature extraction models. This way, all of the models on all of the datasets were evaluated. The experiment was run for three rounds, and the presented metrics are the average of those three results. All of the evaluation metrics were rounded to three decimal places.

4.3.1 Results of the ISOT dataset

This dataset was the largest one, with 10,000 news articles that were used for classification. The dataset was also very balanced, with the ratio between fake and reliable news being 47:53. The length of the articles was restricted to 250 words after stop words removal, as described earlier.

Table 4.1 depicts the results with the highest accuracy from further presented tables. The represented values were selected as the combination of text preprocessing and feature extraction that reached the highest accuracy. The overall results of all classification models on this dataset were very good with all of the models being able to reach over 90% accuracy. Except for the LSTM neural network model. We can see that the gap between the results of other models and LSTM network was almost 30%.

Model	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.954	0.944	0.966	0.955
Random Forest	0.929	0.991	0.891	0.938
CNN	0.974	0.987	0.965	0.976
LSTM	0.656	0.674	0.686	0.680

Table 4.1 The best results of each classifier in regard to accuracy on the ISOT dataset

Tables 4.2 and 4.3 depict the results of random forest and Naive Bayes algorithms. We can observe that in both cases, Word2Vec trained on the training part of the dataset reached much poorer results than the pretrained GloVe embedding vectors obtained from the external file. The differences between approaches that employed TF-IDF and bag-of-words feature extraction models were unremarkable. The n-gram sizes mattered more in terms of all metrics. 2-grams were the overall winners regardless of the feature extraction method. The values of precision, recall, and f1 score reached similar values as accuracy in both cases.

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.923	0.922	0.929	0.926
stemming 2-grams bag-of-words	0.947	0.939	0.958	0.948
stemming 3-grams bag-of-words	0.946	0.956	0.941	0.948
stemming 1-grams tf–idf	0.922	0.906	0.943	0.924
stemming 2-grams tf–idf	0.948	0.937	0.963	0.949
stemming 3-grams tf–idf	0.948	0.949	0.950	0.950
stemming Word2Vec	0.815	0.821	0.822	0.822
lemmatization 1-grams bag-of-words	0.927	0.925	0.935	0.930
lemmatization 2-grams bag-of-words	0.951	0.944	0.961	0.953
lemmatization 3-grams bag-of-words	0.945	0.955	0.940	0.948
lemmatization 1-grams tf–idf	0.926	0.920	0.937	0.928
lemmatization 2-grams tf–idf	0.954	0.944	0.966	0.955
lemmatization 3-grams tf–idf	0.950	0.947	0.956	0.952
lemmatization Word2Vec	0.815	0.819	0.825	0.822
pretrained GloVe	0.916	0.929	0.911	0.920

Table 4.2 Results of Naive Bayes on the ISOT dataset

The convolutional neural network model as depicted in Table 4.4 outperformed the other ones with both accuracy and f1 score over 97%, which was reached by utilising the pretrained GloVe embedding. The precision reached nearly 99%, which is a remarkable result. Word2Vec trained on the training set got slightly worse results, with accuracy slightly over 95%. Both stemming

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.921	0.985	0.882	0.931
stemming 2-grams bag-of-words	0.926	0.991	0.886	0.935
stemming 3-grams bag-of-words	0.843	0.995	0.776	0.872
stemming 1-grams tf–idf	0.925	0.987	0.887	0.934
stemming 2-grams tf–idf	0.927	0.993	0.885	0.936
stemming 3-grams tf–idf	0.845	0.995	0.779	0.874
stemming Word2Vec	0.847	0.903	0.828	0.864
lemmatization 1-grams bag-of-words	0.921	0.987	0.881	0.931
lemmatization 2-grams bag-of-words	0.920	0.990	0.877	0.930
lemmatization 3-grams bag-of-words	0.853	0.994	0.788	0.879
lemmatization 1-grams tf–idf	0.928	0.986	0.891	0.936
lemmatization 2-grams tf–idf	0.929	0.991	0.891	0.938
lemmatization 3-grams tf–idf	0.856	0.995	0.792	0.882
lemmatization Word2Vec	0.830	0.874	0.822	0.847
pretrained GloVe	0.926	0.929	0.934	0.931

Table 4.3 Results of random forest on the ISOT dataset

and lemmatization led to a slight decrease in performance in terms of accuracy, but the recall improved up to 98%.

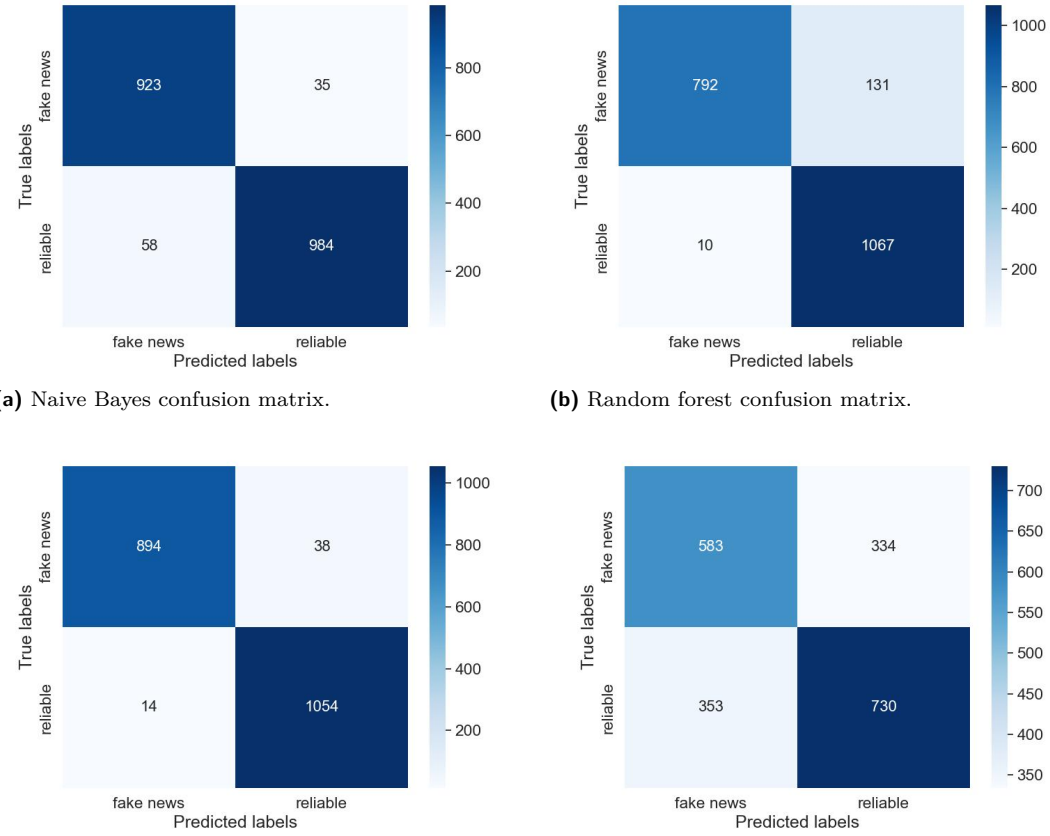
Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.955	0.939	0.975	0.957
stemming Word2Vec	0.952	0.934	0.976	0.954
lemmatization Word2Vec	0.947	0.919	0.981	0.949
pretrained GloVe	0.974	0.987	0.965	0.976

Table 4.4 Results of the convolutional neural network on the ISOT dataset

The final results that the LSTM neural network as depicted in Table 4.5 reached were much poorer than all of those mentioned earlier with only 65% accuracy. The highest accuracy was reached in combination with Word2Vec and without any text normalisation technique. This combination also reached the lowest precision among all preprocessings. The precision of the rest got to over 90%, unlike the recall that achieved similar smaller values as accuracy which means that the model did not classify much fake news as reliable but classified many reliable articles as fake news. Figure 4.3 displays the comparison of confusion matrices of each model on this dataset.

Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.656	0.674	0.686	0.680
stemming Word2Vec	0.524	0.912	0.536	0.675
lemmatization Word2Vec	0.541	1.000	0.541	0.703
pretrained GloVe	0.597	0.961	0.577	0.721

Table 4.5 Results of the LSTM neural network on the ISOT dataset



(a) Naive Bayes confusion matrix.

(b) Random forest confusion matrix.

(c) Convoulutional netural network confusion matrix

(d) LSTM netural network confusion matrix

Figure 4.3 Confusion matricies of classification models on the ISOT dataset

4.3.2 Results of the ReCOVery dataset

This dataset was smaller than the previous one, with 2,029 articles. It is also the least balanced one, with a ratio between reliable articles and fake news of 68:32. This was one of the reasons why the performances of all classifiers on this dataset were worse than on the previous dataset.

