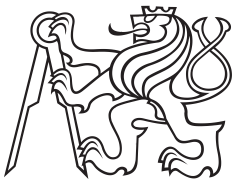
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Detecting Fake News Using NLP Methods

Denis Řeháček

Supervisors: Ing. Jan Drchal, Ph.D.
Field of study: Artificial Intelligence
January 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Řeháček Denis** Personal ID number: **434875**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Branch of study: **Artificial Intelligence**

II. Master's thesis details

Master's thesis title in English:

Detecting Fake News Using NLP Methods

Master's thesis title in Czech:

Detekce fake news metodami zpracování přirozeného jazyka

Guidelines:

The objectives of the thesis are:
1) Study state-of-the-art NLP methods. Focus on text classification and more specifically on algorithms based on Deep Neural Networks (DNNs).
2) Select an existing Fake News dataset.
3) Design, implement, and compare multiple types of DNN classifiers. Experiment with different architectures and optimize parameters.
4) Evaluate the models using the selected dataset. Discuss whether fake news is detectable based solely on the specifics language used.
5) Collect a similar dataset for the Czech

Bibliography / sources:

[1] Hanselowski, Andreas, et al. "A retrospective analysis of the fake news challenge stance detection task." arXiv preprint arXiv:1806.05180 (2018).
[2] Singhania, Sneha, Nigel Fernandez, and Shrisha Rao. "3han: A deep neural network for fake news detection." International Conference on Neural Information Processing. Springer, Cham, 2017.
[3] Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271 (2018).
[4] Davis, Richard, and Chris Proctor. "Fake news, real consequences: Recruiting neural networks for the fight against fake news." (2017).
[5] Kaggle Fake News dataset: https://www.kaggle.com/c/fake-news (2019).
[6] Infobanka ČTK: https://ib.ctk.cz/ (2019).

Name and workplace of master's thesis supervisor:

Ing. Jan Drchal, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **12.08.2019** Deadline for master's thesis submission: **07.01.2020**

Assignment valid until: **19.02.2021**

Ing. Jan Drchal, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

We thank the CTU in Prague for being a very good *alma mater*. I would also like to express my gratitude to my supervisor for giving me constructive comments and warm encouragement. Last but not least, my deepest appreciation goes to my family.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, January 7, 2020

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 7. ledna 2020

Abstract

This thesis introduces the problem of disinformation in an information-rich world. Fake News detection was addressed as a text classification problem. More than a hundred experiments were done to find an appropriate combination of pre-processing and efficient Neural Network architecture, relieving some specifics and limitations of the Fake News detection problem compared to other text classification tasks. An existing Fake News dataset was used as well as several combinations of a self-obtained data. The work is unique in processing news articles in numerous European languages, covering the same topics in both categories - reliable and disinformation news. The best accuracy was achieved by a convolutional based Neural Network, with up to 99.9% of correct prediction on the existing dataset, and over 98% in most experiments on the smaller self-obtained data, outperforming Self-attention mechanism. Better results were achieved when using the original texts instead of human-written summaries (even though the second option was trained on a larger dataset). Considering the datasets properties (same topics in both classes), the results suggest, there are probably language patterns distinctive for each of the two categories that were lost in the human-written summaries.

Keywords: NLP, Natural Language Processing, AI, Artificial Intelligence, Fake News, Disinformation, Multilingual Text Classification

Supervisors: Ing. Jan Drchal, Ph.D.

Abstrakt

Tato práce představuje problematiku dezinformací ve světě bohatém na informace. Detekce Fake News (falešných zpráv) byla řešena jako text classification problem. Bylo provedeno více než sto experimentů s cílem nalézt vhodnou kombinaci zpracování přirozeného jazyka (NLP) a efektivní architektury Neuronové sítě. Specifika a limity tohoto přístupu byla srovnána s jinými úlohami klasifikace textů. Byl použit existující dataset falešných zpráv i několik kombinací dat získaných konkrétně pro tuto práci. Tento projekt jedinečný ve zpracování článků v mnoha evropských jazycích, pokrývajících stejná témata v obou kategoriích - spolehlivé a dezinformační zprávy. Nejlepší přesnosti bylo dosaženo pomocí konvoluční neuronové sítě a to s až 99,9% správné predikce na existujícím souboru dat a více než 98% ve většině experimentů na menších samo-získaných datech, což předčilo Self-attention mechanismus. Lepších výsledků bylo dosaženo při použití původních textů namísto jejich lidmi psanými shrnutími (a to i přes to, že druhá možnost byla otestována na větším souboru dat). Vzhledem k vlastnostem datových sad (stejná témata v obou třídách) se lze předpokládat, že existují jazykové vzory specifické pro každou z kategorií, které byly ve shrnutích ztraceny.

Klíčová slova: NLP, zpracování přirozeného jazyka, AI, umělá inteligence, Fake News, dezinformace

Překlad názvu: Detekce fake news metodami zpracování přirozeného jazyka

Contents

Figures

Tables

Chapter 1

Introduction

The New York Times defines Fake News as "A made-up story with an intention to deceive ".[Tim16]

The term "Fake News" began to be frequently used after the US 2016 election, see Figure 1.1. Two separate investigations by the Guardian and BuzzFeed identified at least 140 websites producing fake news aiming US citizens, all of them ran from a small town in Macedonia. However, the date of the phenomenon is much older. The study of the fake news on social media aspect in the 2016 election [AG17] mentions the **Great Moon Hoax** - a series of six articles about a developed civilization on the moon, beginning on August 25, 1835. Regardless, we can find examples of what would be called "Fake News" today even in ancient history; a famous one dates to 13th century BC when Egyptian pharaoh Ramesses II, also known as Ramesses the Great, spread his propaganda about a fabulous victory in the Battle of Kadesh. Although, the Egyptian–Hittite peace treaty proves it was a stalemate.

In any case, in this work, I will focus on the modern aspect of the phenomenon. In the digital age, spreading of disinformation became cheaper and easier. Before, only media with sufficient resources were able to reach a large audience. Today, anyone can be a producer for the masses. [fEPSoE18]

We also need to consider another related phenomenon - **Content bubbles**. It can occur when a user is interacting with news sources, powered by personalized algorithms. An algorithm capable of learning the user's preferences to serve content that the user is interested in and agrees with. It helps the websites to keep their users and earn more from advertisements, etc.

Seeing the personalized content combining with behavioural biases, for example, "confirmation bias" (explaining that humans tend to like what they already agree with), can close the user in these so-called content bubbles[fEPSoE18]. The effect might be higher combing with the natural behaviour of socializing with people having similar opinions.

Last but not least, 2019 is likely to be the year when the internet will surpass TV in average daily consumption of media worldwide (see 1.2).

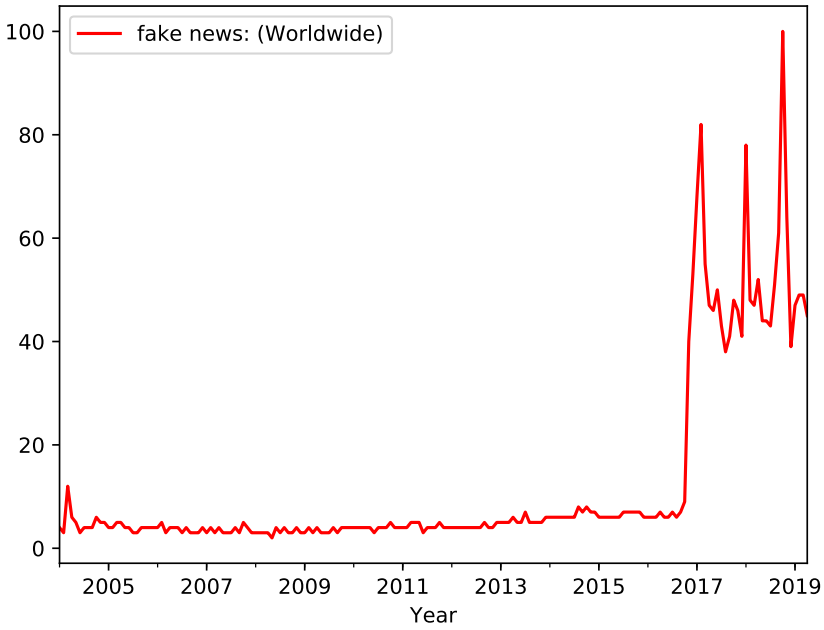


Figure 1.1: 'Fake News' in Google Trends

1.1 Facebook–Cambridge Analytica data scandal

Cambridge Analytica is a data analyzing company working for both Trump and Brexit campaign. They harvested the personal data of millions of people's Facebook profiles. Facebook later admitted it might be up to 87 million people whose information had been "improperly shared" with Cambridge Analytica.[New18]

Whistleblower and former employee of the company, Christopher Wylie, claims they have created psychological profiles of 230 million Americans by their so-called psychographic modelling techniques. The company focused on election disruption techniques, including rumour, disinformation, and fake news. All of them personalized by the psychological profiles databases.[Cad18]

Around 320 000 users permitted them to access their profiles in a Facebook app they created. In that time, the permission also allowed them to see every information that the particular user could see, not only content of the user itself but also everything that his friends published. No only what they shared to "public" but also what they shared with "friends". [Cad18]

Moreover, by default privacy setting, all Facebook likes were public. Meaning anybody could obtain a massive amount of data efficiently, not to mention these given with permission through third-party applications. Therefore, we can only guess how who can use this data for psychographic modelling and microtargeting with fake news.

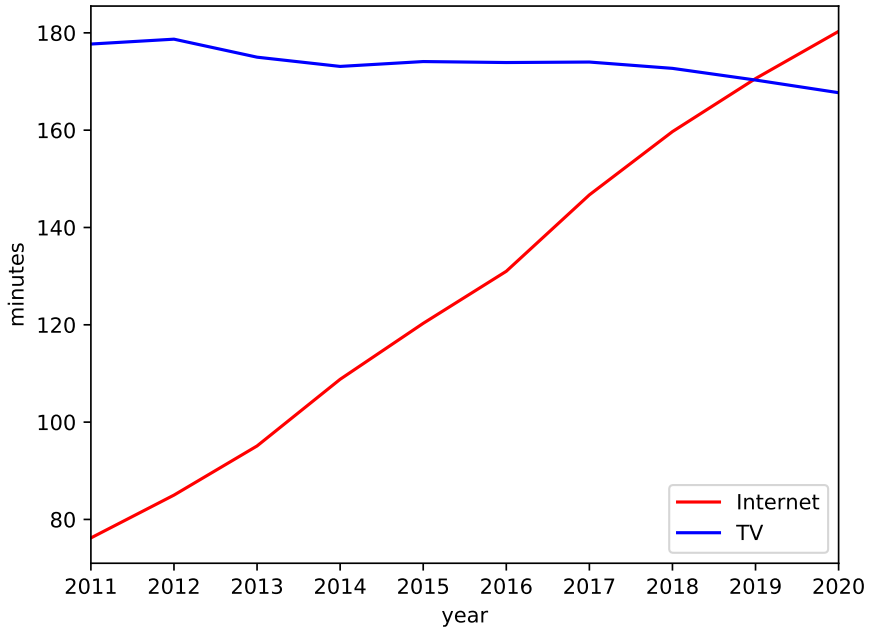


Figure 1.2: Average daily media consumption worldwide [fEPSoE17]

Cambridge Analytica in the Czech Republic

Other proofs about Cambridge Analytica work were brought by British investigative journalist who posed as a wealthy client who hoped to get candidates elected in Sri Lanka. He made videos using a hidden camera during his meetings with Alexander Nix, in that time, the company's CEO, revealing many practices they were using to rig elections. Nix said they had worked in more than two hundred elections worldwide, including Nigeria, Kenya, India, Argentina, and the Czech Republic. In another meeting he captured, company executive Mark Turnbull said: "We've just used a different organization to run a very, very successful project in an Eastern European country where... no-one even knew they were there." [cit, usc18]

Besides that, Wylie claims Cambridge Analytica offered their services to Russian oil company Lukoil. Although he has no proof of Lukoil buying their services, he has documents showing Cambridge Analytica briefed them on Facebook, microtargeting and election disruption. The connection to Czech politics goes through Martin Nejedly, Czech president's economic adviser who was paid by Lukoil.[Cad18]

1.2 East StratCom Task Force

East StratCom Task Force is a diplomatic service of the European Union. Besides other agenda, they have a team addressing pro-Kremlin ongoing

disinformation campaigns, established in March 2015. They published an analysis called **The legal framework to address "fake news": possible policy actions at the EU level** [fEPSoE18], which was one of the main inspirations for this research. The paper describes the spreading of disinformation and summarizes key finding. Including decreasing trust in the internet, low ability of its user to single out reliable news worthy of their attention, the problem of content bubbles. Additionally, they discovered that many of the disinformation news is ignored, there is still a part of them that is spread quickly, affect public opinion, create a "noise" by giving many contradicting versions of a single event, and in the last effect, it causes confusion and trouble for the users.

They also expressed warning of addressing Fake News with a strict solution that would amount to censorship.

In the long-term, they predict the online news will be permeated by artificial intelligence on both sides, the one creating and sharing harmful content and the other trying to keep the internet journalism safe and reliable. Most importantly, they claimed that an AI-powered literacy is the most effective response to disinformation.

Last but not least, the East StratCom Task Force gently provided their disinformation dataset for this research.

1.3 Definitions

By the New York Times definition, **Fake News** is a story with the intention to deceive. The EU framework [fEPSoE18] works with term **disinformation** for news with clear intention to manipulate public opinion. A similar term - **misinformation** - stands for information that is not based on the truth but was not expressed with no intention to manipulate public opinion. This thesis does not investigate the intention of the news articles.

1.4 Problem statement

This work addresses the Fake News detection task as a text classification problem with the aim to discover whether fake news is detectable based solely on the specifics of the language used. It focuses on algorithms based on Deep Neural Networks.

Chapter 2

Fake News detection

The Fake News detection problem can be divided into several sub-problems. I will discuss the possible automatization of some of them and attempts that were already made. Considering the success of Neural Networks in similar tasks, it might be possible to achieve satisfying accuracy in many of the sub-problems, or in all of them. The final score of an article could be then established as a combination of these sub-problems. This approach also minimises the risk of false-positive classification.

There are methods based on Natural Language Processing (NLP) such as Stance Detection - estimating if two pieces of text (in this case, the headline and article body) are related or not. Text classification can provide insight into the article or the comment section.

By image processing, one can uncover photoshopped images or deep fake videos (see 2.2.5). Having this information about article reliability, we can further track the spreading of the article across social media and try to find any differences between spreading fake and reliable news.

2.1 Linguistic characteristics and text classification

Paper [GBEF+18] examines if there are any differences in the language of fake news and satire. The work did not aim to do a deep linguistic analysis. Primarily they tried to find any linguistic characteristic, such as word usage patterns, that would be specific for one of the two types of text. Therefore, they did not use any advanced machine learning models and used bags of words encoding. They achieved 79.1% accuracy with Naive Bayes Multinomial algorithm only and 0.882 using PRC (Precision-Recall Plot) Area on the Fake or Satire task and based on the preliminary results suggested that it seems to be possible to determinate a theme (such as conspiracy theory) of an article based on its word vector only. On the other hand, they used a dataset of only 283 fake news articles and 203 satirical stories, which makes it hard to tell if the model generalises enough or if it only learns a few specific words for each category. For later research, it would be interesting to obtain any additional information about based on what the model predicts.

Text classification can provide fast, reliable information about the content of an article. By visualisation of features that led to the given output, we can prove the correctness of the decision, or at least give more insight into the problem.

2.2 Other possible methods

This work addresses the Fake News detection task as a text classification problem with the aim to discover whether fake news is detectable based solely on the specifics of the language used. However, there are more approaches of the detection. Those that were examined already are briefly summarised in this section. An efficient real-word Fake News detection system should be then a combination of, if possible, all of them.

2.2.1 Stance Detection

Stance Detection is a task estimating the relative perspective (or stance) of two pieces of text relative to a topic, claim, or issue. It was considered useful in Fake News Challenge competition [HSS+18] that aimed to use AI techniques to detect the credibility of an article. They also suggested breaking the checking process into several stages. Stance Detection was the goal of the first stage.

More precisely, they introduced the following classes for stance detection:

- Agrees: The body text agrees with the headline
- Disagrees: The body text disagrees with the headline
- Discusses: The body text discuss the same topic as the headline, but does not take a position
- Unrelated: The body text discusses a different topic than the headline

Moreover, they also introduced the evaluation policy to be a two-level scoring system. The first level giving 25% of the score weighting is to classify if headline and body text are related or unrelated. The second level is to classify related pairs as agrees, disagrees, or discusses. Besides the classification and scoring system, they prepared a baseline solution for participants to compare their models.

Stanford team compared four different architectures in this competition.

- Concatenated multilayer perceptron (Concat MLP)
- Bag-of-words multilayer perceptron (BoW MLP)
- Dual GRU
- Bi-directional, concatenated, stacked LSTM (Bi-dir MSTL)

[RD17]

They ran over 100 experiments to improve their models, but only BoW MLP achieve a better score than the baseline solution. This model got 0.97 accuracy on the first level of scoring (related or unrelated headline), 0.93 on the second level (classification of the related pairs) and 0.89 on overall NFC-1 score - getting 10% improvement compared to baseline solution. Surprisingly, none of their RNN architecture was able to beat the baseline. They gave three possible explanation. First, since they were able to achieve high accuracy on training data but never on validation data, they assume, RNN would need a much larger dataset to generalise.[RD17] Second, even advanced RNN models - Bi-dir MSTL and Dual GRU - are struggling with forgetting long-term information that may be crucial in an article classification problem. [SHS01, RD17] Third, they plan to enhance their RNN models by attention layers. [RD17]

Beside RNN with attention, they also announce more development in NLP branch. Mainly they plan to improve the dependency tree by weighting word token by their inverse depth in the tree. Hoping it helps to distinguish words that are central to the meaning from these that are not. However, they do not provide any proof for this hypothesis.[RD17]

2.2.2 Collective user intelligence

In [RSL17], authors suggest classification based on three Fake News characteristics:

- The text of the article
- The user response
- The source users promoting it

Despite these patterns are only observable after the Fake News spread and exposed to a considerable number of users, there are papers suggesting to consider them. Especially in combination with a User response generator (URG)- a model that tries to predict users responses even before they are available and use them for early detection. [QGSL18] Furthermore, it seems to be important to study URG since it has been shown that automatically generated comments under news articles affect massively users acceptance and trust.[Lee12]

On the other hand, this model relies deeply on collective users intelligence. In the case of closed groups (i.e. groups in which users can not view the group's content until they become a member), this assumption can be untrustworthy. Admins of such a group could filter its members and so also filter the discussed topics.

2.2.3 Fake image detection

Similarly to the stance problem of the headline and article's body, it might be beneficial to check images attached to the article if they are related to

the text. A wrongly chosen picture can manipulate in the same way as a misleading headline. Made up stories can also use a picture from a completely different occasion and use it as "proof" or photoshopped it in such a way it suits their story and can be unable to distinguish from an authentic picture. This problem is even mentioned in **The legal framework to address "fake news": possible policy actions at the EU level** published by the European Parliament.[fEPSoE18]

There are several models that achieve good accuracy in object detection problem, mostly based on Convolutional Neural Networks (CNN), such as YOLO, R-CNN, Fast R-CNN, Mask RCNN and other [ZZXW18]. Their variations can be used to detect if the image is relevant to the article itself. The popular approach is to use with pre-trained models in order to cut training time and cost.

