university of groningen

faculty of science and engineering

MASTER THESIS

Near Real-Time Detection of Misinformation on Online Social Networks

Author:
Lennart VAN DE GUCHTE

Internal Supervisor:
dr. Jennifer SPENADER

External Supervisors:
prof. dr. Stephan RAAIJMAKERS
dr. Erik MEEUWISSEN

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in Artificial Intelligence

in the

Department of Artificial Intelligence
Faculty of Science and Engineering

July 6, 2020

"On the mountains of truth you can never climb in vain: either you will reach a point higher up today, or you will be training your powers so that you will be able to climb higher tomorrow."

Friedrich Nietzsche

Abstract

The massive usage of online social networks has amplified the negative effects that misinformation has on society. To counter misinformation, fact-checkers try to verify and debunk news stories. However, due to the speed at which misinformation is disseminated, manual fact-checking often comes too late. Moreover, misinformation influences people's thoughts, beliefs, and opinions even after it has been corrected. Therefore, to prevent misinformation from being harmful, it is crucial to detect misinformation in real-time, when it begins to spread.

Automatic approaches have been proposed that utilize machine learning techniques combined with a variety of features that discriminate misinformation from trusted information. These approaches rely on micro-blog posts that disseminate misinformation on online social networks, such as diffusion patterns, linguistic cues, or user characteristics. The more micro-blog posts become available, the easier it gets to detect misinformation. This makes misinformation detection a time-sensitive task, in which a trade-off is needed between efficiency and effectiveness.

In this thesis, we focus on the early detection of misinformation on online social networks, and evaluate the effectiveness of different features. We do this by extracting a comprehensive set of network and linguistic features and propose a deep learning model that combines both feature types. Moreover, we combine the network and linguistic features with temporal information about diffusion patterns and evaluate their performances with respect to the earliness of detection.

Because shared misinformation datasets are lacking a method for constructing large, topic-dependent Twitter datasets has been proposed and used to create a novel political misinformation dataset. Experiments on this dataset demonstrate that our proposed method detects misinformation with an accuracy of 93% in near real-time. Moreover, we find linguistic features outperforming network features and provide a deep insight into which individual features are effective to detect misinformation, taking into account the time needed for detection.

Acknowledgements

First, I would like to express my gratitude towards my supervisors at TNO, Stephan Raaijmakers and Erik Meeuwissen, who helped me tremendously in giving this research direction. The brainstorm sessions we had were great and ensured that ideas developed into scientific research. Secondly, I want to thank Jennifer Spenader whose support and enthusiasm fueled me with energy, even when the road got rough. Collectively, I'm grateful that you were my supervisors, the insights you gave were valuable and helped me to publish this work.

Furthermore, I want to thank my family and friends for the necessary distraction and joy that you provided when writing this thesis. A special thanks to my parents who have always supported me during my studies. Finally, I want to thank my girlfriend, Denice, whose support has been invaluable to me.

Contents

List of Figures

List of Tables

List of Abbreviations

OSN	**O**nline **S**ocial **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
API	**A**pplication **P**rogramming **I**nterface
SVM	**S**upport **V**ector **M**achine
RF	**R**andom **F**orest
MLP	**M**ulti **L**ayer **P**erceptron
NLP	**N**atural **L**anguage **P**rocessing
URL	**U**niform **R**esource **L**ocator
LSTM	**L**ong **S**hort-**T**erm **M**emory

Chapter 1

Introduction

1.1 Introduction

As the printing press increased the spread of information, the internet has increased the speed and scale at which information can spread. Nowadays, one-third of the world population uses online social networks (OSNs), like Twitter or Facebook, to obtain and share information (Ortiz-Ospina, 2019). On these networks news stories can be broadcast to large numbers of people within minutes after being published. Therefore OSNs have become a powerful tool in the current media landscape. However, unlike traditional media, everybody can post and share news stories without having to comply with some principles regarding quality and truthfulness. As a result, OSNs are being utilised to disseminate misinformation such as, for example, propaganda, hoaxes, conspiracy theories, and rumors.

Although misinformation has been around for a long time the speed and quantity at which misinformation is currently disseminated amplifies the negative effects that misinformation has on society. Online misinformation has become a weapon to manipulate the public opinion at scale and recent studies have shown what harmful consequences this can have politically, economically, and in various other ways. The use of misinformation to influence people's opinion was brought to light after the U.S. presidential election and the Brexit in 2016. It is now widely believed that misinformation that circulated on OSNs during these elections have influenced their outcomes (Allcott and Gentzkow, 2017; Howard and Kollanyi, 2016). It was also this year that Oxford Dictionaries declared 'post-truth' as international word of the year[1], indicating the disappearance of shared objective standards for truth. Since then the general consensus is that online misinformation is a serious threat to society.