The imbalance and the size of this dataset led to machine learning approaches outperforming the neural networks. The results with the highest accuracy are depicted in Table 4.6. The Naive Bayes algorithm outperformed all of the others with an accuracy of 83%. No other algorithm could not overcome an accuracy of 80%. The metric that was truly remarkable in this case was precision. All of the algorithms could reach over 90% even 100% in the case of random forest and LSTM neural network.

Tables 4.7 and 4.8 depict the results of random forest and Naive Bayes algorithms. The Naive Bayes algorithm achieved overall much better results in terms of accuracy, but the random forest achieved an F1 score over 80% on each preprocessing. There were no significant differences among the feature extraction techniques in the experiments on machine learning models. The results of TF-IDF and BoW were similar. The sizes of n-grams could also lead to only a small improvement in accuracy. A significant difference can be observed in both algorithms between pretrained GloVe embedding and Word2Vec. In both cases, there was an improvement in the accuracy metric.

What stands out about the results of random forest is the precision that reached 100% for

Model	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.835	0.941	0.834	0.884
Random Forest	0.794	0.956	0.781	0.860
CNN	0.781	0.932	0.779	0.849
LSTM	0.695	0.927	0.709	0.803

Table 4.6 The best results of classification models in regard to classification accuracy of models on ReCOVery dataset.

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.796	0.890	0.820	0.854
stemming 2-grams bag-of-words	0.835	0.941	0.834	0.884
stemming 3-grams bag-of-words	0.820	0.890	0.849	0.869
stemming 1-grams tf–idf	0.796	0.901	0.814	0.855
stemming 2-grams tf–idf	0.823	0.945	0.818	0.877
stemming 3-grams tf–idf	0.815	0.901	0.836	0.867
stemming Word2Vec	0.660	0.812	0.718	0.762
lemmatization 1-grams bag-of-words	0.803	0.912	0.816	0.861
lemmatization 2-grams bag-of-words	0.833	0.938	0.833	0.882
lemmatization 3-grams bag-of-words	0.830	0.904	0.851	0.877
lemmatization 1-grams tf–idf	0.808	0.923	0.815	0.866
lemmatization 2-grams tf–idf	0.823	0.934	0.825	0.876
lemmatization 3-grams tf–idf	0.818	0.908	0.834	0.870
lemmatization Word2Vec	0.702	0.871	0.734	0.797
pretrained GloVe	0.724	0.754	0.820	0.785

Table 4.7 Results of Naive Bayes on the ReCOVery dataset

almost all preprocessing techniques despite recall achieving a maximum of 78%. These two metrics were more balanced in the case of the Naive Bayes classifier where recall almost every time overcame 80%.

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.694	1.000	0.683	0.812
stemming 2-grams bag-of-words	0.698	1.000	0.686	0.814
stemming 3-grams bag-of-words	0.713	1.000	0.697	0.821
stemming 1-grams tf–idf	0.689	1.000	0.680	0.809
stemming 2-grams tf–idf	0.706	1.000	0.692	0.818
stemming 3-grams tf–idf	0.713	1.000	0.697	0.821
stemming Word2Vec	0.695	0.999	0.685	0.812
lemmatization 1-grams bag-of-words	0.692	1.000	0.682	0.811
lemmatization 2-grams bag-of-words	0.704	1.000	0.690	0.817
lemmatization 3-grams bag-of-words	0.715	1.000	0.699	0.823
lemmatization 1-grams tf–idf	0.692	1.000	0.682	0.811
lemmatization 2-grams tf–idf	0.703	1.000	0.690	0.816
lemmatization 3-grams tf–idf	0.718	1.000	0.701	0.824
lemmatization Word2Vec	0.694	0.999	0.683	0.812
pretrained GloVe	0.794	0.956	0.781	0.860

Table 4.8 Results of random forest on the ReCOVery dataset

The performance of the convolutional neural network is presented in Table 4.9. We can see that it reached a bit worse accuracy than the two preceding models. The pretrained GloVe in this case outperformed the other feature extraction models among all evaluation metrics except for recall. Remarkable is also the difference in both accuracy and precision between stemming and lemmatization in combination with word2vec in which lemmatization led to significantly better outcomes.

Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.726	0.692	0.872	0.763
stemming Word2Vec	0.717	0.675	0.866	0.759
lemmatization Word2Vec	0.754	0.821	0.813	0.814
pretrained GloVe	0.781	0.932	0.779	0.849

Table 4.9 Results of the convolutional neural network on the ReCOVery dataset

The LSTM neural network reached slightly better accuracy on this dataset than on the previous one as portrayed in Table 4.10. The highest precision was reached by the pretrained GloVe word embedding it was also the only preprocessing that did not end up with 100% precision. The results of the experiments with Word2Vec were practically identical regardless of the text normalization technique. Figure 4.4 displays the comparison of confusion matrices of each model on ReCOVery dataset.

Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.672	1.000	0.672	0.804
stemming Word2Vec	0.672	1.000	0.672	0.804
lemmatization Word2Vec	0.672	1.000	0.672	0.804
pretrained GloVe	0.695	0.927	0.709	0.803

Table 4.10 Results of the LSTM neural network on the ReCOVery dataset

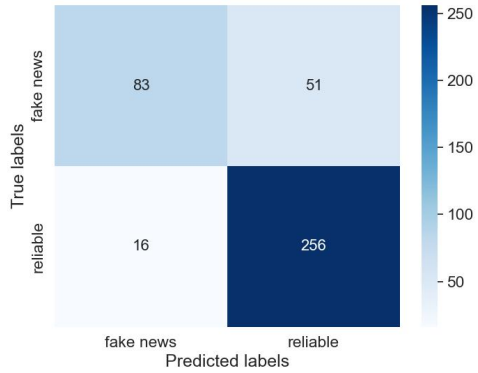
4.3.3 Results of the EUvsDisinfo dataset

Although the size of this dataset was similar to the previous one, there were many key differences. The dataset was more balanced, with a ratio of 50:50 between fake and reliable articles. The articles consisted of short summaries instead of full articles, which meant restricting the maximal length to 50 words only. This surprisingly led to better results than on the previous dataset. Table 4.11 depicts the overall results regarding the highest accuracy.

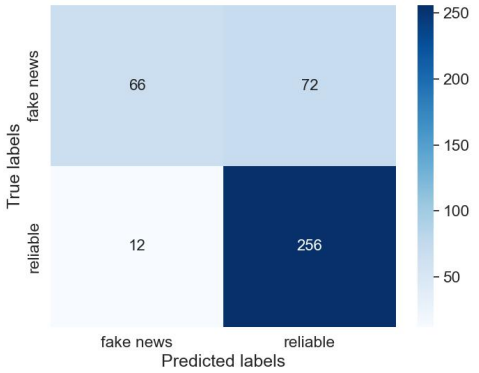
Model	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.909	0.947	0.875	0.910
Random Forest	0.941	0.936	0.939	0.937
CNN	0.945	0.961	0.935	0.947
LSTM	0.951	0.933	0.966	0.949

Table 4.11 The best results of classification models in regard to classification accuracy of models on EUvsDisinfo dataset.

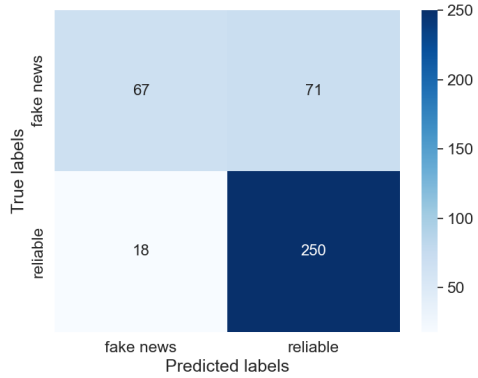
Tables 4.12 and 4.13 outline the results of the Naive Bayes and random forest classifiers. The random forest algorithm performed slightly better in terms of overall accuracy. For the random forest, the sizes of n-grams mattered much more than the feature extraction algorithms.