Moreover, there are other models capable of detection if a picture was edited or not. Beside these NN tasks, we can also check online if the same image was used previously, check the source claimed in the article and compare its content with the original post. The article could be considered as unreliable if it turns out that the picture is actually taken in a different place and circumstances or in a different time then claimed.

2.2.4 Source checking

A proper news article should always contain its author and sources. Some of the Fake News media have already learned this and create a chain of many articles published on different websites to look like it is covered adequately by sources, but the original source is actually missing. Therefore, automatic checking of sources could be an exciting feature to include in the evaluation. This also applies to sources of images.

2.2.5 Deep fake

Deep fake is another element mentioned by the European framework to address Fake News [fEPSoE18] said to become increasingly a problem. The term Deep fake stands for a technique of creating face-snap pictures or videos that can look like real. As face snap-in pictures date back to the mid-19th century. In the last years, it became easier to do the same in videos. With apps like FakeApp and Lyrebird, anyone can release a Deep fake video in a believable quality. In any case, the result is not perfect and can be detected with up to 97.1% accuracy with a Neural Network combining a CNN layer for feature extraction and LSTM for sequence processing.[GD18]

2.2.6 Neural Fake News

Neural Fake News is a relatively new term used for automatically generated Fake News, typically using Neural Networks.

A study [ZHR+19] presents a model named **Grover** for controllable news article generation. It can generate an entire article based on a small piece

of text or headline only. Moreover, if a news website domain is given in the input, it generates the text in the style of that newspaper. The Neural Fake News generated by this model was rated by humans as more trustfully than human-written news.

A demonstration of a fake article generated by the model based on the headline "AI in the Fight Against Fake News".

January 14, 2019 - Denis Rehacek

The Artificial Intelligence (AI) has become a major investment trend in 2018, with major tech companies pouring millions into tech startups focused on combatting fake news and trolling. As the Fake News Season begins, the startups continue to create AI that improves search and filtering of fake stories and computational suppression of fake news, where automated tools automatically censor and evaluate content.

...

The demonstration was generated on the website[1] of the project. It is only the first paragraph of the generated text. The full article is in the appendix A. The classification accuracy on Czech datasets

The paper also claims that the best defence against the model is the model itself, scoring 92% accuracy in the Neural Fake News detection. They trained the model on 120 gigabytes of real news articles obtained from Common Crawl. The Grover architecture is based on the Transformer model [AV17]. Three versions of it were tested, the first with 12 layers and 117 million parameters, the second with 24 layers and 345 million parameters and the largest with 48 layers and 1.5 billion parameters.

The model was trained with randomly sampled sequences of length 1024 in 800 thousand iterations on hardware with 256 TPU v3 cores. Even with powerful hardware, the training time was two weeks.

The concept of Neural Fake News could be helpful for training datasets created for article classification problem since it is hard to find an adequately large dataset of labelled reliable and unreliable news articles.

[1] https://grover.allenai.org/

Chapter 3

Datasets

The Fake News Challenge - Stage 1 dataset[1] only labels the relationship between article headline and body, not considering whether it is fake or credible. Also, the four classes are not equally distributed having 0.73% in the unrelated group, 0.18% labelled as discuss and for the last two categories; agree, disagree only 0.07% and 0.02% respectively.

Fortunately, there are more datasets available.

3.1 Kaggle

This dataset[2] consists of 20.8k entries in training and 5200 entries in the test group. It is available online at Kaggle as a part of a public classification competition that took place in 2018. It contains id, title, author, text, and label:

- 1 - unreliable
- 0 - reliable,

with 50:50 distribution of each.

3.2 Unused datasets

This section contains a summary of other Fake News related datasets available online which were considered for this thesis. None of them was used but could be interesting for further research.

3.2.1 Kaggle

Second dataset[3] available on Kaggle was collected using BS Detector Chrome Extention, which can be considered as a disadvantage since it was not human-label or at least checked after.

[1] https://github.com/FakeNewsChallenge/fnc-1
[2] https://www.kaggle.com/c/fake-news
[3] https://www.kaggle.com/mrisdal/fake-news

3.2.2 BuzzFeed

Dataset[4] of articles published on Facebook during just one week close to US presidential election 2016. In addition to the news content, it also contains social context information.

Another of BuzzFeed datasets[5] contains 2284 human-label Facebook posts enriched by category (left, right, mainstream), page and social content such as the number of comments, reactions, and shares.

The social context information in these datasets can be helpful for social behaviour modelling and similar applications.

3.2.3 LIAR

This dataset was obtained from the fact-checking site PolitiFact. This site collects public figures statements and labels them with one of the following class: pants-fire, false, barely-true, half-true, mostly true, and true. It also includes meta-data about speaker's party affiliations, current job, home state, credit history. In [Wan17], they reached significantly better results while taking into consideration meta-data like this. However, it contains only specific statements that are significantly shorter compared to a news article. This fact makes the dataset inappropriate because the length of a text sequence is one of the main aspects to deal with in the text classification tasks and is discussed in the following chapter.

3.3 Self-obtained dataset

The previous datasets are oriented on US politics news only. The topics included can vary compare to European Fake News and so any Neural Network model trained on this data cannot be accurate enough. Therefore, for this work, another dataset was collected. **EU vs Disinformation campaign** [6] ran by the **European External Action Service East Stratcom Task Force** gently provided their database of 5609 disinformation cases in 18 different languages focused on a European audience. The articles were published between 1.01.2016 and 31.12.2018., and are described by 558 distinct keywords. Each entry consists of:

- Issue
- Date
- Language / Targeted Audience
- Summary of the Disinformation

[4] https://github.com/KaiDMML/FakeNewsNet/tree/master/Data/BuzzFeed

[5] https://github.com/BuzzFeedNews/2016-10-facebook-fact-check/blob/master/data/facebook-fact-check.csv

[6] https://euvsdisinfo.eu/about/

- Link to the Disinformation
- Disinforming Outlet(s)
- Disproof
- Keywords
- Countries
- URL

Some of the entries contain more links 'Link to the disinformation' for one disinformation case, some links lead to 404 HTTP error (not found), other lead to a video material only but there is always a summary in English in 'Summary of the Disinformation' column.

Examples of reliable news were crawled from public service broadcasters (such as British Broadcasting Corporation - BBC) filtered by the published date and keywords obtained from the EU vs Disinformation dataset to ensure similar topics like these in the disinformation group.

3.3.1 Crawling news

News articles were parsed using news-please crawler [HMBG17]. It provides a unique combination of several state-of-the-art libraries and tools; hence, it can parse news articles and additional information from almost any website. The following information was obtained from each article:

- title
- text
- description
- language
- URL
- publish date
- modify date
- filename

First, the original disinformation articles from EU vs Disinformation campaign data were crawled. Links leading to video only material were omitted as well as links leading to social networks because the posts could be edited. Moreover, some of the social network links are not permanent, and a request to them can be redirected several times, making it hard to parse the text correctly. In total, 3313 original documents were found, 247 articles were duplicates (same article body text) and were also omitted, therefore 3066 articles were used.

These texts are in a variety of languages. One of the limitations of the news-please library is the fact that it detected the language in 2790 cases from 5189 only. The most occurred language was Russian, followed by English. The Czech language was not detected in a single time. The language of the disinformation is also written in the "EU vs Disinformation campaign" data, but in some cases, there are more of them for one disinformation, and it is not clear which of the links lead to which language. However, for the Czech language, the dataset contains 716 entries with 746 valid links. Only 491 leads to news articles, and another 11 of them leads to duplicated text. Therefore, only 480 original disinformation texts were found.

Samples of reliable news were crawled from public service broadcasters; British BBC, one of the many German public service broadcaster - German DW (Deutsche Welle) and Czech CTK. Both BBC and DW provide their services in a number of languages. Besides of English, Russian articles were downloaded as well because it is the most common language in the *EU vs Disinformation campaign's* dataset.

A historical version of the websites was visited using WebArchive [7]. The first snapshot of every day in the interval between 01. 01. 2016 and 31. 12. 2018 was used to get the articles published in the main section of the homepage of every broadcaster used (and its Russian language version). Some news articles remained on the main section till the following day, these duplicates were removed. Note that the interval corresponds to the first and the last publish_date int the *EU vs Disinformation campaign's* dataset. It also provides keywords of each disinformation case. Reliable news that did not match any of these keywords were not used either.

English reliable news corpus

In total, 11 981 unique articles were obtained from the English version of BBC and 21 052 from DW. Due to a large number of articles on these websites, only those that were in the popular section of BBC[8] and in the main list of articles on the English version of DW[9] were downloaded. Most of the articles contained at least one of the keyword (32 666 of 33 033), hence the final dataset of reliable news in English consist of 32 666.

Russian reliable news corpus

In Russian, 5 510 unique articles were downloaded from BBC (from the main grid of its Russian website[10]) and 1 382 from the Russian version of DW[11], having totally 6 892 reliable news articles in Russian and only 2 548 of them matched at least one of the keywords.

[7] http://web.archive.org/
[8] https://www.bbc.com/news/popular/read
[9] https://www.dw.com/en/top-stories/s-9097
[10] https://www.bbc.com/russian
[11] https://www.dw.com/ru/s-9119

	web	visits (November 2018)	visits (May 2019)
1.	ParlamentniListy.cz	8.51M	8.47M
2.	cz.sputniknews.com	2.50M	2.73M
3.	AC24.cz	1.04M	1.57M
4.	Aeronet.cz	923.76K	943.73K
5.	ProtiProud.cz	610.90K	604.25K
6.	Zvedavec.org	452.72K	475.83K
7.	Vinegret.cz	282.42K	265.41K
8.	NWOO.org	254.85K	262.69K
9.	Rukojmi.cz	202.48K	200.27K
10.	SkrytaPravda.cz	141.35K	178.16K

Table 3.1: The most visited disinformation websites in the Czech Republic

3.3.2 Czech news

The EU vs Disinformation dataset contains 716 disinformation cases in the Czech language with 746 valid links to the original sources. After dropping social network posts, video materials and duplicates, only 461 articles remained in this category.

Only 716 cases (461 news articles) might not be enough to train a classifier properly. Therefore, it was extended by crawled data from most visited Czech (in the name of content, not owners) disinformation websites.

Top ten Czech disinformation websites by visits can be seen in table 3.1, data were obtained from disinformation analyses[12] by newspaper 'Hospodarske noviny' and the number of visits from Similarweb[13]. Similar results were also find in a study of Information laundering: fake news websites in czech context [Jan18].

Examples of reliable news in the Czech language are obtained from the Czech News Agency (CTK)[14] [cit19], a public service broadcaster. Providing objective and versatile information for the free creation of opinions is control by CTK council: seven-member body constituted by the law. Its members are elected by the Chamber of Deputies[15].

Czech News Agency (CTK) controversy

One of the current members of the CTK council - Petr Zantovsky has a tight connection to Czech disinformation scene. According to [Jan18], he founded so-called Independent Media Association (Asociace nezávislých médií) together with Ondrej Gersl, Jan Koral and Stanislav Novotný. Ondrej Gersl is the founder of disinformation web AC24[16] (see table 3.1 containing list of

[12] https://domaci.ihned.cz/c1-66477660
[13] https://similarweb.com
[14] https://www.ctk.eu/
[15] https://www.ctk.eu/about_ctk/
[16] https://ac24.cz/

the most visited disinformation websites in the Czech Republic) and LajkIt[17]. Jan Koral is the founder of another disinformation website NWOO[18].

3.3.3 Dataset-A

The first dataset considers the texts in 'Disproof' column as reliable samples and 'Summary of the Disinformation' as unreliable. Only rows having more than 32 characters in both of these columns were used, shorter cells usually contained meaningless information such as "No proof for this." or "No evidence given."

The total length of this dataset is then 10 570 with 50:50 ratio and everything in it is in English.

This dataset is probably harder to classify because it does not contain the disinformation text itself, only the summary of it. Moreover, the two columns ('Disproof' and 'Summary of the Disinformation') describe the very same issue and is shuffled randomly, therefore a deep text understanding is needed.

3.3.4 Dataset-EN

There were 519 disinformation documents founded. To maintain 50:50 ratio, another 519 articles were picked randomly from the English reliable news corpus.

3.3.5 Dataset-RU

In Russian, there were 1 622 disinformation documents founded. In the same way as before, another 1 622 articles were picked randomly from the Russian reliable news corpus.

3.3.6 Dataset-M1 and Dataset-M2

Two multilingual dataset were created from 3 066 disinformation texts. The first one (Dataset-M1) was completed with 3 066 random articles from the English reliable news corpus, the second (Dataset-M2) with both English and Russian reliable news corpus in 1/3 ratio.

3.3.7 Dataset-CZ-X

Reliable articles in the Czech language were obtained from CTK within TAČR TL02000288 project. There were 461 unreliable articles in the Czech language in EU vs Disinformation dataset. Only 461 CTK articles were used, filtered by EU vs Disinformation keywords and same published date period, to keep the dataset balanced. **Dataset-CZ-1** of size 922 contains these two groups. **Dataset-CZ-2** is enriched with a discuss category - articles crawled from disinformation websites ParlamentniListy.cz and AC24.cz (see table 3.1 of the

[17] https://www.lajkit.cz/
[18] http://www.nwoo.org/

dataset	entries	labels			
		reliable	unreliable	discuss	satire
Dataset-A	10 570	*	*		
Dataset-EN	1 038	*	*		
Dataset-RU	3 244	*	*		
Dataset-M1/M2	6 132	*	*		
Dataset-CZ-1	922	*	*		
Dataset-CZ-2	1368	*	*	*	
Dataset-CZ-3	1104	*	*		*
Dataset-CZ-23	1565	*	*	*	*

Table 3.2: Self-obtained datasets size and labels comparison

most visited disinformation websites in the Czech Republic). Disinformation on these websites occurred many times in the EU vs disinfo dataset. There were labelled as disinformation websites by many studies, for example, **??**, and by manipulatori.cz (an organisation that is recommended by Ministry of the Interior as an institute of media literacy[19]).

On the other hand, not all articles published on these servers can be automatically labelled label as disinformation. Some of them try to be proper news. Many of them are opinions or columns. Even politics and public figures across the political spectrum use Parlamentni Listy as a platform to share their opinions. Therefore, another category - discuss - was established for these articles.

The last category is satire. Articles for this category were crawled from az247.cz [20] that imitates the style of disinformation and reflect current topics and trends. However, there were only 182 articles available, making this category the only one that is not properly balanced. **Dataset-CZ-3** consists of reliable, unreliable and satire. Finally, **Dataset-CZ-23** is a combination of all previous - reliable, unreliable, discuss and satire.

3.3.8 Datasets summary

The sizes of the final datasets are summaries in table 3.2, all of them are perfectly balanced except the two datasets containing satire due to a lack of appropriate articles. All articles are from the same time period (01. 01. 2016 - 12. 12. 2018) and contain similar topics (filtered by keywords).

It is hard to find a human-labelled dataset of real word Fake News article provided by trustful source. In this project, the classifiers is first tested on the Kaggle dataset. The most successful models and approaches is picked for experiments on the smaller self-obtained datasets.

[19] https://www.mvcr.cz/cthh/clanek/dezinformacni-kampane-dokumenty-a-odkazy-dokumenty-a-odkazy.aspx

[20] https://az247.cz/

Chapter 4

Machine Learning with Sequential Data

This chapter introduces Machine Learning and its techniques capable of sequential data classification. Furthermore, Natural Language Processing and Neural Networks are discussed.

Machine Learning stands for the ability of systems to automatically learn and improve base on previous observation and experiences instead of relying on specific instructions. There are three main types of learning - supervised learning, unsupervised learning and reinforcement.

Supervised learning learning uses labelled dataset, the system observes the input-label pair and tries to find a mapping function between the pair. With more input data, the function is getting more and more general and later is able to give the right prediction for inputs that have not been observed before.

Unsupervised learning system can learn even without labelled input. The so-called "AI bible" [RN09] gives a taxi agent as a good example, the agent can gradually learn a concept of "good traffic day" and "bad traffic day" without being explicitly told (by the input label) which day was "good" traffic and which day "bad".

Reinforcement learning is based on reinforcement rewards or punishment. On the taxi drive example, the agent can observe he did something good when he gets an unusually large tip and something bad if he does not get a tip at all. After more samples, it can determinate which actions led to large-tip journey and no-tip journey respectively.

The news classification problem is clearly an example of supervised learning. On the other hand, some of the pre-trained word embedding models discussed later were trained using unsupervised learning.

4.1 Sequential data

A simple classifier such a Support vector machines or similar is not capable of modelling sequences in time. There were attempts to extend these approaches by giving a number to each data frame describing its order in time. These models could be used for an easy application such as central call automation to route calls. However, any of these models did not show sufficient results in any advanced text processing task. [ZCL15]

4.2 Natural Language Processing

This section contains a brief introduction to text processing - representation of natural language inputs and outputs.

Machine Learning classifiers do not work with the text directly. The documents need to be transformed into a representation that is suitable for processing by machine learning classifiers. For better results, the following actions are usually performed:

- Tokenization - convert sentences to words;
- Remove stop words - Removing unnecessary, repeating words that have not any or low impact on the meaning such as "the", "is" etc. As well as punctuation, tags and so on;
- Stemming or Lemmatization - obtaining the root of words by removing unnecessary characters - usually suffix. This step depends strongly on the natural language characteristic.

The next step is encoding words into numerical values or vectors. There are more algorithms for that.

4.2.1 Bags-of-words

Bags-of-words is a simple representation, where the input text is transformed into a multiset of words that is called a bag. Multiset keeps information about multiplicity. It stores tuples of words and number of its occurrence in the sequence. Having a map storing which word corresponds to which index, it is used as a simple vector. The value is 0 for words of the dataset's vocabulary, which are not in the sequence. However, that also means that any information about grammar and word order is lost and for a more extensive dataset with rich vocabulary, most of the values of small sequences will be 0.

A document can be converted into the Bags of Words encoding using CountVectorizer from sklearn library. Its function **fit_transform(self, raw_documents[, y])** takes a list of document as parameter, returns a term-document matrix and learns its vocabulary which can be obtained by calling **get_feature_names(self)**.

Each line of the matrix stands for one document, and each element of the line represents the count of the word in the corresponding index of vocabulary.

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
```

```
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the',
'third', 'this']
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

In this example, the vocabulary consists of nine words. Therefore, the matrix has nine columns. Let us interpret the second line - [0 2 0 1 0 1 1 0 1]. The first element refers to the first index of vocabulary - the word 'and' it can be seen that there are zero occurrences of this word in the sentence 'This document is the second document.'. Next, there is number 2, showing the word 'document' was used twice. In the same way, we can interpret each element of the matrix.