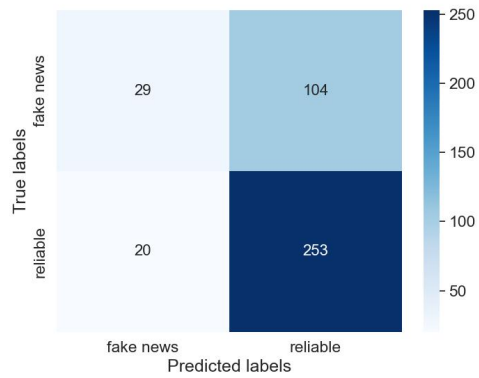
(a) Naive Bayes confusion matrix



(b) Random forest confusion matrix



(c) Convoulutional netural network confusion matrix



(d) LSTM netural network confusion matrix

Figure 4.4 Confusion matricies of classification models on the ReCOVery dataset

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.832	0.884	0.792	0.836
stemming 2-grams bag-of-words	0.897	0.932	0.865	0.898
stemming 3-grams bag-of-words	0.879	0.966	0.816	0.885
stemming 1-grams tf–idf	0.829	0.870	0.796	0.831
stemming 2-grams tf–idf	0.900	0.913	0.883	0.898
stemming 3-grams tf–idf	0.881	0.937	0.836	0.884
stemming Word2Vec	0.673	0.812	0.625	0.706
lemmatization 1-grams bag-of-words	0.827	0.889	0.783	0.833
lemmatization 2-grams bag-of-words	0.909	0.947	0.875	0.910
lemmatization 3-grams bag-of-words	0.888	0.961	0.833	0.892
lemmatization 1-grams tf–idf	0.846	0.879	0.816	0.847
lemmatization 2-grams tf–idf	0.907	0.923	0.888	0.905
lemmatization 3-grams tf–idf	0.895	0.932	0.862	0.896
lemmatization Word2Vec	0.668	0.812	0.620	0.703
pretrained GloVe	0.867	0.889	0.844	0.866

Table 4.12 Results of Naive Bayes on the EUvsDisinfo dataset

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.941	0.936	0.939	0.937
stemming 2-grams bag-of-words	0.926	0.893	0.948	0.919
stemming 3-grams bag-of-words	0.769	0.578	0.898	0.703
stemming 1-grams tf–idf	0.937	0.917	0.947	0.932
stemming 2-grams tf–idf	0.919	0.898	0.929	0.913
stemming 3-grams tf–idf	0.778	0.594	0.903	0.716
stemming Word2Vec	0.774	0.792	0.746	0.768
lemmatization 1-grams bag-of-words	0.935	0.914	0.945	0.929
lemmatization 2-grams bag-of-words	0.926	0.896	0.944	0.920
lemmatization 3-grams bag-of-words	0.780	0.587	0.915	0.716
lemmatization 1-grams tf–idf	0.938	0.917	0.951	0.934
lemmatization 2-grams tf–idf	0.921	0.903	0.929	0.915
lemmatization 3-grams tf–idf	0.780	0.586	0.917	0.715
lemmatization Word2Vec	0.798	0.818	0.768	0.792
pretrained GloVe	0.883	0.843	0.903	0.872

Table 4.13 Results of random forest on the EUvsDisinfo dataset

1 and 2-grams achieved much better accuracy than 3-grams in combination with both BoW and TF-IDF. In contrast, there were no significant differences between stemming and lemmatization.

The sizes of n-grams also highly influenced the precision of the random forest, which ended up being less than 60% in the case of 3-grams. The Naive Bayes classifier did not suffer from this as much as the random forest. The 3-grams reached the highest precision of 96%.

The differences between Word2Vec and GloVe were remarkable in both models. On Naive Bayes, Word2Vec could hardly reach 70% accuracy, whereas GloVe scored almost 90%. In the case of random forest, the gap between those two techniques was narrower but still notable.

The convolutional neural network achieved a bit higher accuracy than the preceding models. As presented in Table 4.14 pretrained GloVe outperformed Word2Vec in terms of all the evaluation metrics, with an accuracy of 94% and precision of 96%. There is also a slight difference in accuracy and recall between utilising stemming and lemmatization in combination with Word2Vec. Lemmatization helped to enhance the performance a bit.

Table 4.15 describes the performance of the LSTM neural network that achieved the best results among all four tested datasets. The accuracy of 95% that was achieved using the pretrained GloVe embedding was the highest among all models. The difference in accuracy between Word2Vec and GloVe is really significant, as well as the difference in accuracy between applying stemming and lemmatization in combination with Word2Vec and no text normalisation. The accuracy of Word2Vec without any text normalisation technique came to only 55%, even poorer than the accuracy of this model on this dataset.

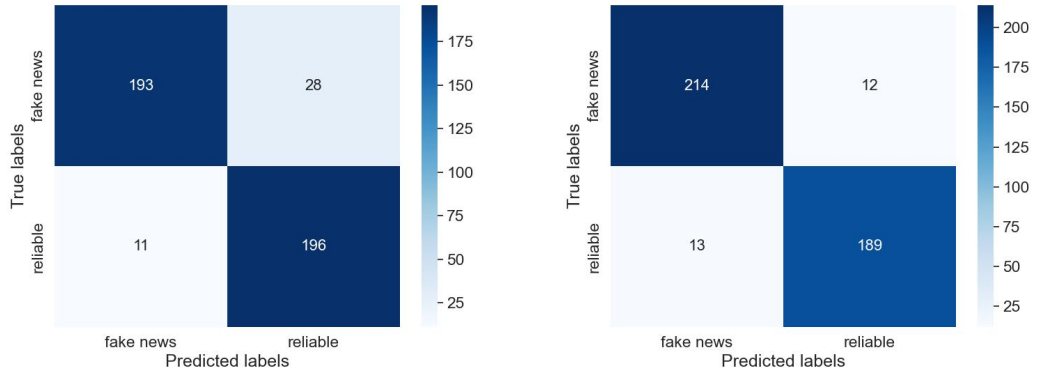
The pretrained word embedding also outperformed the other preprocessing in terms of precision and recall, with precision of 93% and recall of 96%. However, the differences in precision are less significant compared to the differences in recall. Figure 4.4 displays the comparison of confusion matrices of each model on this dataset.

Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.872	0.863	0.887	0.875
stemming Word2Vec	0.873	0.868	0.886	0.876
lemmatization Word2Vec	0.891	0.863	0.922	0.891
pretrained GloVe	0.945	0.961	0.935	0.947

Table 4.14 Results of the convolutional neural network on the EUvsDisinfo dataset

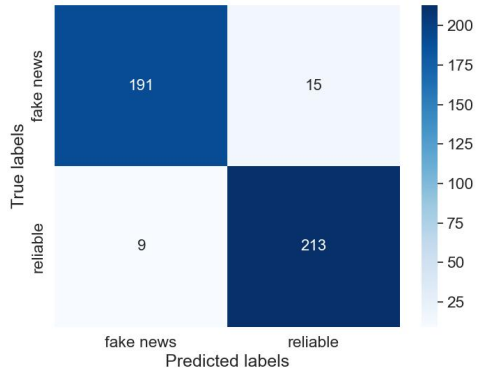
Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.554	0.900	0.526	0.664
stemming Word2Vec	0.757	0.814	0.725	0.767
lemmatization Word2Vec	0.799	0.805	0.790	0.797
pretrained GloVe	0.951	0.933	0.966	0.949

Table 4.15 Results of the LSTM neural network on the EUvsDisinfo dataset

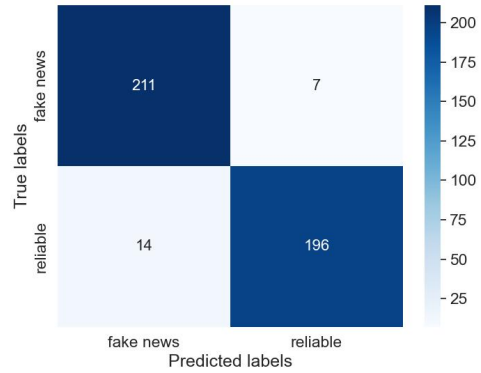


(a) Naive Bayes confusion matrix

(b) Random forest confusion matrix



(c) Convoulutional netural network confusion matrix



(d) LSTM netural network confusion matrix

Figure 4.5 Confusion matricies of classification models on the EuvsDisinfo dataset

4.3.4 Results of the Czech dataset

This dataset was the smallest of those used for the experiments, with only 995 articles. It was also a bit imbalanced, with a ratio between reliable and fake news of 59:41. As depicted in Table 4.16, the models scored poorer in terms of accuracy than on the previous EUvsDisinfo dataset but slightly better than on the ReCOVery dataset. The smaller size of the dataset led to the Naive Bayes classifier achieving higher accuracy than the more complex ones.

Tables 4.17 and 4.18 depict the results of Naive Bayes and random forest classifiers. The influence of BoW and TF-IDF on any of the metrics is as insignificant as in the previous dataset. The results were much more affected by the size of n-grams. 3-grams came to be the worst choice

Model	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.899	0.907	0.922	0.915
Random Forest	0.831	1.000	0.778	0.875
CNN	0.801	0.812	0.831	0.821
LSTM	0.749	0.672	0.903	0.771

Table 4.16 The best results of classification models in regard to the accuracy of models on the Czech dataset

while 2-grams scored the best accuracy. Stemming and lemmatization did not have any serious impact on the performance. The results of Word2Vec and pretrained GloVe were comparable among the metrics, with Word2Vec reaching slightly better accuracy on both models.