Vectorizer options

We can also demonstrate the effect of applying stopwords. The vocabulary, in the tiny corpus example, has five words instead of nine and so the matrix has only five columns.

```
>>> from nltk.corpus import stopwords
>>> vectorizer = CountVectorizer(
                    stop_words=stopwords.words('english')
                               )
...
['document', 'first', 'one', 'second', 'third']
...
[[1 1 0 0 0]
 [2 0 0 1 0]
 [0 0 1 0 1]
 [1 1 0 0 0]]
```

The vectorizer can also be set to a **binary** mode, if so, all no zero values are set to 1. The vocabulary can be restricted to consider only the most occurring words by **max_features** parameter. With **max_df** and **min_df**, terms having document frequency strictly higher and lower respectively are ignored. On the tiny example above, min_df can be set to 3, meaning only words having occurrence higher than three will be considered. It would lead to having only one word; 'document,' which is definitely not representative enough. However, in a larger corpus, this can efficiently filter the vocabulary from words with insignificant impact.

4.2.2 TF-IDF Vectors

TF-IDF Vectors is more complex representation that do not stay on words count only. TF-IDF Vectors consists of two parts, TF stands for Term Frequency and IDF for Inverse Document Frequency, defined by the following formulas:

$$TF(t) = \frac{\text{number of term t occurrence in a document}}{\text{total number of terms}}$$

$$IDF(t) = ln(\frac{\text{total number of documents}}{\text{total number of documents with term t}})$$

Matrix representation in different levels

- Words level - represents TF-IDF scores of terms;
- N-gram level - represents TF-IDF scores of N-Grams;
- Character level - represents TF-IDF scores of character level n-grams;

N-gram uses a Markov model to approximate the probability of next term/character in the sequence.

4.2.3 Word Embeddings

Word vector embedding is word representation standing on a hypothesis that words used in the same context tend to have a similar meaning. It represents each word as a high dimensional dense vector and, concerning the hypothesis, it uses similar vectors for words in a similar context. There are several methods of obtaining such a representation, including Neural Networks, probabilistic models, dimensionality reduction, and more. It can be trained on the processing data or loaded from a pre-trained embedding matrix.

A possible disadvantage of embedding models is giving only one vector to a word that has many different meanings. The problem is address by an extension that gives more vector for a single multiple-meaning word — one for every of its specific meaning.

List of word vector embeddings models:

- Word2Vec [MSC+13]
- GloVe [PSM14a]
- FastText [BGJM16]
- ...

Before using an embedding, the input documents have to be transformed into a sequence of integers.

At first, a list of all words used in the corpus has to be obtained. Second, a dictionary connecting every word to its index in the dictionary is created.

The dictionary is known as **word_index**. For a case, there would be a demand for back transformation, it is beneficial to create another dictionary connecting indexes back to words, to do the transformation efficiently.

The **word_index** dictionary is then used as a look-up table. The input text documents are processed in its natural order replacing every word with its index from the look-up table. Each element of the resulting sequence is a numerical value referring to an index of the given the word in the vocabulary.

```
def to_s(tokenizer, preprocessor, index, text):
  words = tokenizer(preprocessor(text))
  indexes = [index[word] for word in words if word in index]
  return indexes

def get_s(vectorizer: CountVectorizer,corpus,):
  word_index = {word: idx for idx, word in
                  enumerate(vectorizer.get_feature_names())}
  tokenize = vectorizer.build_tokenizer()
  preprocess = vectorizer.build_preprocessor()
  sequences = [to_s(tokenize, preprocess,word_index, x)
                  for x in corpus]
  return sequences, word_index

>>> seq, word_index = get_s(vectorizer, corpus, None)
>>> print(seq)
    [[8, 3, 6, 2, 1],
     [8, 1, 3, 6, 5, 1],
     [0, 8, 3, 6, 7, 4],
     [3, 8, 6, 2, 1]]
>>> print(word_index)
    {'and': 0, 'document': 1, 'first': 2, 'is': 3, 'one': 4,
    'second': 5, 'the': 6, 'third': 7, 'this': 8}
```

The code above shows the transformation on the same corpus as was used for Bags of Words example before. First, each word from vocabulary is stored in word_index dictionary. Again, each line stands for one document from the corpus. The elements of the line, on the other hand, do not mean a count of the word on the same position in vocabulary (as in Bag of Words) but is the index of the word. On the example above, the first element of the first line is eight that stands for 'this' as can be seen in the word_index dictionary. In the same way, the original sentence 'this is the first document' can be obtained.

If needed, another number without any meaning (not contained in the dictionary) can be added to the sequences (usually in the beginning) to make them all the same length.

The benefit of this encoding compared to Bag of Words is the fact that it maintains the order of the words and not only counts its occurrence.

Finally, everything is prepared to load the pre-trained embedding. It is

stored in a matrix in which every line corresponds to a word and every column to a dimension. Typically 300 dimensions are used to save computation cost although the pre-trained embeddings are (usually) in a higher dimension. The size of the matrix is too extreme, and for many applications, it is not necessary to load it fully.

Global Vectors for Word Representation - GloVe

GloVe is one of the most popular, proposed by Stanford in 2014 [PSM14a]. It comes with two main highlights; Nearest neighbours and Linear substructures. The first one ensures that the Euclidean distance in the high dimensional space between a pair of word vectors corresponds to their linguistic or semantic similarity. For example, in the pre-train GloVe embedding, the nearest neighbours for the word "frog" are:

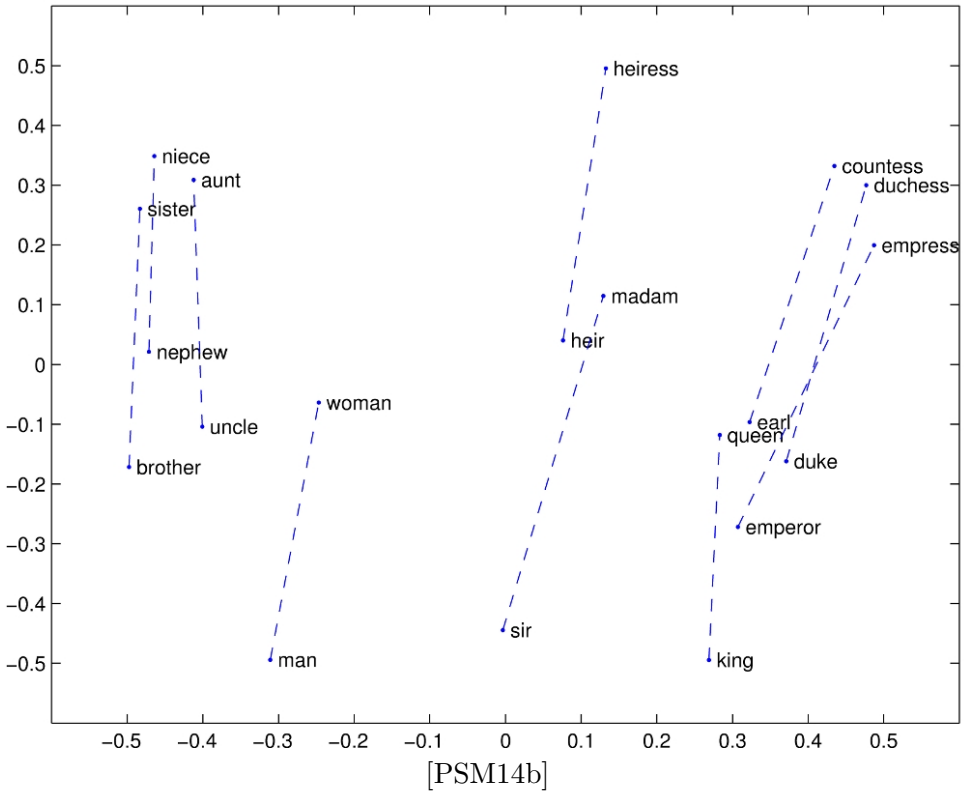
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

Note that the closest one, in this case, is the plural form, followed by frog species, genera, etc.

The second highlight, Linear substructures, is the product of the previous metric operation. The vector difference of a word pair is roughly equal differences of its synonyms and other terms with similar meaning. It is easy to understand to these substructures on an example of the word pair: "man" and "women" in the figure 4.1. Not only that the vector differences are similar to other word pairs, but it also groups gender-specific terms of family members to one place (sister - brother, niece - nephew, aunt - uncle) and terms describing royal families members (queen - king, duchess - duke, and more) to another. Another interesting pattern is the superlative comparative relation in figure 4.2. Thanks to these properties, much more comprehensive information about the words and its meanings are contained.

Pre-trained GloVe word vectors **glove.840B.300d** by the authors [PSM14b] are available online[1]. It was trained on the Common Crawl data and contains 840B tokens, 2.2M vocab, cased and 300 dimensional vector.

[1] http://nlp.stanford.edu/data/glove.840B.300d.zip



[PSM14b]

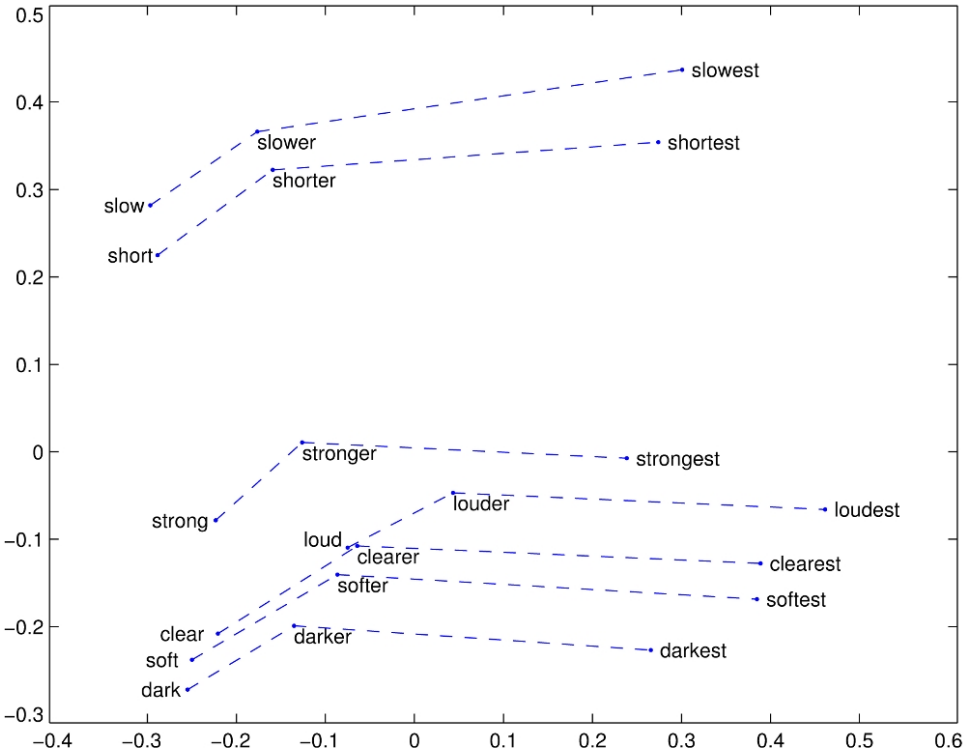
Figure 4.1: Linear substructures: man and woman

Count vs predict embeddings

Paper [BDK14] compares several embeddings models that they distinguished into two main categories: predictive and count-base. Count-based models can be represented as a matrix where each line is a word and columns stand for its contexts with several such an occurrence. In the next step, the second matrix is found in a way that maintains most of the original matrix information but is smaller in size. The predictive models, on the other hand, is represented as two-layer neural network predicting a word in its surrounding context (in case of Continuous Bag of Words - CBOW) or it predicts the context from the word (Skip-Gram). The second variation usually performs better on large datasets. The paper contradicts their hypothesis of predictive models being groundlessly preference without any evidence of their benefits and showing that predict-based models outperform count-base ones in every tested task but not by a large margin. The disadvantage of predictive models is harder training compared to count base model, considering similar results it might not be worth to use in some tasks.

FastText

FastText [MGB+18, BGJM16] is a popular predict-based model enriched by subword information. They proposed more advanced scoring function,



[PSM14b]

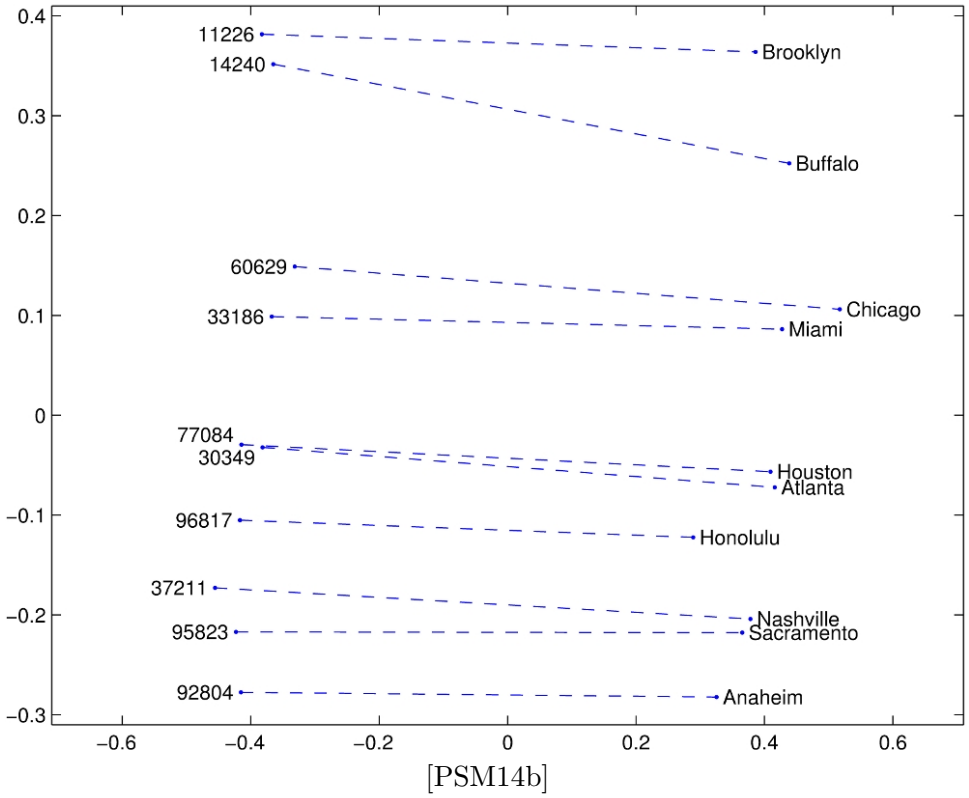
Figure 4.2: Linear substructures: comparative superlative

represent each word as a bag of n-gram and a word itself as another n-gram. To distinguish the beginning and end of words, they used special boundary symbols < and >. For example, on the word "where" and 3-gram for the character-level bag, it used character-level and word-level representation:

```
<wh, whe, her, ere, re>
<where>
```

On the character level (the first line), there are all 3-gram of the word, including the sequence "her" that would be misleading without having the boundary symbols < >. As a result, it computes vector even for words that are not in the learning dataset and keeps information about the morphology of words which is especially important for languages with large vocabularies containing many rare words and languages frequently using prefixes and suffixes for changing the meaning or for emotional colouring of words.

Paper [GBG+18] introduce word embeddings for 157 languages trained using **FastText** on Wikipedia and roughly 24 terabytes of raw text data from Common Crawl - a non-profit organization that crawls the web regularly and every month provides fresh data to the public. The FastText embeddings are current state-of-the-art for word embedding [GBG+18, MGB+18, BGJM16].



[PSM14b]

Figure 4.3: Linear substructures: comparative superlative

4.2.4 Multilingual Word Embeddings (MWEs)

Multilingual Word Embeddings contains words from multiple languages in common vector space which extends the representation with cross-language relation of words. Paper [CLR+17] introduced state-of-the-art FastText multilingual embedding aligned in a single vector space. There are two versions of it: supervised and unsupervised. The supervised was trained on bilingual data (dictionaries, Wikipedia), the unsupervised does not use any parallel data at all but adversarial training and iterative Procrustes refinement instead.

The unsupervised method is essential, especially for languages for which there are not enough of parallel corpora and so the supervised method is not efficient.

Supervised pre-trained models are available online [2] in 30 different languages, aligned in common vector space. There are models for the most frequent languages of the multilingual dataset 3.3.6 - Russian, Ukrainian, Polish, Czech, German, Slovak, Estonian etc.

The multilingual text understanding is nicely seen by exploring the nearest neighbours of a word in a language-specific subset of the common vector space.

[2] https://github.com/facebookresearch/MUSE

For the word 'frog' the nearest neighbours in English are similar as in GloVe embedding. Interesting is the close connection to 'tadpoles' - the larval stage in the life cycle of frogs and toads.

- 1.0000 - frog
- 0.7798 - toad
- 0.7620 - frogs
- 0.6847 - toads
- 0.6367 - tadpoles
- 0.6254 - salamander
- 0.5999 - salamanders
- 0.5786 - boulenger
- 0.5627 - amphibian
- 0.5606 - snake

In Spanish, the nearest neighbour of 'frog' (from the position of its word vector in the common space) are the following:

- 0.5785 - rana
- 0.5471 - ranas
- 0.5440 - sapo
- 0.5124 - bufonidae
- 0.5121 - tortuga
- 0.5081 - frog
- 0.5063 - lagartija

Note that the distance 0.5785 between 'frog' (English subset) and 'rana' (Spanish subset) is smaller than the distance from 'frog' to 'amphibian' which are both in the English subset. The connection is also visible in the visualization 4.4, 'frog', 'rana', 'toad', 'sapo' in one part of the projection, the plural forms 'frogs' and 'ranas' in other but not so far and words 'love', 'amor' in the opposite corner.

Finally, in the Czech language, the closest words for 'frog' are:

- 0.4707 - hadů *(genitive pluar) snakes*
- 0.4706 - žába *frog*
- 0.4559 - klokan *kangaroo*

- 0.4554 - strakapoud *Dendrocopos*
- 0.4516 - živočich *animal / animate being*
- 0.4506 - pavouků *(genitive pluar) snakes*
- 0.4463 - pták *bird*
- (five words skipped)
- 0.4308 - žáby

The words in cursive are the English translation of the Czech terms. The Czech language has difficult declension, for example, the first word in the list 'hadů' is in the genitive case (the term means 'without snakes') nominative case would be 'had' or 'hadi' in plural.

The closest words in the Czech subset was not as closed with its meaning as these in Spanish but also the distance is longer. The limitation is, at least in this case, that the Czech translation of 'frogs' *žáby* (the plural form) is not even in the top ten nearest words.

However, the nearest neighbours of 'žába' *frog* contains frog, frogs, and even tadpoles.

- 0.4753 - platypus
- 0.4726 - frogs
- 0.4706 - frog
- 0.4658 - fish
- 0.4658 - tadpoles
- 0.4527 - burrowing
- 0.4513 - pelican

And as the last example, the nearest neighbours of 'žába' *frog* in the Spanish subset:

- 0.5074 - nutria
- 0.4527 - rana
- 0.4515 - gato
- 0.4488 - sapo
- 0.4457 - ardilla
- 0.4422 - ranas
- 0.4418 - comadreja

Optimally, terms 'rana' and 'ranas' could be nearer than 'nutria', but there is still a connection between the words which supports the training hypotheses that words used in the same context tend to have a similar meaning.