Similarly to the ReCOVery dataset, the precision of the random forest reached 100% many times. However, the recall did never score more than 80%. Precision and recall of the Naive Bayes were more balanced, with both metrics reaching over 90% except for Word2Vec and GloVe.

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.854	0.847	0.901	0.873
stemming 2-grams bag-of-words	0.899	0.907	0.922	0.915
stemming 3-grams bag-of-words	0.829	0.949	0.800	0.868
stemming 1-grams tf–idf	0.824	0.797	0.895	0.843
stemming 2-grams tf–idf	0.864	0.847	0.917	0.881
stemming 3-grams tf–idf	0.844	0.915	0.837	0.874
stemming Word2Vec	0.744	0.644	0.894	0.749
lemmatization 1-grams bag-of-words	0.834	0.805	0.905	0.852
lemmatization 2-grams bag-of-words	0.889	0.924	0.893	0.908
lemmatization 3-grams bag-of-words	0.874	0.941	0.860	0.899
lemmatization 1-grams tf–idf	0.814	0.780	0.893	0.833
lemmatization 2-grams tf–idf	0.874	0.898	0.891	0.895
lemmatization 3-grams tf–idf	0.874	0.932	0.866	0.898
lemmatization Word2Vec	0.764	0.661	0.918	0.768
pretrained GloVe	0.749	0.678	0.870	0.762

Table 4.17 Results of Naive Bayes on the Czech dataset dataset.

The results of CNN, as presented in Table 4.19, were slightly worse than the results of the random forest. Here, Word2Vec outperformed the pretrained GloVe in terms of all metrics, even though the difference is not too big. The difference between stemming and lemmatization in combination with Word2Vec is insignificant, but Word2Vec could reach the highest precision of 85% without any text normalisation. In terms of recall, the results were similar to both of the preceding metrics.

The final results of the LSTM network can be observed in Table 4.20. We can see that in this case, the pretrained GloVe embedding outperformed Word2Vec in terms of accuracy. However, its precision reached only 67% which is notably poorer than 100% achieved by Word2Vec in combination with both stemming and lemmatization. In contrast, a recall of 90% was achieved by the GloVe embedding, which is remarkably higher than 64% of Word2Vec without stemming and lemmatization. Despite the differences between those two metrics, the F1 score reached almost identical values among all preprocessing techniques. Figure 4.5 displays the comparison of confusion matrices of each model on this dataset.

Preprocessing	Accuracy	Precision	Recall	F1-Score
stemming 1-grams bag-of-words	0.812	0.994	0.762	0.863
stemming 2-grams bag-of-words	0.819	1.000	0.766	0.868
stemming 3-grams bag-of-words	0.700	1.000	0.664	0.798
stemming 1-grams tf–idf	0.807	0.992	0.758	0.859
stemming 2-grams tf–idf	0.831	1.000	0.778	0.875
stemming 3-grams tf–idf	0.688	0.992	0.657	0.791
stemming Word2Vec	0.812	0.932	0.790	0.855
lemmatization 1-grams bag-of-words	0.816	0.994	0.765	0.865
lemmatization 2-grams bag-of-words	0.822	0.989	0.774	0.869
lemmatization 3-grams bag-of-words	0.709	1.000	0.670	0.803
lemmatization 1-grams tf–idf	0.806	0.989	0.758	0.858
lemmatization 2-grams tf–idf	0.827	0.997	0.776	0.873
lemmatization 3-grams tf–idf	0.707	1.000	0.669	0.802
lemmatization Word2Vec	0.802	0.929	0.780	0.848
pretrained GloVe	0.799	0.935	0.773	0.846

Table 4.18 Results of random forest on the Czech dataset

Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.767	0.857	0.761	0.805
stemming Word2Vec	0.797	0.798	0.835	0.816
lemmatization Word2Vec	0.801	0.812	0.831	0.821
pretrained GloVe	0.747	0.807	0.759	0.782

Table 4.19 Results of the convolutional neural network on the Czech dataset

Preprocessing	Accuracy	Precision	Recall	F1-Score
Word2Vec	0.648	0.976	0.646	0.777
stemming Word2Vec	0.628	1.000	0.628	0.772
lemmatization Word2Vec	0.628	1.000	0.628	0.772
pretrained GloVe	0.749	0.672	0.903	0.771

Table 4.20 Results of the LSTM neural network on the Czech dataset

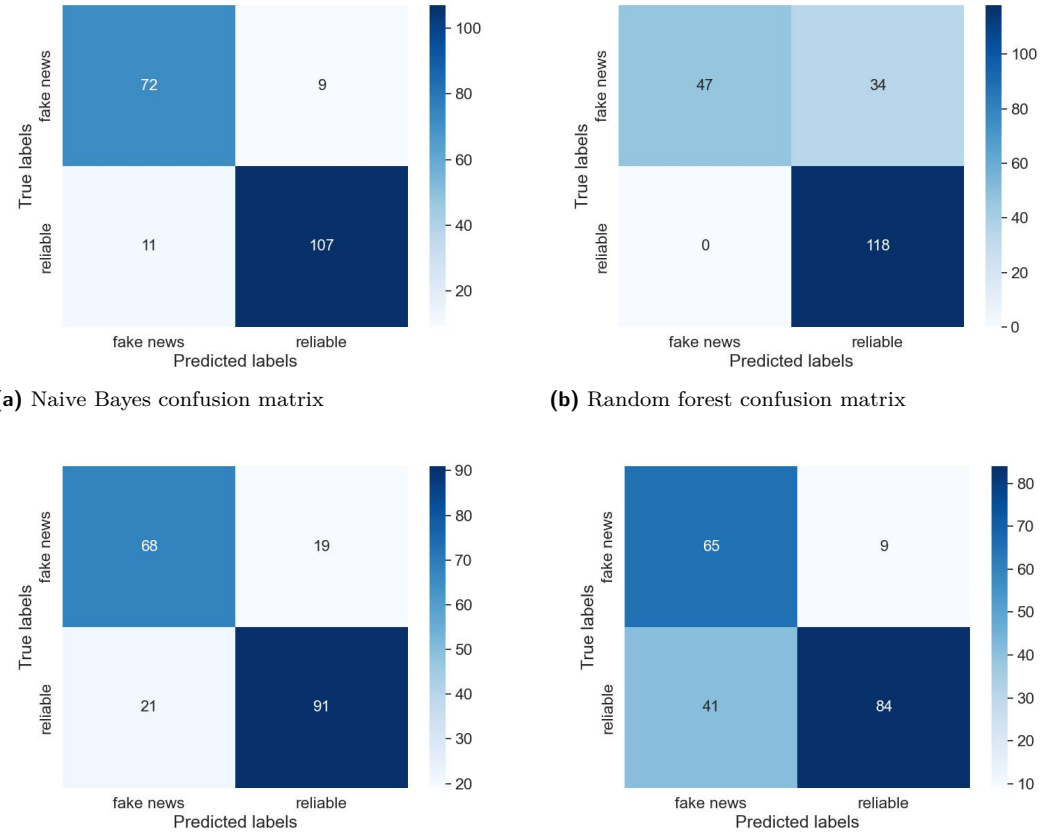
4.3.5 Discussion

The results on all of the test datasets show that the size of the dataset is a significant factor in the quality of the classification factor as well as the ratio of reliable to fake news articles. The imbalanced dataset, in combination with a more conservative model, can spike the difference between recall and precision.

Another important factor in the quality of classification is the length of the classified articles. We saw that even with a smaller dataset, relatively high performance can be achieved in the case of shorter articles. On the other hand, when classifying longer articles, a bigger dataset led to better performance.

Each utilised model was able to produce promising outcomes. We saw that in the case of imbalanced datasets, better results were produced by the machine learning models. Especially Naive Bayes could outperform the other ones on both the ReCOVery and EUvsDisinfo datasets. This classifier could also soften the difference between recall while others could not.

In the case of the experiments performed on more balanced datasets, more robust techniques



(a) Naive Bayes confusion matrix

(b) Random forest confusion matrix

(c) Convoulutional netural network confusion matrix

(d) LSTM netural network confusion matrix

Figure 4.6 Confusion matricies of classification models on the Czech dataset

were more effective than the others. This architecture of the convolutional neural network achieved very good performance on both of the balanced datasets. The simple LSTM architecture turned out to be useful for shorter statements rather than longer articles.

Different text preprocessing approaches also influenced the results of the classification. The role of pretrained word embedding in the classification is significant. We saw that among the neural network models, performance could be notably enhanced. On the other hand, the machine learning algorithms got better results by employing bag-of-words, or TF-IDF, most of the time. For the machine learning models, more significant than the feature extraction was the size of the n-grams which could have a drastic impact on all metrics. The impact of stemming and lemmatization, on the other hand, was very limited.