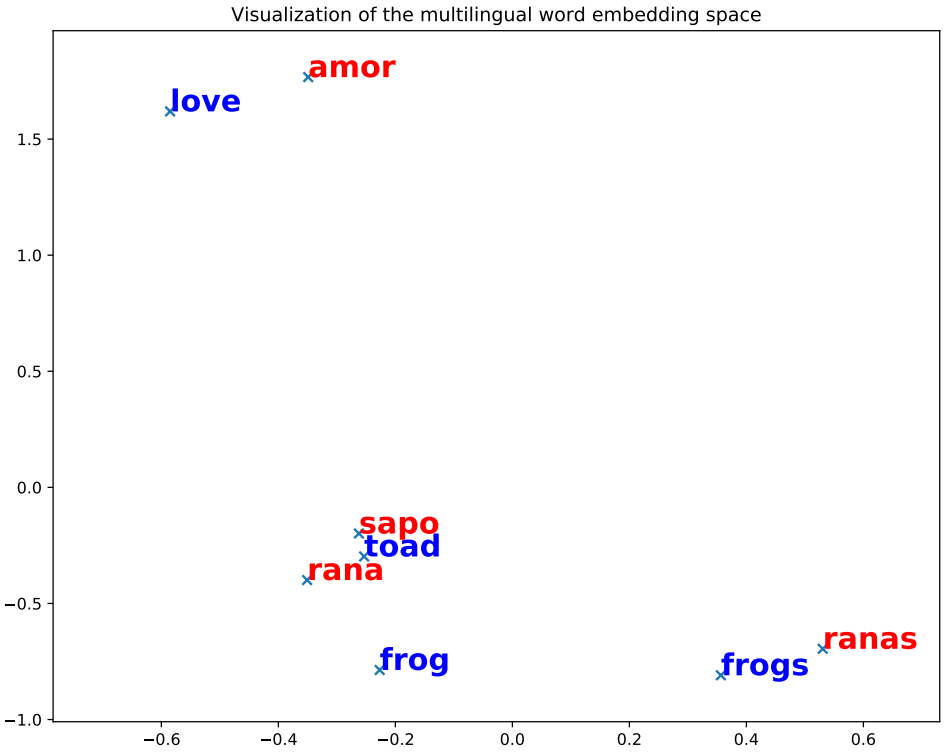


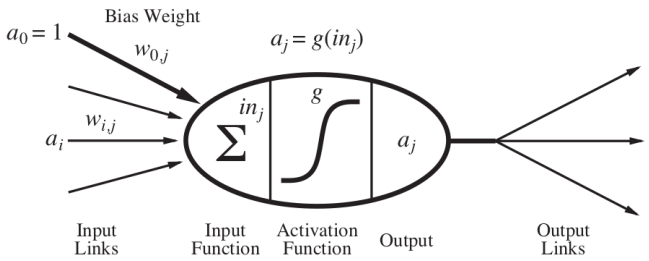
Figure 4.4: An example of a muntilingual word embeddings in a single vector space

4.3 Artificial Neural Network - NN

The creation of Artificial Neural Network was inspired by the findings in neuroscience, more specifically, by brain cells called neurons. The structure of a mathematical model for the artificial neuron can be seen in figure 4.5. The neuron (also called unit) on the figure is neuron j. is the output function of neuron i. On the left, the input links are multiplied by its weight. Note the dummy input with index zero called Bias Weight. On the right side, inside the neuron itself, there is the input function (sum) which is applied on the weighted input links. Then the activation function g is applied to give the output of the neuron. A mathematical expression of the process described is:

$$a_j = g(\sum_{i=0}^{n} w_{i,j} a_i)$$

Learning algorithms are adjusting the Bias Weight, in order to change the behaviour of the unit and so the neuron will learn. Finally, connecting these neurons and other types of units discussed in this section, form the (artificial) neural network. [RN09]



[RN09]

Figure 4.5: Mathematical model for a neuron

4.4 Recursive vs Recurrent Neural Network

Recursive Neural Networks is a class of NN using three architecture that allows the same set of weights recursively over its input. Recurrent Neural Network stands for the family of NN in which nodes are formed into directed graph including circles. This feature gives them the ability to process sequential data. [ZCL15] The notation of these two networks can be confusing since the acronym RNN is sometimes used for both of them. However, in this work, RNN stands exclusively for Recurrent Neural Networks, and the Recursive Network architecture is not applied here.

4.5 Recurrent Neural Network - RNN

RNN has directed cycles in the connection graph from old to new inner state. Unlike feed-forward networks, this feature allows RNN to store some inner state, making them able to pass information across sequential steps. Therefore, RNN is suitable for tasks where data points are related, such as video processing (each frame depends on series of previous and following samples), audio processing or in this case - text processing. In these applications, data frames are not independent. It is necessary to learn the text as a sequence and remember the inner state by processing the entire document to get the desired level of understanding.

On the other hand, it turns out to be hard to write an efficient learning algorithm for RNN. Mainly due to a problem with vanishing/exploding gradient and oscillating weights.

The problem is challenging since it is the main feature of RNN that causes the vanishing of the gradient. In order to preserve the inner state, long-term information has to go through all the cells over and over again through the loop in the directed graph forming RNN. That also means it can be multiplied by a number close to zero many times during the learning period and so the gradient can vanish.

Another problem is the high occurrence of over-fitting in RNN applications. Back Propagation Through Time (BPTT) stands for extension of Back

Propagation capable of modelling time - algorithm often used to learn RNN.

4.5.1 Long Short Memory - LSTM

Many studies developing more and more complex LSTM based networks for natural language processing had been published, thanks to the development of new learning algorithms and advanced architectures, especially the Long Short Term Memory (LSTM) introduced by Hochtreiter and Schimhuber. They proposed to create additional memory gate units responsible for the inner state. **Input gate unit** protects the inner state from perturbation by irrelevant input. **Output gate unit** protects other units in the network from irrelevant memory content.[SH97]

4.5.2 Gated Recurrent Unit - GRU

Gated Recurrent Unit (GRU) is another architecture preventing vanishing/-exploding of the gradient. Similarly, as LSTM, GRU also adds more gates to the networks. **Reset gates** Determinate how much of the information from the previous state is added to the hidden state, **update gates** specify the proportion of information from hidden state and previous state to be added to the new state.

According to [KGS17], these special unit prevent the vanishing gradient problem. Also, they are able to learn much longer sequences. However, additional gates and more complex architecture causes higher computational cost. It was observed that certain modification such as coupling the input and output gates and removing peepholes connection simplified the LSTM architecture without any significant loss of performance. In the case of GRU, thanks to compiling input and output gates, even forget gate and output activation function can be omitted without significantly decreasing performance. Unlikely of LSTM, where these components were found to be critical.

Bi-directional Recurrent Neural Networks introduced by Schuster and Paliwal, 1997 extends RNN to model dependence on both past and future states.

4.6 Convolutional Neural Network - CNN

Although CNNs are widely used in image processing tasks, their application is not so limited. In the last few years, there were published several new methods using CNN in the sequential data processing. Before diving deeper, let us refresh how a simple CNN for image classification works. Consider an image as a matrix, where each value is a three-dimensional vector with each value in an interval from 0 to 255 representing the RGB colour model. For simplicity, consider just grey-scale images with one channel only instead of three RGB channels. Then each value of the matrix is one number only, usually normalized to be in 0 to 1 interval.

After (optional) preprocessing, the first step is convolution. We take a smaller matrix, so-called convolutional filter, put it on the right top corner of the image, multiply all the corresponding numbers, and sum the result. Then shift the filter by one or more pixels to the right and after we reach the right edge of the image, shift it down by the same margin, called stride length, and start again from the left. The sum of multiplication is always written into another matrix, called a convoluted feature, the result of this operation. The number of these filters, its size and stride length are the main parameters.

The convolutional layer is usually connected to a max-pooling or average-pooling layer. It takes the convolution filter as an input, divides it to sub-matrixes of some size and chooses a maximum of each of them. In the case of average-pooling, it just computes the average within the sub-matrix instead of the maximum.

This procedure was the case of image processing, but the design for text processing is surprisingly similar. Instead of the two-dimensional matrix representing a picture, there is a sentence matrix - n x k representation of the input sequence (a single sentence or entire article), n is the length of the input sequence, k is the dimension of word embedding. Having a input sequence:

$$x_{i:n} = x_1 \bigoplus x_2 \bigoplus \cdots \bigoplus x_n \tag{4.1}$$

Where $\bigoplus$ is the concatenation operator. The convolutional filter $\mathbf{w}$ has dimension h x k, where h is a window of h words to be involve in the operation. A new feature is then compute by:

$$feature = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b) \tag{4.2}$$

The convolution layer helps to understand a word within its context. The pooling layer extracts essential words or phrases and is usually followed by a fully-connected layer.

4.6.1 Attention mechanism

Neural attention in Machine learning is a mechanism that allows the network to focus on a specific subset of input. This architecture is many times used together with RNN. However, **Transformer model** introduced in 2017 uses the attention mechanism solely without the usage of RNN or CNN. Moreover, in some applications such as translation task Transformer outperforms any other architecture. [AV17]

For Semantic Role Labeling (SRL) task, a self-attention based model introduced in 2018 outperformed previous state-of-arts approaches. The model consists of several RNN/CNN/FNN sub-layers followed by attention layer. Although still using RNN, it also gives a better computational performance.[ZT18]

SRL is a process that assigns a label to each word in a sentence to indicate their semantic role in the sentence.

There is also a promising attention-based model particularly suitable for question answering problem. Attention Sum Reader, [RK16], 2016 achieved

state-of-art in several datasets, including News Articles dataset that consists of CNN and Daily Mail website. The article body was used as a context, and the question was formed from the headline. It suggests that Attention Sum Reader model could be also efficient for text classification since it was able to get the main "topic" from the text and compare it with the headline. The disadvantage of answering problem application is that this model cannot give an answer, which is not included in the text. However, it does not mean anything for an application in text classification tasks.

Chapter 5

Experiments

This chapter contains several experiments. Fake News detection is addressed as a text classification problem. First, the dataset and hardware used are presented, followed by a description of used NN models and finally a discussion about the impact of preprocessing on training time, computation cost and final accuracy.

5.1 Dataset and hardware

All experiments in this chapter were done on the Kaggle3.1 dataset of 20.8k entries. The NN models and many preprocessing approaches on this single dataset to choose which of them use for another research. The dataset was divided into a train and test part in 0.33 ratio, and 100 samples from the train part were used for validation (that was used for early-stopping). Every model in this initial experiments was designed in Keras and trained using one GeForce GTX 1080 GPU, 2 CPUs and 12GB of RAM (if not specified differently).

The input documents were shortened to the length of 300 encoded-words (in some tests to 1 000, 3 000 or even full sequences were processed) to improve the training speed. The cut was always done as the last step of preprocessing. The effects of different preprocessing are part of the discussion later.

Each experiment ran 15 training epoch, with cross-entropy loss function and Adam optimizer. The validation set was used for early-stopping, but when the time was measured, the early stopping function was turned off.

5.2 Neural Network models

In this section, models used for the initial experiments are presented. The input sequences were pre-processed by 4.2.1 with min_df=3 and max_df=0.9 options. In order to speed up the training and save computing cost, the input length is cut to 300 words (if not specified differently). The cut was always done as the last step of preprocessing.

5.2.1 Simple Sequential

Input text data are reduced by stopwords from **nltk** library and transformed to Bags of Words representation by CountVectorizer from **sklearn** library as described in 4.2.1.

The network is a classical NN with only one fully-connected hidden layer (in the Keras summary 5.2.1 called Dense) consisting of 256 neurons with ReLu activation function, followed by an output layer with one neuron and sigmoid activation function giving the true/false classification. The same output layer is used in every other model tested in this chapter.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 256)               13671936
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 13,672,193
Trainable params: 13,672,193
Non-trainable params: 0
```

This simple model was very fast to train even with full input length (the longest sequence after preprocessing is 11 985 words long). It performed with an unexpected accuracy of 0.97 on test set even though it never passed over 0.96 accuracy on the validation set (it might be caused by its small size). Training loss kept decreasing down to 9.6642e-05 after 15 epochs, not so validation loss which got slightly worse during the training ending on 0.2. Neither more training epochs or hyperparameter tuning did not bring any significant improvement.

These results were obtained on Bags of Words matrix, which was only restricted by NLTK stopwords, min_df=3, and max_df=0.9 as explained in 4.2.

On the other hand, if the input length is cut to 300 words only, the accuracy drop to 0.88.

Bags of Words representation obtained by CountVectorizer is based on the word frequencies only without any other information. That means the classification is made just based on the vocabulary, showing that, at least in this dataset, the vocabulary of Fake News differs a lot from the vocabulary of reliable news.

5.2.2 Embedding layer

Embedding layer passes a deeper understanding of the input text to the following layers. It is the first hidden layer of every model except the Simple-Sequentional described before. The embedding layer is initialized randomly to learn the dense vector representation, described in detail in section 4.2.3. In other tests later, it is initialized with pre-trained embedding. It requires the input data to be integer encoded. That means the input sequence is a

series of integers representing an index of the *word_index* dictionary. Each integer in the sequence corresponds to a word that was on that place in the document encoded.

5.2.3 SimpleLSTM

The model was extended with a 300-dimension Embedding layer and the full-connected layer was replaced by LSTM (see 4.5.1) of the same number of units (256).

In the Keras summary 5.2.3, the output shape of the input layer is the length of the input. The following embedding layer turns each (integer encoded) word of the input sequence into a 300-dimensional dense vector shaping its output into 300 (input length) x 300 (embedding dimension) matrix.

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 300)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 300, 300)          16021500
_________________________________________________________________
lstm_1 (LSTM)                (None, 256)               570368
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,592,125
Trainable params: 16,592,125
Non-trainable params: 0
```

Listing 5.1: Keras summary of the SimpleLSTM model

With input sequence restricted to 300 words, this model ends up 0.927 accuracy on the test set (higher than 0.88 in case of Simple Sequential model with CountVectorizer and the same length of input), 0.92 on the train set. As can be seen in figure 5.1, after eight training epochs it managed to perform with 1.0 accuracy on the train set. However, it never broke 0.95 accuracy on a validation set. The network probably was not able to generalize.[RD17] Moreover, the training time was significantly slower. It took about 1 minute per epoch on the short input length (cut to 300 words) when the simple sequential needed just a few seconds for full-size data.

5.2.4 SimpleCNN

Now, there is the same input and embedding layer as in the previous model, followed by one-dimensional CNN as described in 4.6, ending with fully-connected layer and an output layer with the same parameters as the first model in this chapter.

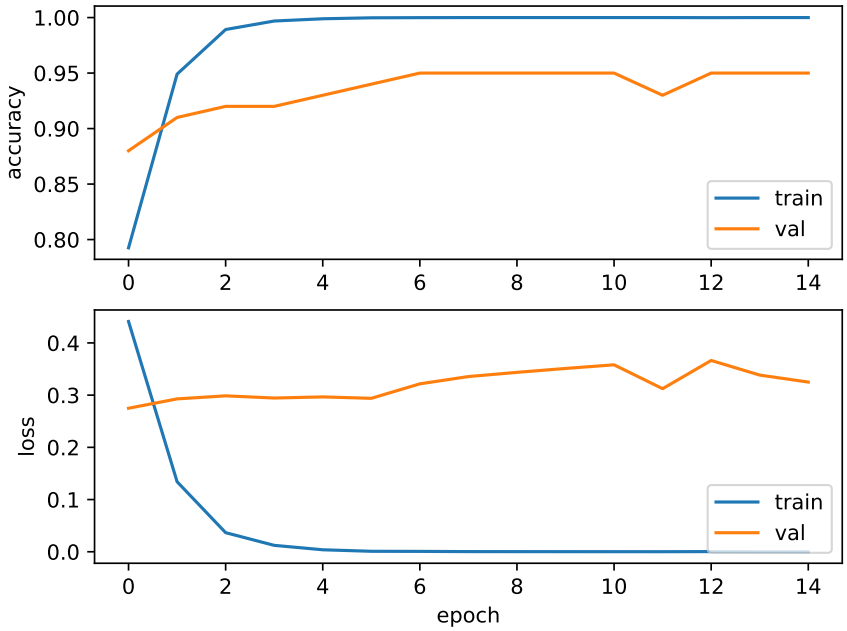


Figure 5.1: SimpleLSTM performance, input length of 300 words

```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 300)               0
_________________________________________________________________
embedding_2 (Embedding)      (None, 300, 300)          16021500
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 296, 64)           96064
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 59, 64)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3776)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               966912
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 17,084,733
Trainable params: 17,084,733
Non-trainable params: 0
```

Listing 5.2: Keras summary of the SimpleCNN model

The input was again cut to 300 vectors only in the same way as in the case of the LSTM model, which was outperformed with final accuracy 0.954 on the test set and 0.24 loss. As figure 5.2 shows, it also learned faster as the

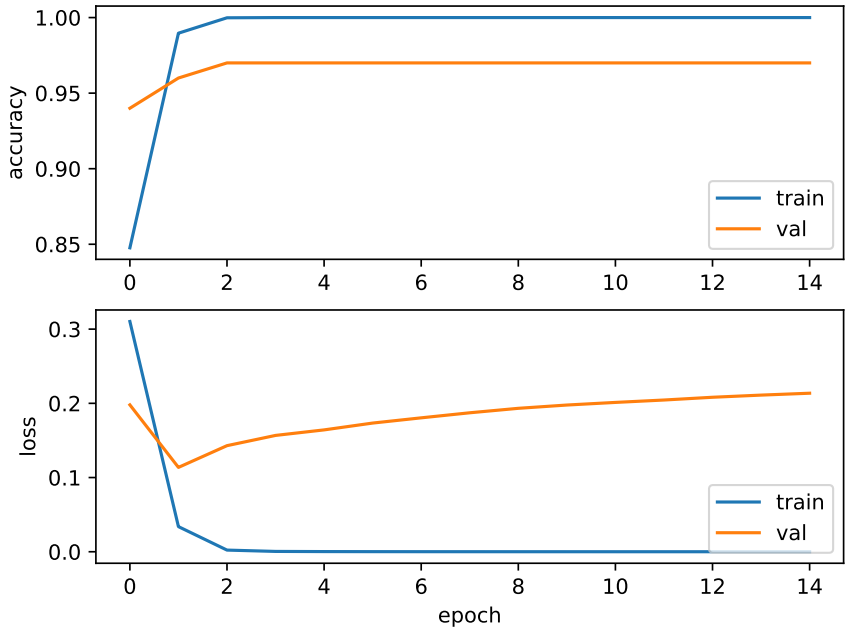


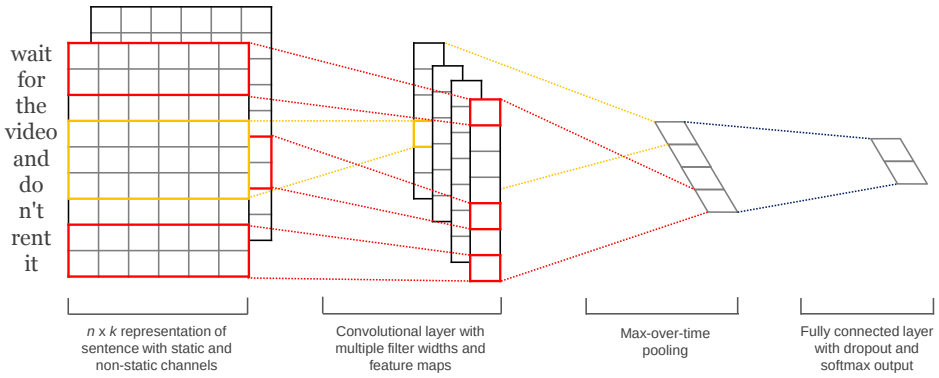
Figure 5.2: SimpleCNN performance, input length of 300 words

training accuracy has risen to 1.00 right on the end of the fourth epoch and loss of 4.4330e-04 that kept decreasing down to 9.6733e-06 at the end of the last 15th epoch. Moreover, the training time was significantly lower as each epoch took 2-8 seconds only. Validation loss, on the other hand, was getting slightly worse over the training time.

Next, the model was trained on full-size input sequences with a vocabulary of 5000 most common words. The number of trainable parameters has risen to 28,794,017 (including embedding layer) and so has risen the test accuracy to 0.96.

5.2.5 TextCNN

This model was inspired by paper [Kim14]. The input and embedding layer is the same as before, but then, instead of just one Convolution and MaxPooling layer, there are four of them, connected in parallel as shown in the Keras summary below (note the 'Connected to' column) and figure 5.3. The input sentence matrix of dimensions n x k is a representation of the input sequence (article), n is the length of the input sequence, k is the dimension of word embedding. In this example, the first row of the matrix is loaded 300-dimensional embedding vector of the word "wait". The convolutions are computed as described in 4.6 but there are several convolutional layers connected in parallel.



[Kim14]

Figure 5.3: TextCNN Architecture

```
Layer (type)         Output Shape      Param     Connected to
==================================================================
i1 (InputLayer)      (-, 300)          0
------------------------------------------------------------------
e1 (Embedding)       (-, 300, 300)     16021500  i1[0][0]
------------------------------------------------------------------
r1 (Reshape)         (-, 300, 300, 1)  0         e1[0][0]
------------------------------------------------------------------
c1 (Conv2D)          (-, 300, 1, 36)   10836     r1[0][0]
------------------------------------------------------------------
c2 (Conv2D)          (-, 299, 1, 36)   21636     r1[0][0]
------------------------------------------------------------------
c3 (Conv2D)          (-, 298, 1, 36)   32436     r1[0][0]
------------------------------------------------------------------
c4 (Conv2D)          (-, 296, 1, 36)   54036     r1[0][0]
------------------------------------------------------------------
mxp1 (MaxPooling2D)  (-, 1, 1, 36)     0         c1[0][0]
------------------------------------------------------------------
mxp2 (MaxPooling2D)  (-, 1, 1, 36)     0         c2[0][0]
------------------------------------------------------------------
mxp3 (MaxPooling2D)  (-, 1, 1, 36)     0         c3[0][0]
------------------------------------------------------------------
mxp4 (MaxPooling2D)  (-, 1, 1, 36)     0         c4[0][0]
------------------------------------------------------------------
conc1 (Concatenate)  (-, 4, 1, 36)     0         mxp1[0][0]
                                                 mxp2[0][0]
                                                 mxp3[0][0]
                                                 mxp4[0][0]
------------------------------------------------------------------
f1 (Flatten)         (-, 144)          0         conc1[0][0]
------------------------------------------------------------------
dr1 (Dropout)        (-, 144)          0         f1[0][0]
------------------------------------------------------------------
de1 (Dense)          (-, 1)            145       dr1[0][0]
==================================================================
```

Listing 5.3: Keras summary of the TextCNN model

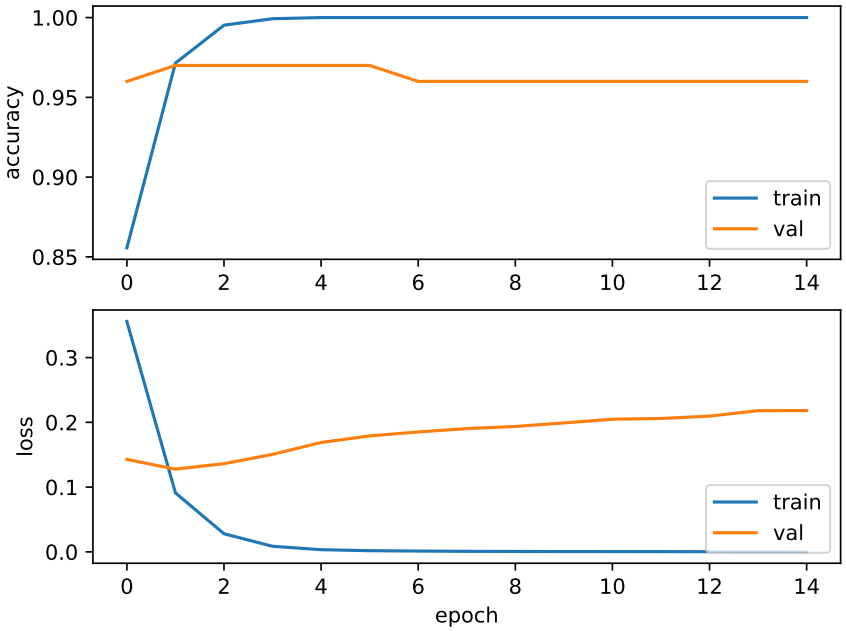


Figure 5.4: TextCNN performance, input length of 300 words

Later, it was extended by an extra fully-connected (Dense) layer between the Dropout dr1 and Dense de1 layers which improved its performance at had a negligible impact on the training time.

CNN showed its potential in text classification tasks. With the short input as before, it was able to achieve 0.968 accuracy on the test set with 0.1 loss. The learning time dropped was only 4-8 seconds per epoch.

This model outperforms all the previous. When trained on full-length input (preprocessed in the same way as for SimpleCNN network), it scored with 0.98 accuracy with about 87 second per epoch which is comparative with LSTM model on short data (input length of 300 words).

5.2.6 2biLSTM with Attention

Once again, the input layer and embedding are the same. Two LSTM layers are bi-directional followed by Attention layer. There is no implementation of an attention layer in Keras yet; therefore a third-party open-source implementation was used [ker19].

	CountVectorizer	TFIDFVectorizer
acc	0.8842	0.9620
time per epoch (s)	6.0	6.1

Table 5.1: NN performance with the comparison of Count and TFIDF Vectorizer on the Simple Sequentional model