One way to improve the performance would be to collect a larger dataset. We saw that even 10,000 articles can lead to very good results. We also saw a performance gap between the balanced and imbalanced datasets. So, to enhance the performance, new articles on related issues would be obtained.

4.4 Chapter summary

In this chapter, we describe the tools and libraries used to implement the program for performing the experiments on the datasets described in Chapter 3. Then we presented the implementation

details, including the structure of the program. The description of models and neural network architectures used in the experiments follows the list of hyperparameters that were tuned during the training.

Finally, we presented the evaluated experiments and discussed the results. The implemented classification models could reach up to 97% accuracy on a larger already-made dataset and over 95% on a smaller self-obtained one. The most efficient classification models were those based on the neural network, but the machine learning ones could also reach remarkable results.

Chapter 5
Conclusion

The goal of this thesis was to analyse, implement, and compare several existing machine learning approaches for fake news classification using natural language processing. Firstly, we researched existing academic literature regarding this topic in Chapter 2. The aim of this chapter was to give the reader an introduction to text classification methods and to create a summary of frequently used terms in this thesis. It also contains a description of the selected models and methods of natural language processing implemented in the last chapter. In the third chapter, we created a survey of existing datasets of fake news articles and created new experimental ones in both the Czech and English languages. Finally, we performed multiple experiments to find which combination of text preprocessing and classification model was the most efficient.

One of the aims of this thesis was to create a brand new fake news dataset using online fake news databases. The one dataset presented in English comes from the EU project EUvsDisinfo and consists of over 1,000 summaries of fake news texts, which were combined with over 1,000 short, reliable articles from a serious media outlet.

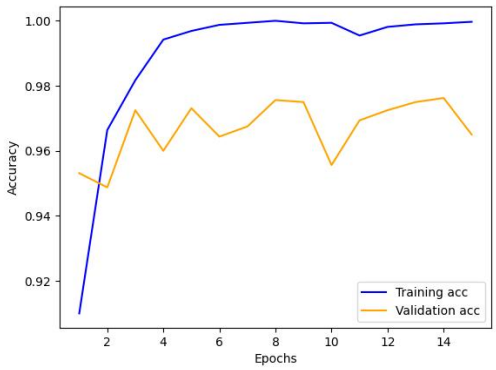
This objective included obtaining a brand new dataset in Czech. This was achieved by extracting over 400 fake news mails and articles from the database of chain emails Eldariel, which is a project of a Czech citizen initiative called Čeští elfové. The purpose of this project is to track fake news in the Czech Republic. The obtained content was combined with articles from a serious Czech media outlet and used for the experiments.

Numerous experiments were performed on four datasets with two machine learning and two neural network classifiers. Overall, the best performance was achieved by the convolutional neural network in combination with pretrained GloVe embedding on the largest examined ISOT dataset, with an accuracy of over 97%. The results of experiments on the smaller self-obtained dataset of summaries of articles were also remarkable, with an accuracy of over 95% that was attained by a simple LSTM neural network in combination with the pretrained GloVe embedding. Even though the highest accuracy was attained by the neural networks, the traditional machine learning models still achieved comparable results, especially on imbalanced datasets.

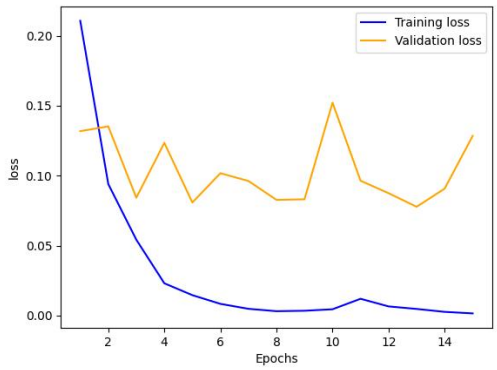
The goals and subgoals of this thesis are considered fulfilled. However, there are still many possibilities for improvement. For example, a more robust dataset could be obtained with a more granular distinction of the level of disinformation in the texts. Also, satirical texts could be added to the datasets. The neural network architectures used in this thesis could also be part of some complex real-world program, where not only text but also other fake news media such as images, videos can be classified.

Appendix A

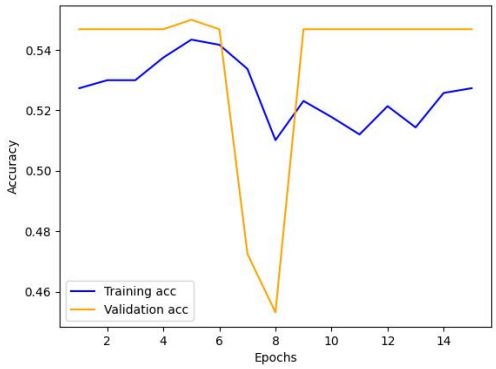
Training performance of the neural networks



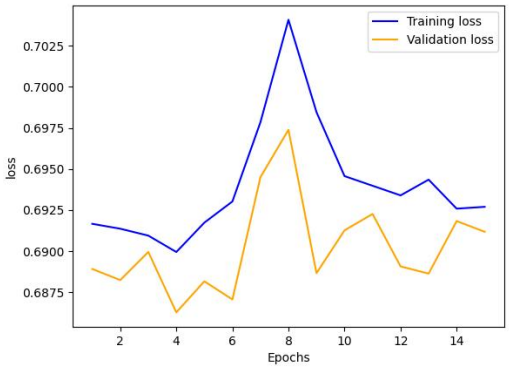
(a) The accuracy of the CNN per epoch.



(b) The loss of the CNN per epoch.

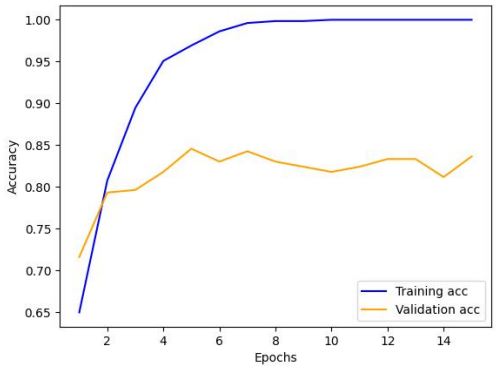


(c) The accuracy of the LSTM neural network per epoch.

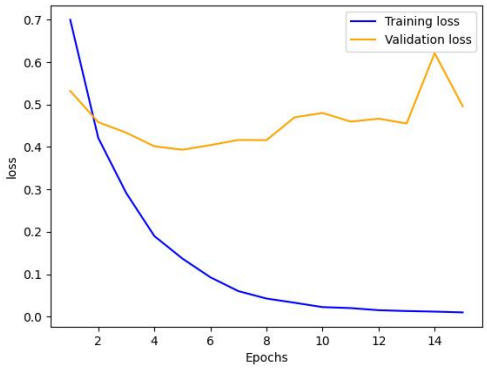


(d) The loss of the LSTM neural network per epoch.

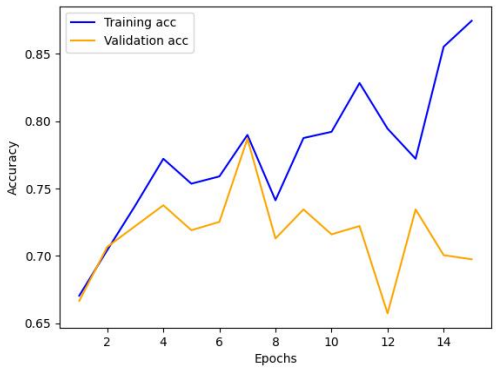
Figure A.1 Examples of training performance of the neural networks per epoch on the ISOT dataset.



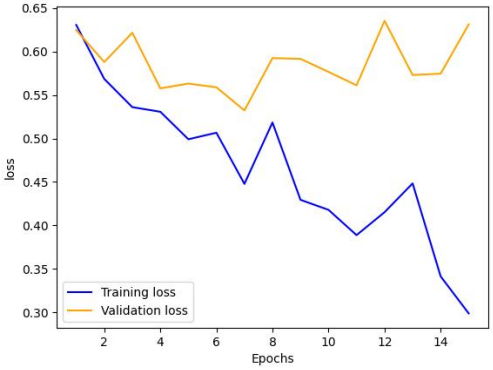
(a) The accuracy of the CNN per epoch.



(b) The loss of the CNN per epoch.

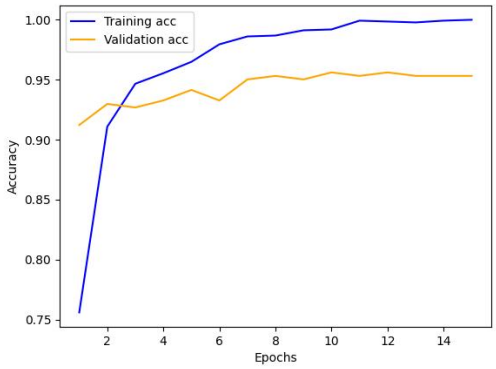


(c) The accuracy of the LSTM neural network per epoch.