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 300)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 300, 300)          16021500
_________________________________________________________________
bidir_1 (Bidirection LSTM)   (None, 300, 512)          1140736
_________________________________________________________________
bidir_2 (Bidirection LSTM)   (None, 300, 256)          656384
_________________________________________________________________
attention_1 (Attention)      (None, 256)               556
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
```

Listing 5.4: Keras summary of the 2biLSTM with Attention model

Difficulties occurred while training these network as the validation accuracy and loss is inconsistent. Also, the learning time per epoch was the slowest from all models tested, around 230 seconds. Attention layer performs better with an adaptive learning rate - starting very low, slowly rising to a maximum after, typically, 4 000 epochs and then being decreased, this setting was not applied here since a different architecture has been found more efficient later. The final accuracy was only 0.9268.

Vectorizer compare

CountVectorizer 4.2.1 just counts the word frequencies. TFIDFVectorizer keep more information, see 4.2.2. These two vectorizers were compared using Simple Sequential model. There was a big gap in their performance. As can be seen in table 5.1 TFIDFVectorizer had 0.96 accuracy on the test data when CountVectorizer ends up with only 0.88 with almost the same time needed.

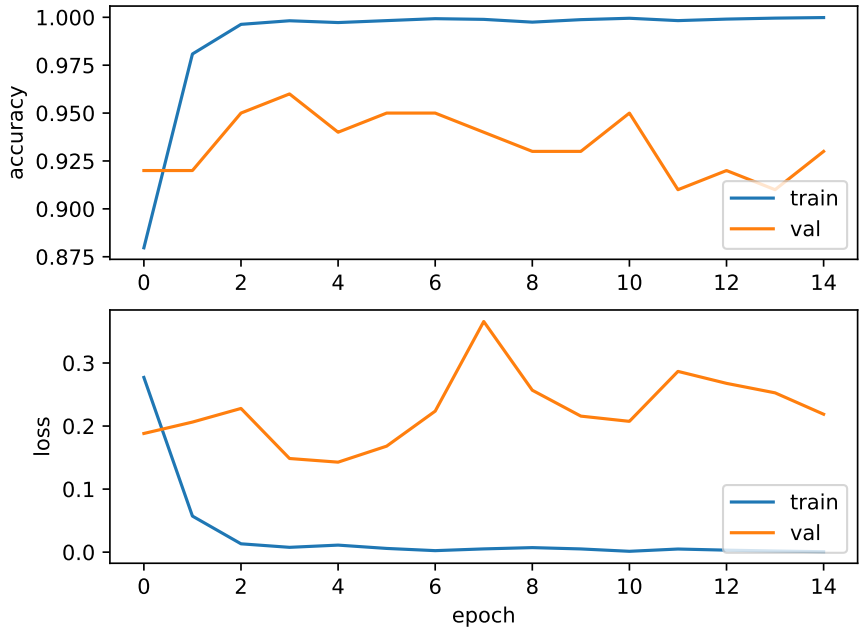


Figure 5.5: 2biLSTM with Attention performance, input length of 300 words

5.3 The effect of pre-processing

In this section, we study the effect of preprocessing the data, especially filtering vocabulary and using pre-trained word embedding.

The full vocabulary of the dataset used in this chapter is 146 211 words; most of them are numbers. Words with low meaning impact can be filtered out by stopwords, by the number of minimum occurrences or by setting the maximum size of vocabulary and fill it with the most common words.

Several options are tested in the following subsections. All of them with input sequences cut to 300 words, all words transformed to lowercase and with the vectorizer 4.2.1 set to min_df=3, max_df=0.9. It makes the vocabulary size to drop from 146 211 to 53 608.

The stated accuracy is obtained during the evaluation on the test set divided in the same way as before. The training and validation loss is from the last - 15th epoch of training. The time is an average time per epoch during the training.

Filtering vocabulary from unnecessary words would be beneficial, especially for models struggling with long input sequences. That is not a case of Simple Sequential model. Therefore it is omitted in the test. Also, it does not support word embedding.

	SimpleLSTM	SimpleCNN	TextCNN	Attention
acc	0.9269	0.9543	0.9677	0.9268
loss	0.4791	0.2399	0.1032	0.4810
training loss	4.2e-5	9.7e-6	1.8e-4	6.3e-4
validation loss	0.3250	0.2136	0.2195	0.2187
time per epoch (s)	53.5	2.4	4.3	230.9

Table 5.2: NN performance with no additional restriction on the vocabulary, input length of 300 words

	SimpleLSTM	SimpleCNN	TextCNN	Attention
acc	0.9279	0.9538	0.9673	0.9464
loss	0.2948	0.3160	0.1300	0.3271
training loss	0.0305	6.4e-05	1.7e-04	0.0060
validation loss	0.1993	0.1775	0.0928	0.2797
time per epoch (s)	48.9	1.3	4.0	243.2

Table 5.3: NN performance with restriction on the vocabulary size only, input length of 300 words

5.3.1 No restriction on vocabulary

The first, baseline run is with no other restriction besides the setting mention above, results in table 5.2

5.3.2 Maximum number of features

The vocabulary can be restricted by setting the maximal number of feature to be considered by the vectorizer 4.2.1. In this case, it was set to 5000, meaning that only 5000 most common words are given to the vocabulary and considered for training. Compare to the vocabulary size (after applying minimum and maximum words frequencies as described before) that was 53 608 and the original size of 146 211 this brings noteworthy computation cost-saving but also filtered out many numerical values that might carry crucial information.

The results (in table 5.3) do not vary a lot compared to the previous training (5.2) without any vocabulary restriction. In most cases, the differences are so small it can be considered as a statistical error. It suggests that most of the numerical values were not as important as one could have thought and that more than 96% of the vocabulary can be omitted without any significant loss of performance.

5.3.3 Stopwords and maximum number of features

Using both stopwords and 5000 as the maximum number of features gives following results 5.4.

	SimpleLSTM	SimpleCNN	TextCNN	Attention
acc	0.9260	0.9543	0.9569	0.9419
loss	0.3225	0.1600	0.2183	0.3136
training loss	0.0188	7.2e-04	0.0095	0.0075
validation loss	0.2628	0.0955	0.1135	0.4263
time per epoch (s)	53.2	1.4	4.0	226.7

Table 5.4: NN performance with both stopwords and restriction on the vocabulary size used, input length of 300 words

	SimpleLSTM	SimpleCNN	TextCNN	Attention
acc	0.9376	0.9557	0.9678	0.9401
loss	0.4250	0.2147	0.1028	0.3772
training loss	1.9e-5	9.7e-6	2.0e-4	3.7e-4
validation loss	0.2271	0.2053	0.2232	0.4867
time per epoch (s)	53.1	1.3	4.0	257.13

Table 5.5: NN performance with only stopwords, no restriction on the vocabulary size, input length of 300 words

5.3.4 Stopwords only

Another test was done to distinguish between the effect of stopwords and restriction on the maximum number of features. This time with stopwords only, results can be seen in table 5.5

5.3.5 Vocabulary restriction conclusion

The original corpus consists of 146 211 terms (words and numerical values). Table 5.6 shows the effect of preprocessing on vocabulary size and the length of the longest sequence.

All the experiments described in section 5.3 show that cutting the vocabulary size from 146 211 to as low as 5 000 have a negligible effect on the final accuracy. This preprocessing cuts of a large number of numerical values from the corpus. It is a common practice to do so in text classification tasks and works well on this dataset. The results show, there is no need to use the entire vocabulary for every experiment, but it might be beneficial for the final model even for the price of higher computation cost - especially when using an evidence-aware model, which might uncover even small lies in numbers.

5.3.6 Input length

The experiments on vocabulary size and cutting it from 146 211 to 5 000 have shown only a low or no impact on the final accuracy on the test dataset and on the loss. There was no noticeable impact on the training time. However,

	vocabulary size	max sequence length
No pre-processing	146 211	23 374
+ min_df=3	53 608	23 077
+ max_df=0.9	53 603	19 741
+ stopwords	53 464	11 985
+ max_features=5000	5 000	9 830

Table 5.6: The impact of pre-processing on the vocabulary size and length of the longest sequence in the corpus

	SimpleLSTM	SimpleCNN	TextCNN	Attention
acc	0.9363	0.9620	0.9827	0.9640
loss	0.3890	0.2165	0.0564	0.2252
training loss	3.9e-4	1.3e-5	4.5e-4	0.0049
validation loss	0.3677	0.1318	0.0680	0.0823
time per epoch (s)	443.7	8.7	37.1	876.3

Table 5.7: NN performance with input length of 1000 words, pre-processed with stopwords, vocabulary restricted to 5000 most frequent words

the number of training parameters have dropped significantly, and so drop the amount of GPU memory necessary for the training.

The length of input sequences used in previous tests was 300 words only (the cut was always done as the last step of preprocessing). In this part, the same models are evaluated with longer input length (1000 words in each input sequence) with both stopwords and vocabulary of 5000 most frequent words.

As we can see in table 5.7, convolution models improved their performance by more than 0.1 in accuracy. The improvement was bigger than in any preprocessing experiments before. It shows the importance of processing entire articles instead of their shorter version. LSTM model ended up with similar results as before. It shows the struggle of RNN/LSTM based networks in learning long-term dependencies. [SHS01]

Long-term dependencies might be crucial in this application. Fake News articles can contain valid information in the majority of the text and only in a small part include a piece of information that changes the overall meaning. The valid text can be used to attract reader, gain his trust in facts and then turn it into a misleading impression. If the entire article is not used, this information can be lost, and even the best classifier has no chance to come up with the correct prediction.

Another important aspect of this experiment was the time per epoch. With longer input, the time grew in all cases. The average time for TextCNN was 37s instead of 4s but also 443.7s instead of 53.1s for LSTM.

CNN-based models are able to process even full-length data, but LSTM-based models run out of GPU memory (11GB).

This test once again confirms the advantages of CNN architectures in

	SimpleLSTM	SimpleCNN	TextCNN	Attention
acc	0.9617	0.9470	0.9679	0.9650
loss	0.1426	0.2930	0.1329	0.0965
training loss	0.0173	3.6e-5	0.0048	0.0834
validation loss	0.0923	0.2962	0.1444	0.0492
time per epoch (s)	51.3	1	3.1	231.9

Table 5.8: NN performance with GloVe embedding, input length of 300 words

prediction accuracy, training speed, and computation cost. Moreover, it shows the importance of learning (if possible) long input sequences. If the computation cost would be too high, the problem with extremely long input sequences could be workaround by following steps;

- use reasonable long sequences for training
- split sequences for classification into smaller sub-sequences
- if any of them is predicted to be unreliable, mark the entire sequence as unreliable.

5.3.7 Word embeddings

The parameters of the Embedding layers were trained together with the succeeding layers. In the following test, pre-trained embedding is used and is not trainable.

Both stopwords and maximum features restriction were used with the same vectorizer options as before, but pre-trained word embeddings were loaded and used as weights of the Embedding layer.

GloVe

First, GloVe 4.2.3 **glove.840B.300d** embedding was used with 300 sequences length. This was the first test when LSTM based models (LSTM and Attention) outperformed SimpleCNN and were comparative with TextCNN as can be seen in table 5.8. However, there is still a huge gap in 1s per epoch training for TextCNN and 231.9s for Attention network.

In any case, this test shows that RNN/LSTM layers are still powerful tools and are used in the next experiments.

The same test was run once again with 1000 input sequences length, see the results in table 5.9

*Attention model was trained using 4 CPUs; other hardware specification stays unchanged. Even with double CPU power, it took 781.4s per epoch to train, and the final accuracy was still lower compared to TextCNN model that learned almost ten times faster.

The longer input length did not bring almost any improvement to LSTM model. It might be due to losing long-term information in LSTM layer. On the other, CNN based and Attention models end up with higher accuracy

	SimpleLSTM	SimpleCNN	TextCNN	Attention
acc	0.9627	0.9538	0.9815	0.9778*
loss	0.1053	0.2299	0.0686	0.0653*
training loss	0.0982	1.4e-4	0.0021	0.0485*
validation loss	0.0917	0.1019	0.0134	0.0226*
time per epoch (s)	185.1	2.3	8.3	781.4*

Table 5.9: NN performance with GloVe embedding, input length of 1000 words. * - this test was run on 4 CPUs

compare the 300 length run. The training of the TextCNN model is still extremely fast - 8.3s.

5.4 Initial experiments and pre-processing conclusion

The experiments showed that even strict restriction on the vocabulary does not influence the results significantly. The length of the longest article is less than half of the original when encoded using a filter on the less and the most frequent words (min_df, max_df) together with stopwords. The original length was 23 374, after the preprocessing, it is only 11 985. With the additional filter on maximum 5000 words used, the length is 9 830 which do not bring a big saving, and the performance in classification was slightly worse, compared to using stopwords and min_df, max_df only. Therefore the maximum number of features is restricted in future experiments.

A noteworthy test was training with longer input length, mainly CNN based network gave significantly better results and kept fast training performance. On the other hand, the training time of LSTM based models have risen to hundreds of second per epoch (compared to less than 10s needed for CNN), and their accuracy did not improve (LSTM case) or did not improve as much as expected (Attention case). Both LSTM models (LSTM, Attention) needed more hardware resources for input longer than 1000 vectors.

The most important test was comparing pre-trained embedding or learning it on the dataset. Using pre-trained embedding on an enormous amount of data brought more considerable improvement than any other test.

Another interesting observation was the success of the Simple Sequential model showing there are notable differences in the vocabulary of Fake News and reliable news, respectively. However, it is a low-level understanding and can be dataset-specific. The authors of the dataset used in this chapter do not give any information about which types of news were used as "reliable" samples. Later, this model is used on a self-obtained dataset containing news of similar topics in both reliable and unreliable parts. It might give better insight into the problem, showing if the vocabulary used in disinformation is really different or if only the topics in unreliable media vary.

	SimpleLSTM	SimpleCNN	TextCNN	Attention
No restriction on vocabulary				
acc	0.9269	0.9543	0.9677	0.9268
time per epoch (s)	53.5	2.4	4.3	230.9
Maximum number of features				
acc	0.9279	0.9538	0.9673	0.9464
time per epoch (s)	48.9	1.3	4.0	243.2
Stopwords and maximum number of features				
acc	0.9260	0.9543	0.9569	0.9419
time per epoch (s)	53.2	1.4	4.0	226.7
Stopwords only				
acc	0.9376	0.9557	0.9678	0.9401
time per epoch (s)	53.1	1.3	4.0	257.13
Input length of 1000 words				
acc	0.9363	0.9620	0.9827	0.9640
time per epoch (s)	443.7	8.7	37.1	876.3
Pre-trained GloVe				
acc	0.9617	0.9470	0.9679	0.9650
time per epoch (s)	51.3	1	3.1	231.9
Pre-trained GloVe and input length of 1000 words				
acc	0.9627	0.9538	0.9815	0.9778*
time per epoch (s)	185.1	2.3	8.3	781.4*

Table 5.10: A brief summary of the main pre-processing configurations tested

Table 5.10 summarize the accuracies and training times of the preprocessing configuration tested in this chapter.

5.4.1 Keras vs PyTorch

The models and the training flow in this chapter were developed in Keras (version 2.2.4) with TensorFlow (version 1.13.1) backend and Binary Cross-Entropy loss function. The two libraries, Keras (with TensorFlow backend) and PyTorch, were compared on a simple LSTM model.

Surprisingly, a corresponding network to 5.2.3 designed in PyTorch gave significantly better results, especially when running with BCEWithLogitsLoss (a combination of a Sigmoid layer and Binary Cross Entropy) loss function and pre-trained word embeddings. Moreover, the training time of the LSTM layer was much faster compare to Keras, especially for longer input sequences 5.13. The data were prepared using TorchText preprocessing build-in functions. It contains everything needed for text processing in a single library; data loading, building vocabulary, loading embedding vectors and even an iterator preparing training tensors of given batch size. Minimum word frequency was set to 3 and **NLTK word_tokenize** was used. Table 5.11 compares the performance of Keras and PyTorch, respectively, on the Simple LSTM model

	Simple LSTM		TextCNN	
	(-)	with GloVe	(-)	with GloVe
acc	0.9269	0.9617	0.9324	0.9945
loss	0.3225	0.1426	0.2640	0.0200
training loss	0.0188	0.0173	0.0186	0.0082
validation loss	0.2628	0.0923	0.1918	0.0898
time per epoch (s)	53.5	51.3	9.37	8.2

Table 5.11: Keras (left part) vs PyTorch (right part) LSTM model. The first run is without pre-trained embedding (-), the second with Glove

first with no additional pre-processing and without pre-trained vectors (as in table 5.2) and with pre-trained GloVe embedding (as in table 5.8)

There are two critical observations. First, the training time drops from around 52 seconds per epoch to just 8-9 second when using PyTorch. Second, the accuracy of the LSTM model written in PyTorch and using GloVe vectors was the highest achieved so far. It might be caused by the tokenizer used (nltk compare to scikit used before) or by slightly different preprocessing (86 017 words were left in the corpus compare to 53 603). Either way, the time saved was so striking that there is no more reason to worry about more extensive vocabulary.

5.4.2 Summary

This chapter introduced five different Neural models written in Keras and two of them were rewritten to PyTorch for comparison. A big effort was put to the comparison of typical preprocessing frequently used in text classification. It was shown that the vocabulary of the input data could be restricted significantly without a relevant loss on the final test accuracy.

Later the importance of pre-trained word embedding was proven to bring a larger impact on the final accuracy than any other experiment.

Crucial was also a comparison of Keras (with TensorFlow backend) and PyTorch. The same architecture achieved higher accuracy when written and trained in PyTorch. Moreover, the training time dropped by more than 80%.

When using an optimized model and pre-trained embedding, there is no need for a strick preprocessing. Skipping words that occur less than three times is enough. The GloVe embedding provides a strong understanding of the natural language and even relationships between some numerical values and words that are commonly used in the same context, for example, a relation of city and its zip code, as can be seen in figure 4.3. If the vocabulary is reduced too strictly, these numerical values having a relationship in the embeddings might be lost.

Lastly, as we can see in the summary in table 5.10, using only 300 (encoded) words from each article is not enough, and at least 1 000 words should be used. Longer input brings higher accuracy. On the other hand, it also causes higher computation cost and training time of LSTM layers.

	Accuracy
With input length of 300 words	
no pre-trained embedding	0.9977
GloVe	0.9986
FastText	0.9968
With input length of 1000 words	
no pre-trained embedding	0.9936
GloVe	0.9984
FastText	0.9980
GloVe + embedding training	0.9991
FastText + embedding training	0.9985

Table 5.12: TextCNN accuracy with different embeddings. In the last two rows (with *+ embedding training*), the embedding layer weights were also adjusted during the training.

	Input length	Keras	PyTorch
SimpleLSTM	300	51.3	9.7
TextCNN	300	3.1	6.6
TextCNN	1000	185.1	13.3
TextCNN + embedding training	1000	NaN	20.2

Table 5.13: Training time per epoch (s) with pre-trained GloVe embedding.

The most successful approach

TextCNN outperformed every other model in all tests 5.10 presented in this chapter. Another crucial observation is the importance of pre-trained embedding. Table 5.12 compares TextCNN accuracies with GloVe, FastText and no pre-trained embedding. The highest accuracy achieved was 0.9991, with only five false positives and one false negative prediction out of 6864 samples in the test set. It was achieved using GloVe embedding when the embedding layer was included in the optimization to adjust the embedding for specific needs of the dataset. In contradiction to [BDK14], a predict-based GloVe embedding performed better than count-based FastText. It might be caused by a specifics of the dataset used rather than an observation that could apply to the Fake News detection in general.

Chapter 6

Experiments on self-obtained datasets

Considering the results of the initial experiments presented in chapter 5, with the most crucial observation summarized in 5.4.2, the following experiments were prepared in PyTorch with emphasis on training longer input sequences. The vocabulary was restricted only by filtering out words that occur in the entire corpus only three times or less.

Two best models from the previous experiments were used; TextCNN and SelfAttention. The TextCNN remains almost the same. The Attention model uses a single bi-directional LSTM layer (instead of two), and all its parameters follow the self-attention mechanism described in [LFdS+17]. The attention model is considered thanks to faster learning of LSTM layers in PyTorch (observed in the initial experiments).

The SimpleLSTM and SimpleCNN models are not used in this chapter since they were outperformed by their more advanced variation (TextCNN and Attention) in every experiment of the initial testing.

At first, a TextCNN model trained on the Kaggle dataset in chapter 5 with the configuration that achieved the best results was loaded to evaluate Dataset-A 3.3.3. Even though it had 0.9991 accuracy on the Kaggle set, it could not classify this data at all, ending with 0.4991 accuracy. It shows a massive difference between the datasets and disinformation topics in them.

6.1 Word embeddings comparison

In this section, four pre-trained word embeddings are tested on Dataset-A 3.3.3 containing disinformation summaries as positive samples and disproofs as negative samples and on smaller Dataset-EN containing real news articles.

6.1.1 Pre-trained word embedding

A brief introduction of each embedding used. All of them are 300-dimensional, differences in their training and their size are summarized below.

GloVe

A count based embedding trained on 840B tokens of Common Crawl data. In total, it contains vocabulary of 2.2M words. More details in 4.2.3. Available online[1]. [PSM14a]

157

A predicted based FastText embedding, already described before in 4.2.4. Available online[2]

Wiki-News

Another pre-trained FastText embedding, available online[3]. Trained on 1 million word vectors with subword information on Wikipedia, UMBC webbase and statmt.org news dataset (16B tokens). [MGB+18]

Multilingual

A multilingual FastText embedding 4.2.4 trained on Wikipedia dataset, available online[4] for 30 languages, aligned in a single vector space. [CLR+17]

6.1.2 Embedding comparison

The input sequences (an article representation - a sequence of integers referring to a word in the word_index dictionary maintaining the order of words) are cut to 300 items (words encoded to integer values). In this configuration, the vocabulary size of the corpus is only 9 260, making it extremely fast to train. The time per epoch was around 9s for Self-Attention model and 2s for TextCNN model.

As can be seen in table 6.1, the best accuracy was surprisingly achieved with GloVe embedding even though the FastText is claimed to be current state-of-the-art [MGB+18, GBG+18, CLR+17].

Each embedding was tested in two configurations. First, it was used 'as it is' and excluded from the training. Next, it was loaded as the initial weight of the embedding layer but kept in the training process, and so it was adjusted for the given dataset.

The same experiments were repeated with Dataset-EN, its vocabulary (after the pre-processing) consist of 15 145 words (compare to 9 260 words in Dataset-A). It contains 1 038 entries only and so is smaller then Dataset-A with 10 570 entries and significantly smaller compared to the Kaggle dataset. Even though the tiny dataset, the results are similar as on Dataset-A. GloVe

[1]

[2]https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.en.300.vec.gz