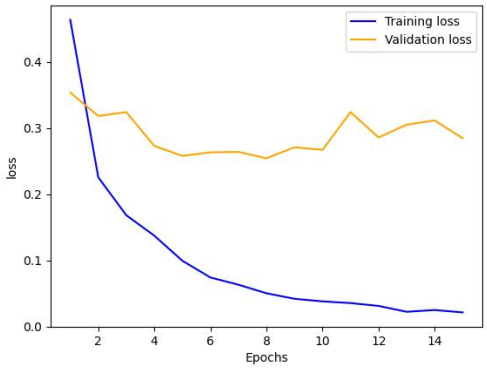


(d) The loss of the LSTM neural network per epoch.

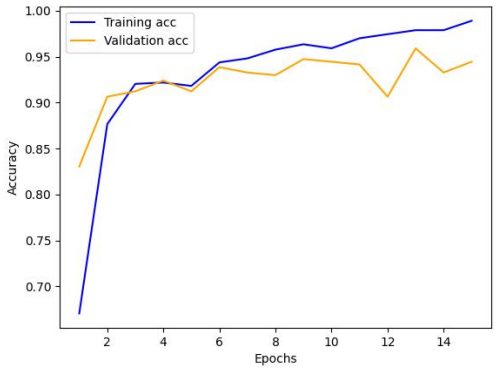
Figure A.2 Examples of training performance of the neural networks per epoch on the ReCOVery dataset.



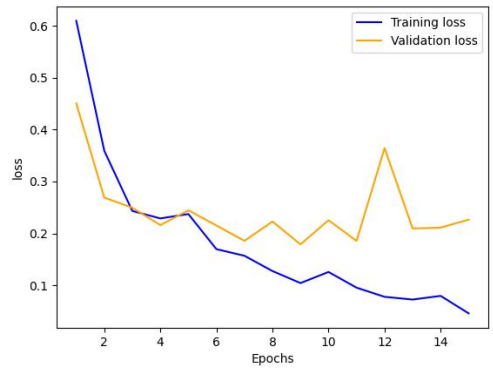
(a) The accuracy of the CNN per epoch.



(b) The loss of the CNN per epoch.

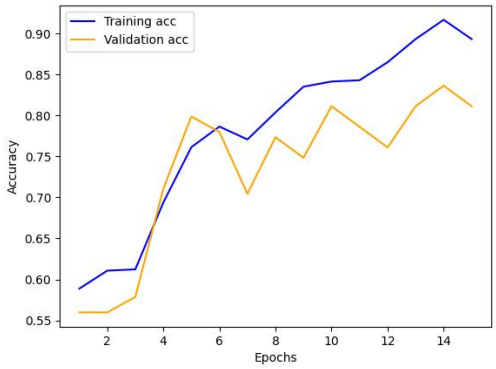


(c) The accuracy of the LSTM neural network per epoch.

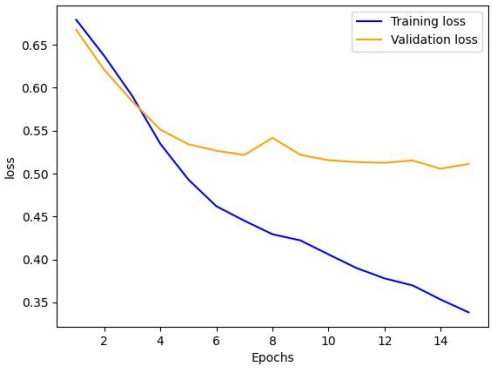


(d) The loss of the LSTM neural network per epoch.

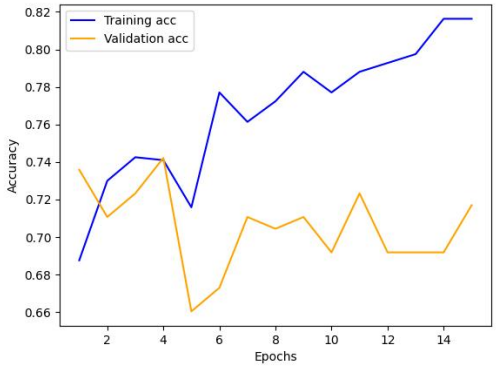
Figure A.3 Examples of training performance of the neural networks per epoch on EuvsDisinfo dataset.



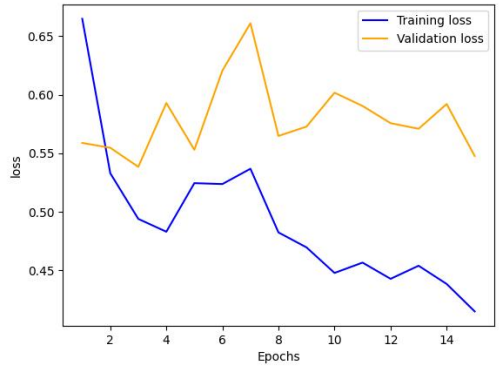
(a) The accuracy of the CNN per epoch.



(b) The loss of the CNN per epoch.



(c) The accuracy of the LSTM neural network per epoch.



(d) The loss of the LSTM neural network per epoch.

Figure A.4 Examples of training performance of the neural networks per epoch on the Czech dataset.

Bibliography

1. SILVERMAN, C.; JEREMY, Singer-Vine. *Most Americans Who See Fake News Believe It, New Survey Says* [online]. BuzzFeedNews, 2016 [visited on 2023-04-21]. Available from: https://www.buzzfeednews.com/article/craigsilverman/fake-news-survey.
2. SHU, Kai; SLIVA, Amy; WANG, Suhang; TANG, Jiliang; LIU, Huan. *Fake News Detection on Social Media: A Data Mining Perspective* [online]. 2017. [visited on 2023-04-21]. Available from arXiv: 1708.01967 [cs.SI].
3. COMMISSION, Europan. *Tackling online disinformation* [online]. 2020. [visited on 2023-04-21]. Available from: https://digital-strategy.ec.europa.eu/en/policies/online-disinformation.
4. IRETON CHERILYN Posetti Julie, Unesco. *Journalism, fake news & disinformation : handbook for journalism education and training / Cherilyn Ireton and Julie Posetti* [online]. Paris: UNESCO, 2018 [visited on 2023-04-21]. ISBN 978-9231002816. Available from: https://digitallibrary.un.org/record/1641987.
5. HAMELEERS, Michael; POWELL, Thomas E.; MEER, Toni G.L.A. Van Der; BOS, Lieke. A Picture Paints a Thousand Lies? The Effects and Mechanisms of Multimodal Disinformation and Rebuttals Disseminated via Social Media. *Political Communication* [online]. 2020, vol. 37, no. 2, pp. 281–301 [visited on 2023-04-21]. Available from DOI: 10.1080/10584609.2019.1674979.
6. THOMPSON, Robyn C.; JOSEPH, Seena; ADELIYI, Timothy T. A Systematic Literature Review and Meta-Analysis of Studies on Online Fake News Detection. *Information* [online]. 2022, vol. 13, no. 11 [visited on 2023-04-21]. ISSN 2078-2489. Available from DOI: 10.3390/info13110527.
7. VOICHICI, Oana. Internet Hoaxes: Chain Letters – How Dangerous Are They? [online]. 2020, pp. 219–229 [visited on 2023-04-21]. ISBN 978-606-14-1417-8. Available from: https://www.researchgate.net/publication/344339739_Internet_Hoaxes_Chain_Letters_-_How_Dangerous_Are_They.
8. CHEN, Yimin; CONROY, Nadia K.; RUBIN, Victoria L. News in an online world: The need for an "automatic crap detector". *Proceedings of the Association for Information Science and Technology* [online]. 2015, vol. 52, no. 1, pp. 1–4 [visited on 2023-04-22]. Available from DOI: https://doi.org/10.1002/pra2.2015.145052010081.
9. SHU, Kai; SLIVA, Amy; WANG, Suhang; TANG, Jiliang; LIU, Huan. *Fake News Detection on Social Media: A Data Mining Perspective* [online]. 2017. [visited on 2023-04-22]. Available from arXiv: 1708.01967 [cs.SI].

10. ISLAM, Taminul; HOSEN, Md; MONY, Akhi; HASAN, MD; JAHAN, Israt; KUNDU, Arindom. A Proposed Bi-LSTM Method to Fake News Detection. In: 2022. Available from DOI: 10.1109/ICONAT53423.2022.9725937.

11. GILDA, Shlok. Evaluating machine learning algorithms for fake news detection. In: [online]. 2017, pp. 110–115 [visited on 2023-04-22]. Available from DOI: 10.1109/SCORED.2017.8305411.

12. CORNEY, D.; ALBAKOUR, M-Dyaa; MARTINEZ-ALVAREZ, Miguel; MOUSSA, Samir. What do a Million News Articles Look like? In: *NewsIR@ECIR* [online]. 2016 [visited on 2023-04-22]. Available from: https://ceur-ws.org/Vol-1568/paper8.pdf.

13. AHMED, Hadeer; TRAORE, Issa; SAAD, Sherif. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: 2017, pp. 127–138. ISBN 978-3-319-69154-1. Available from DOI: 10.1007/978-3-319-69155-8_9.

14. KHAN, Junaed Younus; KHONDAKER, Md. Tawkat Islam; AFROZ, Sadia; UDDIN, Gias; IQBAL, Anindya. A benchmark study of machine learning models for online fake news detection. *Machine Learning with Applications* [online]. 2021, vol. 4, p. 100032 [visited on 2023-04-22]. ISSN 2666-8270. Available from DOI: https://doi.org/10.1016/j.mlwa.2021.100032.

15. WANG, William Yang. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. *CoRR*. 2017, vol. abs/1705.00648. Available from arXiv: 1705.00648.

16. GHANEM, Bilal; ROSSO, Paolo; RANGEL, Francisco. Stance Detection in Fake News A Combined Feature Representation. In: THORNE, James; VLACHOS, Andreas; COCARASCU, Oana; CHRISTODOULOPOULOS, Christos; MITTAL, Arpit (eds.). *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 66–71. Available from DOI: 10.18653/v1/W18-5510.

17. KIDU, Hailay; MISGNA, Haile; LI, Tong; YANG, Zhen. User Response-Based Fake News Detection on Social Media. In: FLOREZ, Hector; POLLO-CATTANEO, Ma Florencia (eds.). *Applied Informatics* [online]. Cham: Springer International Publishing, 2021, pp. 173–187 [visited on 2023-04-22]. ISBN 978-3-030-89654-6. Available from DOI: https://doi.org/10.1007/978-3-030-89654-6_13.

18. *BI-ML1.21 přednáška 2 handout 5. prosince 2022* [online]. CTU, Faculty of Information Technology, 2022. Available also from: https://courses.fit.cvut.cz/BI-ML1/lectures/files/BI-ML1-02-cs-handout.pdf.

19. JURAFSKY, Daniel; MARTIN, James H. *Naive Bayes and Sentiment Classification* [online]. Stanford university, 2023. Available also from: https://stanford.edu/~jurafsky/slp3/4.pdf.

20. GASPARETTO, Andrea; MARCUZZO, Matteo; ZANGARI, Alessandro; ALBARELLI, Andrea. A Survey on Text Classification Algorithms: From Text to Predictions. *Information*. 2022, vol. 13, no. 2. ISSN 2078-2489. Available from DOI: 10.3390/info13020083.

21. ALLEN, James F. Natural Language Processing. In: *Encyclopedia of Computer Science*. GBR: John Wiley and Sons Ltd., 2003, pp. 1218–1222. ISBN 0470864125.

22. FENG, Song; BANERJEE, Ritwik; CHOI, Yejin. Syntactic Stylometry for Deception Detection. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 171–175. Available also from: https://aclanthology.org/P12-2034.

23. KOWSARI, Kamran; JAFARI MEIMANDI, Kiana; HEIDARYSAFA, Mojtaba; MENDU, Sanjana; BARNES, Laura; BROWN, Donald. Text Classification Algorithms: A Survey. *Information*. 2019, vol. 10, no. 4. ISSN 2078-2489. Available from DOI: 10.3390/info10040150.

24. PUNDE, Abhishek; RAMTEKE, Sanchit; SHINDE, Shailesh; KOLTE, Shilpa. Fake Product Review Monitoring & Removal and Sentiment Analysis of Genuine Reviews. *International Journal of Engineering and Management Research*. 2019, vol. 9, pp. 107–110. Available from DOI: 10.31033/ijemr.9.2.12.

25. MRIDHA, M. F.; KEYA, Ashfia Jannat; HAMID, Md. Abdul; MONOWAR, Muhammad Mostafa; RAHMAN, Md. Saifur. A Comprehensive Review on Fake News Detection With Deep Learning. *IEEE Access*. 2021, vol. 9, pp. 156151–156170. Available from DOI: 10.1109/ACCESS.2021.3129329.

26. Optimization and improvement of fake news detection using deep learning approaches for societal benefit. *International Journal of Information Management Data Insights*. 2021, vol. 1, no. 2, p. 100051. ISSN 2667-0968. Available from DOI: https://doi.org/10.1016/j.jjimei.2021.100051.

27. KHYANI, Divya; B S, Siddhartha. An Interpretation of Lemmatization and Stemming in Natural Language Processing. *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology*. 2021, vol. 22, pp. 350–357.

28. QADER, Wisam; M. AMEEN, Musa; AHMED, Bilal. An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges. In: 2019, pp. 200–204. Available from DOI: 10.1109/IEC47844.2019.8950616.

29. *BI-VZD přednáška 12 handout 21. dubna 2021* [online]. CTU, Faculty of Information Technology, 2021. Available also from: https://kam.fit.cvut.cz/bi-vzd/lectures/files/BI-VZD-12-cs-handout.pdf.

30. QAISER, Shahzad; ALI, Ramsha. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International Journal of Computer Applications*. 2018, vol. 181. Available from DOI: 10.5120/ijca2018917395.

31. GHANNAY, Sahar; FAVRE, Benoit; ESTÈVE, Yannick; CAMELIN, Nathalie. Word Embedding Evaluation and Combination. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), 2016, pp. 300–305. Available also from: https://aclanthology.org/L16-1046.

32. LAI, Siwei; LIU, Kang; XU, Liheng; ZHAO, Jun. *How to Generate a Good Word Embedding?* 2015. Available from arXiv: 1507.05523 [cs.CL].

33. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. *Efficient Estimation of Word Representations in Vector Space*. 2013. Available from arXiv: 1301.3781 [cs.CL].

34. RATHI, R.N.; MUSTAFI, A. The importance of Term Weighting in semantic understanding of text: A review of techniques. *Multimedia Tools and Applications*. 2023, vol. 82, pp. 9761–9783. Available from DOI: 10.1007/s11042-022-12538-3.

35. TRUICĂ, Ciprian-Octavian; APOSTOL, Elena-Simona. It's All in the Embedding! Fake News Detection Using Document Embeddings. *Mathematics*. 2023, vol. 11, no. 3. ISSN 2227-7390. Available from DOI: 10.3390/math11030508.

36. PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. GloVe: Global Vectors for Word Representation. In: *Empirical Methods in Natural Language Processing (EMNLP)* [online]. 2014, pp. 1532–1543 [visited on 2023-04-22]. Available from: http://www.aclweb.org/anthology/D14-1162.

37. SESARI, Emeralda; HORT, Max; SARRO, Federica. An Empirical Study on the Fairness of Pre-trained Word Embeddings. In: HARDMEIER, Christian; BASTA, Christine; COSTA-JUSSÀ, Marta R.; STANOVSKY, Gabriel; GONEN, Hila (eds.). *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*. Seattle, Washington: Association for Computational Linguistics, 2022, pp. 129–144. Available from DOI: 10.18653/v1/2022.gebnlp-1.15.

38. *BI-ML1.21 přednáška 4 handout 26. října 2023* [online]. CTU, Faculty of Information Technology, 2023 [visited on 2024-04-28]. Available from: https://courses.fit.cvut.cz/BI-ML1/lectures/files/BI-ML1-04-cs-handout.pdf.

39. *BI-ML1.21 přednáška 9 handout 28. listopadu 2023* [online]. CTU, Faculty of Information Technology, 2023 [visited on 2024-04-29]. Available from: https://courses.fit.cvut.cz/BI-ML1/lectures/files/BI-ML1-09-cs-handout.pdf.

40. GERON, Aurelien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, Inc., 2019. ISBN 9781492032649.

41. *BI-ML2.21 přednáška 5 handout 29. března 2023* [online]. CTU, Faculty of Information Technology, 2023. Available also from: https://courses.fit.cvut.cz/BI-ML2/lectures/files/BI-ML2-05-cs-handout.pdf.

42. SHINDE, Pramila P.; SHAH, Seema. A Review of Machine Learning and Deep Learning Applications. In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–6. Available from DOI: 10.1109/ICCUBEA.2018.8697857.

43. ZHANG, Lei; WANG, Shuai; LIU, Bing. *Deep Learning for Sentiment Analysis : A Survey*. 2018. Available from arXiv: 1801.07883 [cs.CL].

44. *BI-ML2.21 přednáška 7 handout 13. dubna 2023* [online]. CTU, Faculty of Information Technology, 2023. Available also from: https://courses.fit.cvut.cz/BI-ML2/lectures/files/BI-ML2-07-cs-handout.pdf.