[3]https://fasttext.cc/docs/en/english-vectors.html

[4]https://github.com/facebookresearch/MUSE

	Self-Attention	TextCNN
No pre-trained embedding	0.9510	0.9533
GloVe	0.9513	0.9642
GloVe + embedding training	0.9584	0.9655
157	0.9475	0.9550
157 + embedding training	0.9584	0.9579
Wiki-News	0.9490	0.9596
Wiki-News + embedding training	0.9584	0.9527
Multilingual	0.9541	0.9581
Multilingual + embedding training	0.9536	0.9544

Table 6.1: Test set accuracies of pre-trained embedding performance on the Dataset-A 3.3.3, with 300 input sequence length

achieved the highest accuracy again, especially with weights of the embedding layer included in the training.

However, promising is the excellent performance of the multilingual dataset showing that the alignment of vectors from several languages does not cause a loss on accuracy.

6.2 Limitations

This section discusses problems that occurred or might occurred during the experiments.

6.2.1 Small datasets

Unfortunately, self-obtained datasets are not large. For example, Dataset-EN has only 1 038 articles. Dataset-A, the largest one, has 10 570. Much larger datasets are typically used in Natural Language Processing. Even the Kaggle dataset used in the initial experiments consist of 20 800 articles and so is almost double the size of the Dataset-A and twenty times larger than Dataset-A.

On a small dataset, an inconsistency of training can appear. Both SelfAttention and TextCNN were trained ten times on Dataset-EN that contain only 1038 articles. The accuracy was measured after each training before the models had been initialized again for another round of training. The results can be seen in table 6.2. The inconsistency is evident, especially on the SelfAttention model performance, where the difference between the maximum and minimum accuracy achieved is more than 2%. In the case of TextCNN, the difference is only 0.87%. Moreover, it was lower when running with longer sequences, as we can see from a comparison of tables 6.2 and 6.3.

Training	SelfAttention	TextCNN
1.	0.9388	0.9825
2.	0.9417	0.9767
3.	0.9563	0.9825
4.	0.9592	0.9854
5.	0.9475	0.9854
6.	0.9446	0.9854
7.	0.9534	0.9825
8.	0.9592	0.9854
9.	0.9534	0.9854
10.	0.9475	0.9767
Mean	0.9501	0.9827

Table 6.2: Test accuracies after 10 runs on the same data showing the inconsistency of training on a small dataset. Trained with Pre-trained GloVe on Dataset-EN. The input sequences length: 1000 words.

Training	SelfAttention	TextCNN
1.	0.9359	0.9708
2.	0.9329	0.9592
3.	0.9329	0.9679
4.	0.9359	0.9621
5.	0.9359	0.9650
6.	0.9271	0.9679
7.	0.9359	0.9679
8.	0.9300	0.9650
9.	0.9417	0.9650
10.	0.9446	0.9767
Mean	0.9353	0.9668

Table 6.3: Test accuracies after 10 runs on the same data showing the inconsistency of training on a small dataset. Trained with Pre-trained GloVe on Dataset-EN. The input sequences length: 300 words.

6.2.2 The length of input sequences

The input sequence is a series of integers representing an index in the *word_index* dictionary. Each integer in the sequence corresponds to a word that had been on that place before the encoding. In NLP applications, often only a smaller sample of the processing document is used. For example, Grover [ZHR+19] use input length 1024, the Transformer model [AV17] 512 only. Even for a general text classification problem, only a cut of the text samples is typically used. For example, news classification into categories (such as sport, politics, travel, and more) can be done based on a few words only. However, the Fake News classification problem is a specific example. The information in a Fake News article could be mostly correct, and only a small part of it can be so misleading that it changes the overall meaning of the entire article. If this part would be cut off and the rest of the article

remained in the training set, it would cause noise. Especially on the automatically crawled self-obtained dataset, no one can say anymore which parts of the disinformation article put it into that group. Therefore the usage of larger sequences is beneficial in this case. An example of it can be seen from comparison of table 6.3 and 6.2. The first of them contains accuracies of models trained with input sequences length of 300 encoded words and the second sequences of 1000 encoded words. The same observation can be found in most of the experiments summarized in 5.10.

6.3 Dataset-A

This is the largest of the self-obtained datasets. However, it does not contain news articles but their summaries only, see details in section 3.3.

At first, it was trained using short input sequences of length 300 (encoded) words. As we can see in table 6.1, the best accuracy achieved was 0.9655 in combination with pre-trained GloVe embedding. Training on a longer input did not bring any improvement. When trained on sequences of 5 000 (encoded) words with the multilingual embedding, the final accuracy stayed on 0.9584.

Nevertheless, the weight of the hidden layers trained on this data could be used as initial weights for the training of a dataset that is too small to be trained efficiently. This hypothesis is tested later on Dataset-EN.

6.4 Dataset-EN

With only 1 038 articles, this is the smallest of the self-obtained datasets. It consists of disinformation from EU vs disinfo and reliable news from public service broadcasters, all of them in English (for more information about the dataset see section 3.3.4). The limitation of such a small dataset is the inconsistency of training - meaning the final results can vary even when every parameter of the model remain unchanged. As can be seen in table 6.2, the final accuracy can be different. However, considering the mean value of ten separated training, final accuracy 0.9827 exceeds the expectations. When the input length of 10 000 (encoded) words were used, the final mean value of accuracies was **0.9834**.

6.4.1 Pre-train model on Dataset-A

Later, the TextCNN model trained on the Dataset-A was used to evaluate this dataset. The accuracy was just a little over 51%, meaning the classification cannot be done based on a network trained on an article summaries only. It shows that the language of reliable and unreliable news is an essential aspect of the classification.

As we can see in table 6.4, with additional training of the pre-trained model, the accuracy has risen to 0.92 but never got close to the results of a

Model	Accuracy
Pre-trained	0.88
+ embedding training	0.92
Not pre-trained	0.96
+ embedding training	0.96

Table 6.4: A comparison of using TextCNN model pre-trained on Dataset-A vs freshly initialized model

newly initialized model trained solely on this dataset.

Note that the Dataset-A does not contain the disinformation, but their summaries and disproofs instead (more details in 3.3.3).

6.5 Models comparison

In contrast to [AV17], the SelfAttention model was outperformed by the TextCNN model not only in all of the initial experiments (from chapter 5, summarized in table 5.10) but also in most of the experiments with different embedding in this chapter (as we can see in table 6.1).

Moreover, the TextCNN can process longer input sequences which were found to be crucial on datasets tested in this project 6.2.2. TextCNN can process sequences of 10 000 words at once. In contrast, Self-Attention model has run out of GPU memory (14GB was available) already with the input of 5 000 words. Furthermore, the training of TextCNN was much faster.

Therefore, only the TextCNN model is used for the final experiments.

6.6 Handing multilingual data

In this section, a dataset containing articles in 18 different languages was used together with multilingual embedding 4.2.4 to compare which approach gives better results; split the multilingual dataset into multiple single language datasets and train different network for each of them or keep multilingual dataset and use a single network. If the multiple language approach would be successful, the network will be tested to tell if it can give a reasonable prediction even for inputs in a language that was not included in the training dataset.

6.6.1 Dataset-M1

This multilingual dataset consists of 6 132 articles. The disinformation samples are in 18 different languages; the reliable are only in English.

As we can see in table 6.5, the model was still precise in the classification.

Using five times larger length of the input brings an improvement in accuracy of about 0.3%. The model was also tested with an input length of 10 000 words. However, the improvement was negligible.

	Accuracy
Input sequences of 1 000 words	0.9867
+ embedding training	0.9906
Input sequences of 5 000 words	0.9901
+ embedding training	0.9911

Table 6.5: The classification accuracy on multilingual Dataset-M1

	Accuracy
Input sequences of 1 000 words	0.9832
+ embedding training	0.9867

Table 6.6: The classification accuracy on multilingual Dataset-M2

6.6.2 Dataset-M2

One might expect that the model trained on Dataset-M1 has learned a simple rule - if the language is not English, it is disinformation - since only the disinformation entries were in other languages. That is the reason for another test on Dataset-M2.

This multilingual dataset is similar to the previous one. It also consists of 6 132 articles. However, the positive samples are now not only in English, yet also in Russian - the most common language of disinformation in the dataset.

Comparing the results with Dataset-M1 (in tables 6.5 and 6.6), the accuracies are similar in the run without embedding training and slightly worse with the embedding layer included in the training.

6.6.3 Unknown language test

The model trained on the multilingual Dataset-M2 containing disinformation cases in several languages and reliable news in English and Russian was then evaluated on a dataset containing reliable news only, written in Ukrainian (crawled from Ukrainian version of DW). The test accuracy was only 53% confirming the hypothesis from section 6.6.2 that the network has learned only simple rule - if the language is not English (or Russian in the case of learning on Dataset-M2), it is disinformation.

Lastly, the same model (trained on Dataset-M2) was evaluated on Dataset-RU, containing both reliable and disinformation news in Russian, the test accuracy without any additional training was **0.9920** with only 18 false-positive and 8 false-negative predictions out of 6 132 articles with 50:50 ratio of reliable and disinformation samples.

6.7 Results

Based on the results in table 6.6, the performance is almost the same as it was on Dataset-M1 containing only English reliable news. However, the Russian test set might include some disinformation samples that were also included

in the multilingual dataset. Neural Networks, in general, perform better on samples that were used during training. Therefore, the results cannot be directly compared with different experiments and additional experiments would be needed to prove if the usage of one multilingual dataset is a better approach than creating various datasets - one for each language.

The results suggest that using multilingual embedding, data in different languages could be processed by a single model. With the pre-trained cross-lingual relation, it might be enough to include a disinformation case in a few languages only (or maybe just in one) for the training and, ideally, the model could be able to classify variation of the same disinformation written in a different language. However, there were not enough data to confirm this hypothesis. For a clear conclusion, a larger dataset would be needed, optimally, having 50:50 ratio of reliable vs disinformation samples and balanced language occurrence. Regrettably, the multilingual datasets used in this work are too far from the perfect state.

6.8 Czech datasets

In this section, similar experiments are done on Czech datasets 3.3.7. Some of the articles are actually in the Slovak language even though they were published on Czech websites. There can also be some terms or phrases in English. Therefore, Czech, Slovak, and English multilingual embedding were used. Since the datasets are small in size, all accuracies provided in this section are mean values over ten runs. All experiments are done with TextCNN model; its parameters remain unchanged. Except, there are more output neurons. In previous tests, there was just one. Now, there is always one for each category of a given dataset. The maximum length of articles is 3000 words.

6.8.1 Dataset-CZ-1

For a review, this dataset 3.3.7 contains 461 samples of each of the two categories - reliable and disinformation. Reliable articles are obtained from CTK and discuss the same topics as those in disinformation group (based on keywords provided within the articles).

Final accuracy on this data was **0.9874**, slightly higher than on comparable English Dataset-EN of similar size 6.4 showing that with the corresponding embedding, two very different languages can be processed by the same NN architecture. Out of 922 articles, there were only 3.8 (on average) false-negative predictions (disinformation articles mistakenly marked as reliable). The network showed excellent performance in detecting reliable articles, with no false-positive prediction.

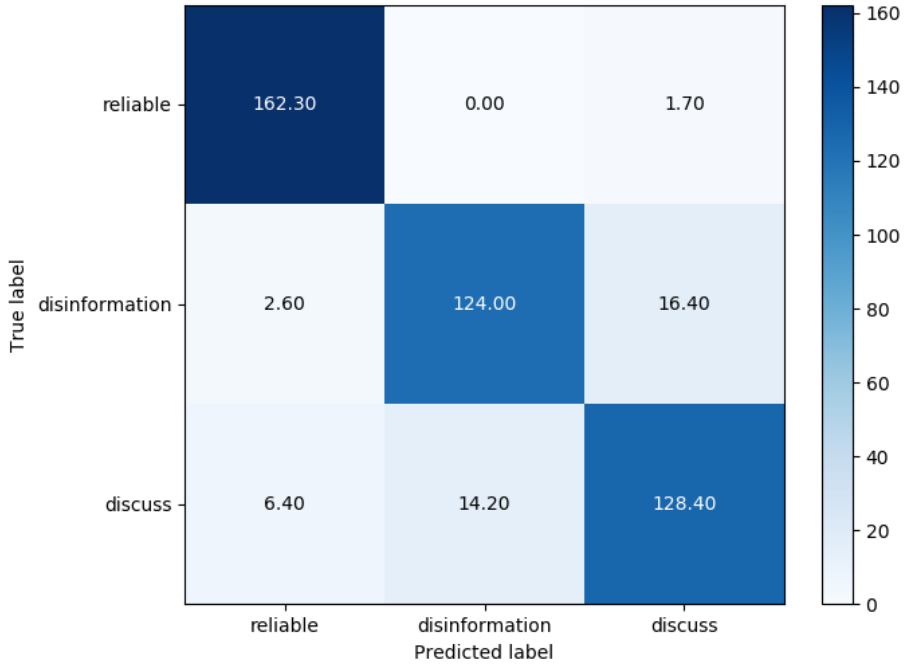


Figure 6.1: Dataset-CZ-2: confusion matrix (average values over 10 runs)

6.8.2 Dataset-CZ-2

This dataset is the same as the previous one but enriched with discuss category - articles from Parlamentni Listy and AC24, i.e. websites that often publish disinformation (see 3.3.2). However, it was not human-labelled hence it is not clear if it is disinformation, reliable news, tabloid, or another type of text.

The accuracy was lower compared to the single output experiments, only **0.9094**. Examining the results as confusion matrix (see 6.1), we can see that performance on reliable news and disinformation is the same as before. There were also a few errors in reliable and discuss groups, but most of them were in false predictions of discuss articles being disinformation and vice versa. Since the third category is a mixture of disinformation, reliable news and other articles, lower accuracy, in this case, is not a mistake but property of the dataset.

6.8.3 Dataset-CZ-3

This dataset contains reliable and disinformation articles (the same as Dataset-CZ-1) and satire articles obtained from AZ247. This classification was expected to be problematic since the satire is imitating disinformation in vocabulary and style of writing. Even the name, AZ247.cz (satire website), refers to AC24.cz (disinformation website).

However, accuracy is 0.9651. Considering there are only 1104 articles in total and satire group is smaller than the others (182 articles compare to 461

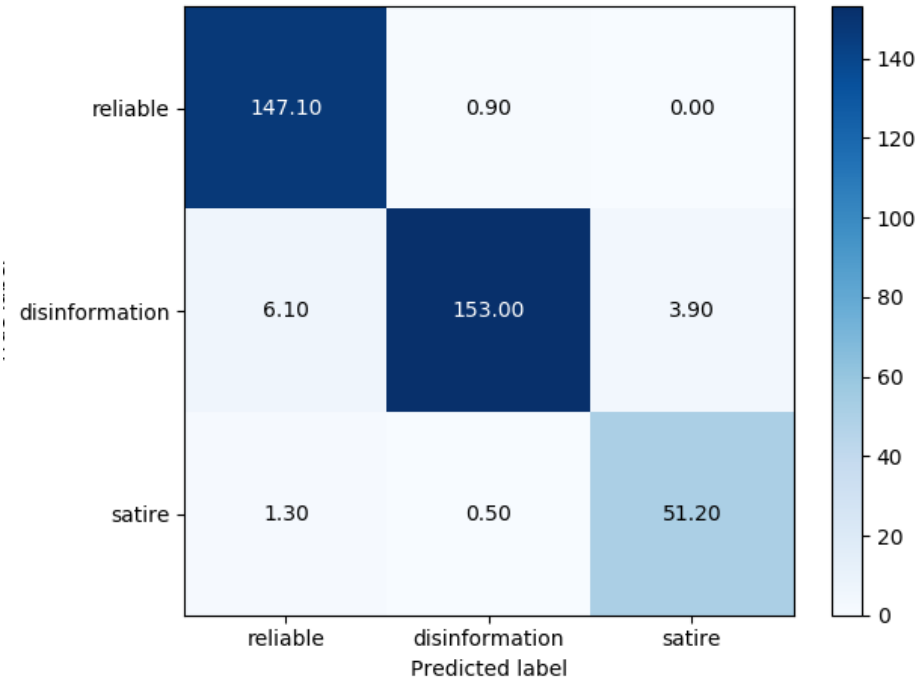


Figure 6.2: Dataset-CZ-3: confusion matrix (average values over 10 runs)

	Accuracy
Dataset-CZ-1	0.9874
Dataset-CZ-2	0.9094
Dataset-CZ-3	0.9651
Dataset-CZ-23	0.8593

Table 6.7: The classification accuracies on Czech datasets

in both reliable and disinformation), the result is surprisingly good. Details can be seen in confusion matrix 6.2.

6.8.4 Dataset-CZ-23

This dataset contains all four categories - reliable, disinformation, discuss, and satire. The accuracy was the lowest of all tests in this work, only 0.8593. However, from confusion matrix 6.3, we can see that most mistakes were made on the "unknown" discuss category. Numbers of errors between the other three categories are comparable to Dataset-CZ-1 and Dataset-CZ-3.

6.8.5 Brief analyses of Czech disinformation websites

In this section, neural networks trained on the Czech news datasets are used to evaluate articles on the main disinformation websites.

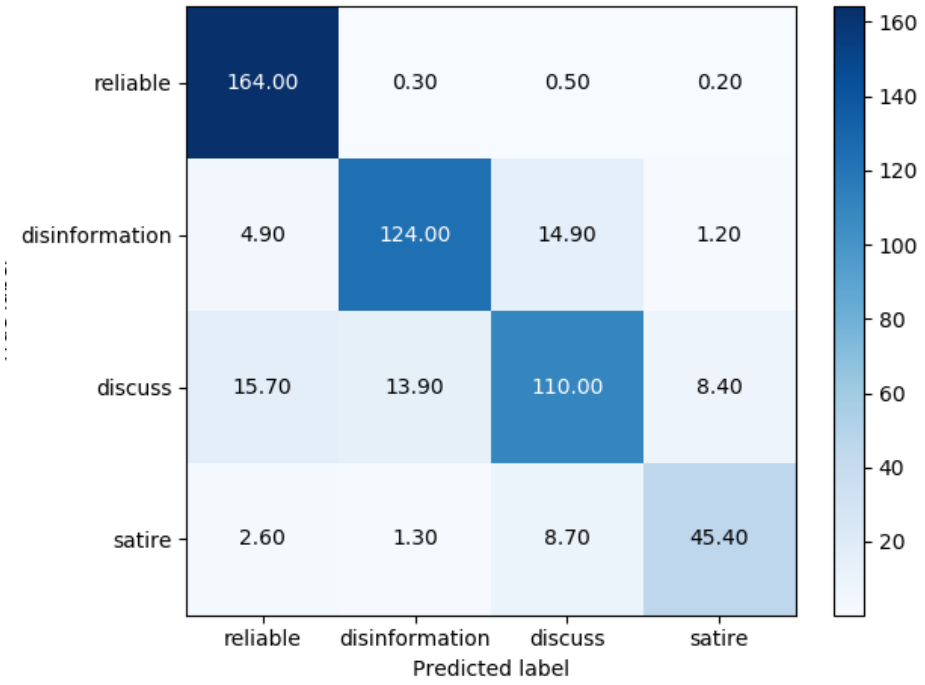


Figure 6.3: Dataset-CZ-23: confusion matrix (average values over 10 runs)

Parlamentni Listy

In total 3261 were crawled from news (zpravy) section of Parlamentni Listy [5], but only 231 randomly chosen articles were used to form 'discuss' label (together with 230 articles from AC24) of the Dataset-CZ-2 above. Trained TextCNN model then evaluated the entire corpus.

The Neural Network trained on Dataset-CZ-1 is familiar with only two categories of news - reliable and disinformations. From that point of view, 81.84% articles on Parlamentni Listy were labelled as disinformation.

On the other hand, classifying the same data by a model trained on the three-category Dataset-CZ-3 (containing satire) gives different results. The percentage of disinformation goes from 81.84% to 68.32%, and 19.93% articles were classified as satire. Considering that also the percentage of reliable articles decreased from 18.16% to 11.74%, it is evident that there is a need for separating more categories of journalism - for example, tabloid. Sadly, we do not have these in the training data, and so the network is unable to detect them.

Most importantly, NN gives a probability of input article belonging to one of the known group. The group with the highest probability is considered as the prediction. For example, on the test set of Dataset-CZ-3, the probability of the most probable label was more than 0.80 in 87% of cases, and the other two probabilities were usually close to zero. In this case, the probability

[5] https://www.parlamentnilisty.cz/zpravy

was higher than 0.80 only in 57% of articles, proving there is a need for more categories in training set to classify entire corpus of website's articles correctly.

AC24

From 7540 articles crawled from AC24 [6], only 230 were used to form Dataset-CZ-2 above. All of them were later evaluated by Text CNN trained.

Classification by model trained on Dataset-CZ-1 (reliable news and disinformation) came up very similar as on Parlamentni Listy, 0.8225 of the content was classified as disinformation. Nevertheless, by model trained on Dataset-CZ-3 (enriched by satire), only 49.19% of AC24 content was classified as disinformation and 33.53% as satire. We could see from the evaluation of Dataset-CZ-3 that the network was able to distinguish satire from disinformations. The high percentage of articles marked as satire is likely caused by the fact that the NN is not acquainted with any other type of journalism, and from the three categories, satire was the closest one.

Analyses summary

A large number of articles in these websites were classified as disinformations which follows the fact that they are known disinformation providers 3.3.2, and some of its articles occurred even in the EU vs Disinfo dataset.

It was also shown that there is a need for more categories in the training data, for example, tabloid.

6.9 Should we trust the Neural Network prediction?

Well-trained Neural Networks can give correct prediction with very high accuracy but do not provide any explanation. Due to their complexity and possibly a vast number of neurons/parameters, they are often termed as "black boxes". In some applications, it is not an issue, but in other, it is a crucial disadvantage. Such a sensitive topic as The Fake News problem is sadly in the second group.

There are some model-specific methods to obtain additional information giving insight into its prediction by inspecting model weights, but then there is also a universal one explaining prediction of any classifier, presented in "Why should I trust you" paper [RSG16].

The explainer creates a fake dataset from a single input data. In the Fake News problem, it would take a single article as the input and generates thousands of combinations of the text. Each of them omitting a different part of the text. The "black box" model is used to give predictions of every generated text combination. These predictions are used to train a new,

[6] https://ac24.cz

simpler "white-box" classifier. It is a new sub-system that can mimic the behaviour of the "black box" model.

The original article can be explained by weights of the new-trained "white-box" classifier. If the new classifier does not give good results on the generated dataset, the explanation should not be trusted.

Example

An exciting example of the prediction is explaining one of the AZ247 satire. It was correctly predicted to be a satire with probability over 99.83%. Probability of belonging to disinformation group was only 0.62%. Even though the enormous gap between the two probabilities, the LIME explanation of article belonging to disinformation (see 6.5) looks almost the same as the explanation why is it satire (see 6.4). The green colour indicates words that make the article belong to satire, red colour to disinformation. The darker the shade, the more important the word. The only significant difference of the predictions is the phrase "vesmírní ještěři" that is only highlighted in the explanation of the article belonging to satire. It might signify that the prediction was made solely on this one phrase. Therefore, all three occurrences of the phrase were removed, and the article was evaluated by the NN once again. The output probabilities remained almost unchanged, over 99.83% for satire and 0.64% for disinformation.

Although the similar explanations, the model was certain with its prediction, showing that this method cannot explain Neural Network that was design to understand words in their context. The thousands of variations of the article needed for the explanation were created randomly. The performance might be better if the variation would be done in a way that maintains context. Explaining an article sentence by sentence could lead to more informed visualization that could be used for human fact-checking.

6.10 Summary

A variety of experiments were done on the self-obtained datasets. First, four types of pre-trained embedding were tested and compare with ad-hoc learned embedding with no initial pre-trained weights of the embedding layer.

Additionally, for each embedding, two types of runs were done. First having the embedding excluded from the training, second having it included. The results vary but are mostly better when the embedding weights were adjusted for the specific need of every dataset. However, this approach can be dangerous since the embedding weights trained on enormous data are shifted base on tiny data. Moreover, the improvement observed was not significant and could be a statistical error only.

That brings us to other topic discussed in this chapter. The self-obtained datasets are, unfortunately, too small. For fair results, tests on the smallest datasets were trained ten times, and the mean value of test set accuracies was presented.

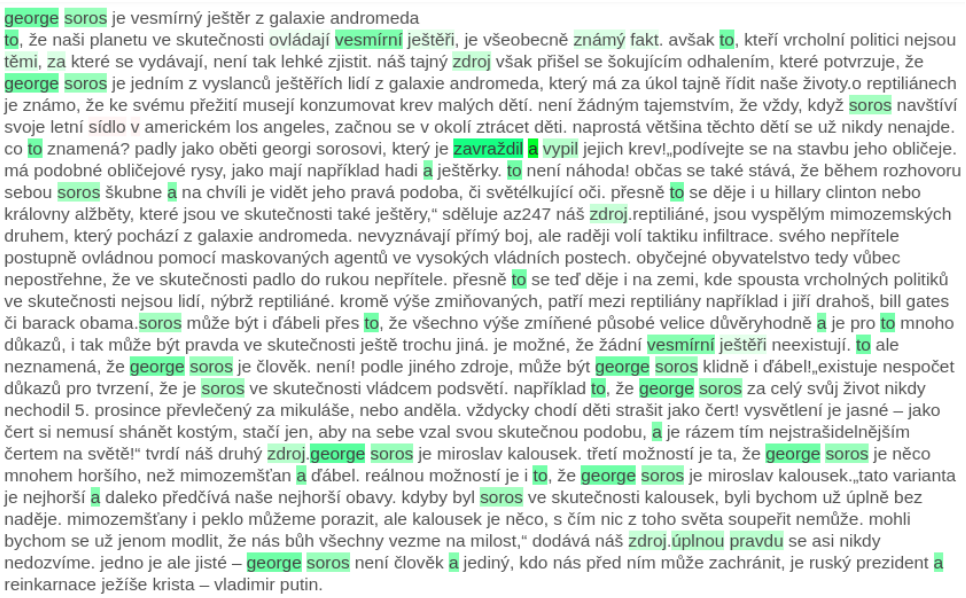


Figure 6.4: LIME explanation of a AZ247 (satire) article with highlighted words making the article being **satire**

Nevertheless, the model was able to give precise prediction even when learned on such tiny datasets. For example, Glover [ZHR+19] used 120 gigabytes of data for training. However, the vast amount of data, in that case, was needed because it is mainly a text generator — those needs larger training data in general.

Another limitation to mention is the length of input sequences (number of encoded-words in the samples after pre-processing). It was shown that in Fake News detection task, using the length of - at least - 1 000 words is crucial. A small improvement was detected when using 5 000 words. Longer sequences than 5 000 did not bring any or only limited enhancement and was not worth considering the higher computation cost. Learning long sequences is costly with LSTM layers, as discussed in 4.3, it can also suffer from long term memory loss. These problems do not occur on convolutional layers which might lead to the better performance of TextCNN model compared to SelfAttention.