45. LIPTON, Zachary C.; BERKOWITZ, John; ELKAN, Charles. *A Critical Review of Recurrent Neural Networks for Sequence Learning*. 2015. Available from arXiv: 1506.00019 [cs.LG].

46. *BI-ML2.21 přednáška 8 handout 18. dubna 2023* [online]. CTU, Faculty of Information Technology, 2023. Available also from: https://courses.fit.cvut.cz/BI-ML2/lectures/files/BI-ML2-08-cs-handout.pdf.

47. *Data Mining, Lecture 6 Artificial Neural Networks* [https://ocw.mit.edu/courses/15-062-data-mining-spring-2003/f89f20fe95a7a1993b5b8c6da9eb2e51_NeuralNet2002.pdf]. MIT, 2023.

48. RUDER, Sebastian. *An overview of gradient descent optimization algorithms*. 2017. Available from arXiv: 1609.04747 [cs.LG].

49. SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014, vol. 15, no. 56, pp. 1929–1958. Available also from: http://jmlr.org/papers/v15/srivastava14a.html.

50. WERBOS, P.J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*. 1990, vol. 78, no. 10, pp. 1550–1560. Available from DOI: 10.1109/5.58337.

51. SARI, Winda Kurnia; RINI, Dian Palupi; MALIK, Reza Firsandaya. Text Classification Using Long Short-Term Memory. In: *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*. 2019, pp. 150–155. Available from DOI: 10.1109/ICECOS47637.2019.8984558.

52. WANG, Jin; WANG, Zhongyuan; ZHANG, Dawei; YAN, Jun. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In: *International Joint Conference on Artificial Intelligence* [https://api.semanticscholar.org/CorpusID:21347790]. 2017.

53. SONI, Sanskar; CHOUHAN, Satyendra Singh; RATHORE, Santosh Singh. *TextConvoNet:A Convolutional Neural Network based Architecture for Text Classification*. 2022. Available from arXiv: 2203.05173 [cs.CL].

54. HOSSIN, Mohammad; M.N, Sulaiman. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*. 2015, vol. 5, pp. 01–11. Available from DOI: 10.5121/ijdkp.2015.5201.

55. MUELLER, Andreas; FILLION-ROBIN, Jean-Christophe; BOIDOL, Raphael; TIAN, Font; NECHIFOR, Paul; RAMPIN, Remi; CORVELLEC, Marianne; MEDINA, Juan; DAI, Yuchao; OTHERS. *amueller/word_cloud: WordCloud 1.5.0*. Zenodo, 2018. Version 1.5.0. Available from DOI: 10.5281/zenodo.1322068.

56. D'ULIZIA, Arianna; CASCHERA, Maria Chiara; FERRI, Fernando; GRIFONI, Patrizia. Fake news detection: a survey of evaluation datasets. *PeerJ Computer Science*. 2021, vol. 7. Available from DOI: doi.org/10.7717/peerj-cs.518.

57. AHMED, Hadeer; TRAORE, Issa; SAAD, Sherif. Detecting opinion spams and fake news using text classification. *SECURITY AND PRIVACY*. 2018, vol. 1, no. 1, e9. Available from DOI: https://doi.org/10.1002/spy2.9.

58. ZHOU, Xinyi; MULAY, Apurva; FERRARA, Emilio; ZAFARANI, Reza. ReCOVery: A Multimodal Repository for COVID-19 News Credibility Research. In: Virtual Event, Ireland: Association for Computing Machinery, 2020. CIKM '20. ISBN 9781450368599. Available from DOI: 10.1145/3340531.3412880.

59. SHU, Kai; MAHUDESWARAN, Deepak; WANG, Suhang; LEE, Dongwon; LIU, Huan. *FakeNewsNet: A Data Repository with News Content, Social Context and Spatialtemporal Information for Studying Fake News on Social Media*. 2019. Available from arXiv: 1809.01286 [cs.SI].

60. TACCHINI, Eugenio; BALLARIN, Gabriele; VEDOVA, Marco L. Della; MORET, Stefano; ALFARO, Luca de. Some Like it Hoax: Automated Fake News Detection in Social Networks. *CoRR*. 2017, vol. abs/1704.07506. Available from arXiv: 1704.07506.

61. DIGGELMANN, Thomas; BOYD-GRABER, Jordan; BULIAN, Jannis; CIARAMITA, Massimiliano; LEIPPOLD, Markus. *CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims*. 2021. Available from arXiv: 2012.00614 [cs.CL].

62. *news-please: A Generic News Crawler and Extractor*. Zenodo, 2017. Available from DOI: 10.5281/zenodo.4120316.

63. EUVSDISINFO. EUvsDiSinfo: About. In: [online]. 2015. Available also from: https://euvsdisinfo.eu/about/.

64. BARBARO, Florian; SKUMANICH, Andy. Addressing Socially Destructive Disinformation on the Web with Advanced AI Tools: Russia as a Case Study. In: *Companion Proceedings of the ACM Web Conference 2023*. Austin, TX, USA: Association for Computing Machinery, 2023, pp. 204–207. WWW '23 Companion. ISBN 9781450394192. Available from DOI: 10.1145/3543873.3587348.

65. TECHNICAL UNIVERSITY OF DENMARK. *Abundance of Information Narrows Our Collective Attention Span* [https://www.sciencedaily.com/releases/2019/04/190415081959.htm]. 2019. Accessed: April 30, 2024.

66. ELFOVÉ, Čeští. Čeští elfové: O nás. In: [online]. 2022. Available also from: https://cesti-elfove.cz/uvodni-strana/.

67. KLUYVER, Thomas; RAGAN-KELLEY, Benjamin; PÉREZ, Fernando; GRANGER, Brian; BUSSONNIER, Matthias; FREDERIC, Jonathan; KELLEY, Kyle; HAMRICK, Jessica; GROUT, Jason; CORLAY, Sylvain; IVANOV, Paul; AVILA, Damián; ABDALLA, Safia; WILLING, Carol. Jupyter Notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (eds.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas* [online]. IOS Press, 2016, pp. 87–90.

68. BIRD, Steven; KLEIN, Ewan; LOPER, Edward. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

69. BARBARESI, Adrien. *Simplemma*. Zenodo, 2023. Version v0.9.1. Available from DOI: 10.5281/zenodo.7555188.

70. GOMES, Luís. *Czech_stemmer* [online]. 2010. [visited on 2024-05-06]. Available from: https://research.variancia.com/czech%5C_stemmer/.

71. MCKINNEY, Wes et al. Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference* [online]. Austin, TX, 2010, vol. 445, pp. 51–56.

72. HARRIS, Charles R.; MILLMAN, K. Jarrod. Array programming with NumPy. *Nature* [online]. 2020, vol. 585, no. 7825, pp. 357–362 [visited on 2024-05-06]. Available from DOI: 10.1038/s41586-020-2649-2.

73. PEDREGOSA, Fabian; VAROQUAUX, Gaël; GRAMFORT, Alexandre; MICHEL, Vincent; THIRION, Bertrand; GRISEL, Olivier; BLONDEL, Mathieu; PRETTENHOFER, Peter; WEISS, Ron; DUBOURG, Vincent, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*. 2011, vol. 12, no. Oct, pp. 2825–2830.

74. MARTÍN ABADI; ASHISH AGARWAL; PAUL BARHAM; EUGENE BREVDO; ZHIFENG CHEN, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* [online]. 2015. Available also from: https://www.tensorflow.org/. Software available from tensorflow.org.

75. CHOLLET, Francois et al. *Keras*. GitHub, 2015. Available also from: https://github.com/fchollet/keras.

76. REHUREK, Radim; SOJKA, Petr. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* [online]. 2011, vol. 3, no. 2.

77. SVOBODA, Lukáš; BRYCHCÍN, Tomáš. New word analogy corpus for exploring embeddings of Czech words. In: *Computational Linguistics and Intelligent Text Processing*. Springer, 2016, pp. 103–114. Available from DOI: 10.1007/978-3-319-75477-2.

78. HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* [online]. 2007, vol. 9, no. 3, pp. 90–95. Available from DOI: 10.1109/MCSE.2007.55.

79. SALEH, Hager; ALHARBI, Abdullah; ALSAMHI, Saeed Hamood. OPCNN-FAKE: Optimized Convolutional Neural Network for Fake News Detection. *IEEE Access*. 2021, vol. 9, pp. 129471–129489. Available from DOI: 10.1109/ACCESS.2021.3112806.

Contents of attached medium

- `README.md` ..... a brief description of the content of the medium
- `src` .....
  - `datasets` ..... the datasets used for the experiments
- `text` .....
  - `thesis.pdf` ..... text of the thesis in PDF format
  - `thesis.zip` ..... source code of the Bachelor's thesis in LaTeX