Two different Neural Network architectures were compared, convolutional-based and SelfAttention (LSTM based with an attention layer). TextCNN unexpectedly outperformed the SelfAttention in almost every experiment in training time, computation cost and chiefly - in the final accuracy.

Most importantly, impressive results were observed on the multilingual datasets. In the first case (on Dataset-M1), samples of reliable news were in English only, but disinformation samples were in up to 18 different languages. In the second case (Dataset-M2), the reliable news was from 2/3 in Russian - the most common language of disinformation, at least in the dataset used. The final accuracy was unexpectedly high - scoring over 98% in accuracy in all test and even over 0.99% on Dataset-M1.

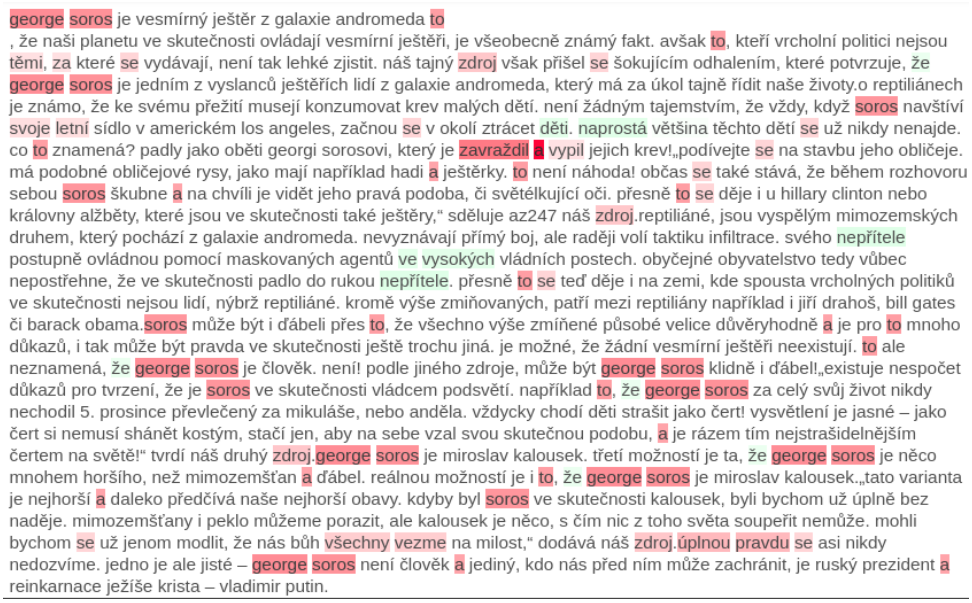


Figure 6.5: LIME explanation of a AZ247 (satire) article with highlighted words making the article being **disinformation**

The trained model was later used to evaluate Dataset-RU and reliable Ukrainian news. Despite excellent performance on the Russian dataset, it was unable to classify the Ukrainian articles. It shows that for this application, the train dataset would have to be balanced in terms of languages and positive vs negative samples in each language.

Noteworthy is also the observation of low chance of reliable news being classified as disinformation or satire.

Last but not least, better results were achieved when using the original texts instead of human-written summaries. The reliable news articles were filtered based on the keywords found in the disinformation dataset, and so both groups contained the same (or similar) topics. It suggests, there are language patterns specific for each of the categories that were lost in the human-written summaries.

Chapter 7

Outline

This chapter gives a summary of the work done. It presents third-party libraries used and gives an outline of the main observations.

This thesis introduced Fake News in the modern age and the information-rich world. Previous studies have predicted that online journalism will probably eventually turn to AI vs AI battle. AI is already used for automatic news classification into sections, keywords selection, text summarization to generate appealing headlines or summaries of the main information. On the other hand, AI-powered systems were proven to be used in malicious application as well - misleading information and disinformation repetitively occur in higher frequency before elections [fEPSoE18] a personal data can be used to psychographics models of humans, which can also be misused for a disinformation campaign. Last but not least, Fake News itself can be controllably generated based on a small piece of text, a headline, for example. The generated news is even rated to be more trustfully by humans, more details in section 2.2.6.

Later, existing Fake News datasets are presented; namely, the Kaggle 3.1 dataset that was later used for initial experiments and several self-obtained datasets based on the **East Stratcom Task Force** disinformation data and public service broadcasters data that were crawled and filtered by self-developed tools based on the news-please library [HMBG17]. The intention was to find reliable news that discusses the same topics as were found in the disinformation articles. Only articles containing at least one of the keyword extracted from the disinformation cases were used. The second important rule was picking only articles published in the same period as the disinformations.

In the next part, all of the NLP methods used during this work were explained in chapter 4, together with Neural Networks. Word embeddings and each Neural Network layer used in this work were explained in details. Along with the current state-of-the-art in the related application.

Finally, the experiments were presented. In total, more than a hundred experiments were done to find out the best practices. First, the aspect of many different pre-processing was tested, consequently with five individual Neural Network architectures. The experiments were done on the Kaggle dataset 3.1 achieving up to 99.9% accuracy.

The next part of the experiments focused on the self-obtained dataset. A

convolutional-based model performed better than the state-of-the-art [AV17] attention-based model. However, only the simplest, original version of the SelfAttention model was tested. Recent studies are using a much deeper version of the architecture with up to 48 layers and are trained on hardware with hundreds of GPU or TPU [ZHR+19]. The TextCNN model is effortless to train even on reasonable hardware. It is fast, reliable and outperforms the attention-based model in almost every experiment achieving excellent accuracy of **99.9%** on the Kaggle dataset and at least **98% accuracy** on the self-obtain dataset when trained with long input sequences and pre-trained embedding.

Lastly, it was shown that using a multilingual dataset containing news articles in several languages can be correctly classified at once thanks to embedding pre-trained word embeddings, opening another way in the Fake News detection research. However, it is likely to make more wrong prediction on languages that were covered less frequently in the training dataset.

The multilingual self-obtained datasets of real-world news articles covering the same topics in both classes - reliable and disinformation - makes this work unique. In such a case, the classification is harder since it cannot rely on keywords only (there are the same in both cases) and so is more relevant.

7.0.1 Libraries used in this project

This section contains a quick summary of the most important libraries used.

NLTK

NLTK [LB02] is one of the most used Natural Language Processing library. In this project, it was used for tokenization of the input texts. For word stemming (which is implemented specifically for several languages), namely it was used for filtering reliable news based on the keywords of the disinformation samples 3.3. Last but not least, it provides stopwords used in pre-processing.

news-please

news-please [HMBG17] (named in lowercase) is a generic news crawler and extractor combining several state-of-the-ask approaches. It was used for crawling news websites and extracting the title and body text of the articles.

Keras

Keras [Cho15] is a Neural Network library running on top of TensorFlow, Theano, or CNTK back-end offering a user-friendly API for a more advanced tool.

TensorFlow

TensorFlow [ABC+16] is the recommended backend for Keras [Cho15]. It is a powerful mathematical and Neural Network library using dataflow graphs. It also supports Keras in backward, for example, its Neural Network layers, which make the change from Keras to TensorFlow easier.

PyTorch

PyTorch [PGC+17] is another Neural Network library. With the torchtext package, it provides full tool-stack needed for NLP. It is more difficult to create training, and testing flow compare to Keras (where a single function does the training of a compiled model), but as a result, the training was faster, especially for RNN based layers and even the final accuracy can be higher as can be seen in table 5.11.

Scikit-learn

Scikit [PVG+11] is a simple machine learning library, no model in this work was designed in it but a few of its function were used, such as tokenizers or scoring function.

Matplotlib

A plotting library used to present results.

LIME

LIME is a library implementing the explainer presented in [RSG16]. It can explain any black-box classifier by creating a simple white-box classifier that is trained to mimic the behaviour of the original model. Then, the explanation is given based on the weights of the white-box classifier.

ELI15

A python package containing several tools for debugging of Machine Learning models, including LIME and Permutation Importance algorithm.

Chapter 8

Conclusion

This thesis introduces the problem of disinformation in an information-rich world. Fake News detection was addressed as a text classification problem. More than a hundred experiments were done to find an appropriate combination of pre-processing and efficient Neural Network architecture, relieving some specifics and limitation of the Fake News detection problem compared to other text classification tasks. An existing Fake News dataset was used as well as several combinations of a self-obtained dataset. The work is unique in processing news articles in numerous European languages, covering the same topics in both categories - reliable and disinformation news. The best accuracy was achieved by a convolutional based Neural Network, with up to 99.9% of correct prediction on the existing dataset, and over 98% in most experiments on the smaller self-obtained data, outperforming Self-attention mechanism. It proves that a good classification can be done even with a Neural Network architecture of a reasonable size running on restricted hardware. Better results were achieved when using the original texts instead of human-written summaries (even though the second option was trained on a larger dataset). The results suggest, there are language patterns distinctive for each of the two categories that were lost in the human-written summaries.

The Neural Network classifiers presented in this thesis could be implemented in more complex systems that would not focus on the text only, but also the images, videos, sources and even the comment section of the news websites. Audio and video materials can be transcripted into text by already existing (usually AI-powered) software with astonishing accuracy; thus, the text classification approach could be applied to them as well.

The results suggest that using multilingual embedding, data in different languages could be processed by a single model. Neural Network, in general, performs better on data similar to the training data. With the pre-trained cross-lingual relation, it might be enough to include a disinformation case in a few languages only (or maybe just in one) for the training and, ideally, the model could be able to classify variation of the same disinformation written in a different language. However, there were not enough data to confirm this hypothesis. For a definite conclusion, a larger dataset would be needed, optimally, having 50:50 ratio of reliable vs disinformation samples and balanced language occurrence. Regrettably, the multilingual datasets

used in this work are too far from the perfect state.

A real-word Fake News detection systems based on the work discussed in this theses could be used by the EU and its member states agencies to detect harmful content as well as in private sectors. Leading internet companies provide News aggregators (or Feed readers) serving content from a vast number of websites, personalised for each user. Reliability of a source and a particular article should be taken into consideration by the personalisation algorithms. Someone can reject this idea with a warning of censorship. However, studies have shown that many disinformations are ignored by the public(see 1.2), hence filtering disinformation articles from the feeds can improve the personalised experience. If providing reliable, relevant content would be flagged as censorship, then - by the same logic - any search engine or even encyclopedias should be flagged the same as it also provides only relevant results to a searched term, which is clearly a nonsense.

I would also like to express a recommendation for government agencies and non-profit organisations studying the effect of disinformation and mainly fact-checking teams to cooperate on a cross-country level as it has been shown that a single Neural Network could process even multilingual data. There is a need to create a large dataset to train the classifiers correctly. Lastly, it has found to be essential to store the original text instead of its summaries and links only, because the links might not be active anymore, or it can be redirected to different content, and the article itself can be edited. Also, the Neural Network classifier performs better on the original texts compare to its human-written summaries. Moreover, even using state-of-the-art technologies to obtain the article from a link, there can be still some errors during the parsing, causing a noise in the final data leading to mistakes in learning and wrong classification.

Appendix A

Grover example

January 14, 2019 - Denis Rehacek

The Artificial Intelligence (AI) has become a major investment trend in 2018, with major tech companies pouring millions into tech startups focused on combatting fake news and trolling. As the Fake News Season begins, the startups continue to create AI that improves search and filtering of fake stories and computational suppression of fake news, where automated tools automatically censor and evaluate content.

While many researchers are still looking at the problems involved in this field, two startups on these pages are developing the underlying technologies. Both companies are looking at the general problem of inauthentic content and offer competing products.

Antisense

In 2016, the startup Recorded Future hired researchers to investigate and create the first AI that could parse and analyse the accuracy of on-the-ground reporting. The goal was to create an algorithm that would analyse sources and use the knowledge to speed up more efficient accuracy ranking. This algorithm is called AIX, short for AI reporting platform. The end goal is to improve brand trust, and this is also the end goal of the data journalists use to justify their work.

Less than a year after the team's heady idea, Recorded Future now has AI that can mine Twitter and analyse its content. It does this by testing and scanning what users and experts are saying about Donald Trump and other politicians. It can recognise the words, quotes, and quotes attached to what people are writing, analysing the meaning behind them, and searching for patterns in the comments that reveal the trend, topic, or sentiment of the story being promoted.

Similarly, the company's tech can analyse the language used to promote fake news. In July, the company revealed its AI's ability to identify fake news on Twitter in a single tweet.

The AI's ability to be able to scan through tweets is also a big breakthrough. Previously, Twitter has been very restrictive in how it uses data. It has licensed it to companies who are experts in optimising it to find trends and quickly display this information for companies like Spredfast, Uber, Waze and others.

The reason Twitter wants to extract as much value as possible out of its

data is that it, along with others, have been unhappy with Facebook and Google over the past few years as they run the platforms and create bots that inject their content into social media feeds around the world. By analysing the social media content, AI companies are finding more accurate ways to sift through this content.

Reputation Defender

Reputation Defender, which also operates out of Israel, just released AI-based authentication. This AI can determine if users are fraudulently using news updates on the company's platform and can prevent them from publishing fresh news stories that could be fake. The company claims that its algorithm can scan more than 100,000 links per second, tens of thousands of hours of video, and is incredibly fast.

The AI scans all links, including those in 100,000 headlines, to identify if they are fraudulent. Once it has identified the material as such, the human algorithm assesses whether or not it has tried to make a traffic link through Autocorrect, picked up a scammy link on Hacker News, and even follows a link to a site where stolen or fraudulently generated IP addresses have been spotted.

If the company spot the problematic content, the AI can immediately block it. It can also scour Twitter to look for news stories that were spammy because the users have used bots and key words.

The company, which just raised $5 million last month, claims that its tech uses an approach to AI that is very different from what others are using. "It works by applying machine learning inference to structured data in a much more real-time way", Gil Duvdevani, the co-founder and chief scientist of the company, told the Times of Israel.

These technologies allow journalists to better track down and curb hate speech and fake news, while still maintaining full faith in the ability of humans to scrutinise content.

Neither AI can stop fake news completely. However, AI does have a role to play in improving the quality of user reporting and brand trust. Not only does this result in being faster, but the technology can also lead to more robust protection against these problems. More progress is likely to be made in the coming years, as AI and machine learning are becoming more advanced, and data journalism becomes more complex. AI is changing everything — except our relationships with each other.

Click HERE to read more

Appendix B

Attachments

- src.zip - source codes
- thesis.pdf - the thesis itself in pdf
- thesis.zip - the thesis in Latex
- assignment.pdf - official ssignment of the thesis

Appendix C

Bibliography

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *Tensorflow: A system for large-scale machine learning*, Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (Berkeley, CA, USA), OSDI'16, USENIX Association, 2016, pp. 265–283.

[AG17] Hunt Allcott and Matthew Gentzkow, *Social media and fake news in the 2016 election*, `http://www.aeaweb.org/articles?id=10.1257/jep.31.2.211`, May 2017, Accessed on 2019-01-01.

[AV17] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer, *Attention is all you need.*

[BDK14] Marco Baroni, Georgiana Dinu, and Germán Kruszewski, *Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors.*

[BGJM16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, *Enriching word vectors with subword information*, CoRR **abs/1607.04606** (2016).

[Cad18] Carole Cadwalladr, *The cambridge analytica files*, The Guardian (2018).

[Cho15] François Chollet, *keras*, `https://github.com/fchollet/keras`, 2015.

[cit] *Revealed: Trump's election consultants filmed saying they use bribes and sex workers to entrap politicians.*

[cit19] *Infobanka ctk.*

[CLR+17] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou, *Word translation without parallel data*, arXiv preprint arXiv:1710.04087 (2017).

[fEPSoE17] Andrea Renda (CEPS Centre for European Policy Studies and College of Europe), *Media consumption forecasts 2018*.

[fEPSoE18] ———, *The legal framework to address "fake news": possible policy actions at the eu level*.

[GBEF+18] Jennifer Golbeck, Jennine B. Everett, Waleed Falak, Carl Gieringer, Jack Graney, Kelly M. Hoffman, Lindsay Huth, Zhenya Ma, Mayanka Jha, Misbah Khan, Varsha Kori, Matthew Mauriello, Elo Lewis, George Mirano, William T. Mohn IV, Sean Mussenden, Tammie M. Nelson, Sean Mcwillie, Akshat Pant, and Paul Cheakalos, *Fake news vs satire: A dataset and analysis*, 05 2018, pp. 17–21.

[GBG+18] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov, *Learning word vectors for 157 languages*, CoRR **abs/1802.06893** (2018).

[GD18] David Güera and Edward J Delp, *Deepfake video detection using recurrent neural networks*, 1–6.

[HMBG17] Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp, *news-please: A generic news crawler and extractor*, Proceedings of the 15th International Symposium of Information Science (Maria Gaede, Violeta Trkulja, and Vivien Petra, eds.), March 2017, pp. 218–223.

[HSS+18] Andreas Hanselowski, Avinesh P. V. S., Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych, *A retrospective analysis of the fake news challenge stance detection task*, CoRR **abs/1806.05180** (2018).

[Jan18] Martin Janda, *Information laundering: fake news websites in czech context*.

[ker19] *Keras lstm attention glove840b,lb 0.043*, 2019.

[KGS17] Jan Koutník Bas R. Steunebrink Klaus Greff, Rupesh K. Srivastava and Jürgen Schmidhuber, *Lstm: A search space odyssey*, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS **28** (2017).

[Kim14] Yoon Kim, *Convolutional neural networks for sentence classification*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Doha, Qatar), Association for Computational Linguistics, October 2014, pp. 1746–1751.

[LB02] Edward Loper and Steven Bird, *Nltk: The natural language toolkit*, Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1 (Stroudsburg, PA, USA), ETMTNLP '02, Association for Computational Linguistics, 2002, pp. 63–70.

[Lee12] Eun-Ju Lee, *That's not the way it is: How user-generated comments on the news affect perceived media bias*, Journal of Computer-Mediated Communication **18** (2012), no. 1, 32–45.

[LFdS+17] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio, *A structured self-attentive sentence embedding*, CoRR **abs/1703.03130** (2017).

[MGB+18] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin, *Advances in pre-training distributed word representations*, Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, *Distributed representations of words and phrases and their compositionality*, CoRR **abs/1310.4546** (2013).

[New18] Facebook Newsroom, *An update on our plans to restrict data access on facebook.*

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, *Automatic differentiation in pytorch*, NIPS-W, 2017.

[PSM14a] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, *Glove: Global vectors for word representation*, Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.

[PSM14b] ———, *Glove: Global vectors for word representation*, 2014, Accessed on 2019-05-02, pp. 1532–1543.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[QGSL18] Feng Qian, Chengyue Gong, Karishma Sharma, and Yan Liu, *Neural user response generator: Fake news detection with collective user intelligence*, 3834–3840.

[RD17] Chris Proctor Richard Davis, *Fake news, real consequences: Recruiting neural networks for the fight against fake news.*

[RK16] Ondrej Bajgar Jan Kleindienst Rudolf Kadlec, Martin Schmid, *Text understanding with the attention sum reader network.*

[RN09] Stuart Russell and Peter Norvig, *Artificial intelligence: A modern approach*, 3rd ed., Prentice Hall Press, Upper Saddle River, NJ, USA, 2009.

[RSG16] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin, *"why should I trust you?": Explaining the predictions of any classifier*, CoRR **abs/1602.04938** (2016).

[RSL17] Natali Ruchansky, Sungyong Seo, and Yan Liu, *Csi: A hybrid deep model for fake news detection.*

[SH97] Jürgen Schmidhuber Sepp Hochreiter, *Long short-term memory.*

[SHS01] Paolo Frasconi Sepp Hochreiter, Yoshua Bengio and Jürgen Schmidhuber, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.*

[Tim16] New York Times, *As fake news spreads lies, more readers shrug at the truth*, `https://www.nytimes.com/2016/12/06/us/fake-news-partisan-republican-democrat.html`, Dec 2016, Accessed on 2019-01-01.

[usc18] *Menendez questions government contract to cambridge analytica parent company and raises ethical and legal concerns about its work in foreign countries*, Apr 2018.

[Wan17] William Yang Wang, *"liar, liar pants on fire": A new benchmark dataset for fake news detection*, CoRR **abs/1705.00648** (2017).

[ZCL15] Charles Elkan Zachary C. Lipton, John Berkowitz, *A critical review of recurrent neural networks for sequence learning*, Cambodian Mathematical Transactions **5** (2015).

[ZHR+19] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi, *Defending against neural fake news*, arXiv preprint arXiv:1905.12616 (2019).

[ZT18] Jun Xie Yidong Chen Xiaodong Shi1 Zhixing Tan, Mingxuan Wang, *Deep semantic role labeling with self-attention.*

[ZZXW18] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu, *Object detection with deep learning: A review*, CoRR **abs/1807.05511** (2018).