To counter misinformation fact-checkers verify news stories and correct inaccurate or false information. However, manual fact-checking is not resistant to the quantity and speed at which deceptive information is currently propagated. Further, researchers have concluded that correcting misinformation after dissemination is too late to be fully effective e.g. (Lewandowsky et al., 2017), due in part to the "continued-influence effect" (Lewandowsky et al., 2012): damage caused by exposure to misinformation is hard to undo. This is why detecting and verifying misinformation in real-time, as it begins to spread, is crucial.

Previous efforts to automate misinformation detection have utilized machine learning techniques and combined this with features extracted from article content or

[1] https://languages.oup.com/word-of-the-year/2016/

social context information, such as micro-blog posts, diffusion behaviour, and user characteristics (Shu et al., 2017). Although various studies proved that these features are effective in detecting misinformation after dissemination, only a few studies applied these features to early detection (Pierri and Ceri, 2019).

In this thesis, we want to investigate what features are optimal for automatic misinformation detection on OSNs with respect to the earliness of detection.[2] To do this we focus on micro-blog posts that broadcast hyperlinks to news articles, either misinformation or not. We ignore the actual context of these hyperlinks but focus on the linguistic, network, and temporal properties of these posts. This approach is motivated by the fact that it appears to be difficult and non-trivial to use only the text of an article for detection (Shu et al., 2017).

Another issue within this research domain is the lack of ground-truth data to evaluate detection methods. This is due to a number of reasons that make creating a benchmark misinformation dataset very difficult. One of these reasons is that misinformation comes in all shapes and sizes, which makes it hard, or even impossible, to let the dataset include all types of misinformation. This has as consequence that the performance of a feature highly depends on the data that is used during evaluation. In this thesis our aim is to find the optimal features for misinformation detection in general, however, because misinformation detection is topic-dependent and over-represented in political news (Vosoughi et al., 2018), we decided to evaluate these features on politically related misinformation articles.

1.2 Research Questions

In essence, the main objective of this thesis is to find out what linguistic, network, and temporal features, extracted from OSNs, are optimal for the early detection of misinformation. Additionally, we identify the problems with current datasets and propose a method to create a novel dataset to evaluate our methods. To support the main objective we formulated the following research question:

...

Which linguistic, network and temporal features can be used effectively for the early detection of political related misinformation on online social networks?

...

To answer the main research question we decomposed this question into the following sub-questions:

- **SQ1.** How can we construct a dataset for political misinformation detection that includes linguistic, network and temporal information about news articles broadcast on an online social network?
- **SQ2.** What linguistic, network, and, temporal features discriminate misinformation from trusted information and how can we best extract them?
- **SQ3.** What classification model is able to incorporate all feature spaces while being able to evaluate each feature space independently from one and other?

[2]Part of this work has been published in the conference proceedings of the 2nd Multidisciplinary International Symposium on Disinformation in Open Online Media (Van de Guchte et al., 2020).

- **SQ4.** Using the classification model from **SQ3** and the identified relevant features from **SQ2** to detect misinformation, what is the relative contribution of these feature spaces?
- **SQ5.** How do the different feature spaces perform with respect to the earliness of detection?

These sub-questions will be answered in the upcoming chapters. For each question, we refer to this question in the introduction of the corresponding chapter and answer it in the summary section of that chapter. Finally, we answer the main research question in the conclusion of this thesis.

1.3 Thesis Layout

The rest of this thesis is organized as follows. The next chapter presents a detailed background on misinformation detection that relates to this research. In Chapter 3 we describe how we constructed a novel dataset for misinformation detection. Chapter 4 and 5 explain our approach which include the features and algorithms that are being evaluated. Chapter 6 describes the experiments we conducted and discusses the results. Finally, in Chapter 7 we draw conclusions and present ideas for future work.

Chapter 2

Background

In recent years, concerns about the influence of misinformation on society have led to an increasing interest from academia. Both social and computer scientists have studied the influence of online misinformation and how to counteract this phenomenon. Due to the versatility of the problem, different approaches have been suggested. In this chapter, we describe what previous research has been done, which challenges remain, and how our research fits in.

2.1 Terminology

The burst in academic research on deceptive online information has resulted in inconsistent and overlapping terminology. Examples of terminology used for deceptive online information in previous research are: *fake news*, *false news*, *disinformation*, *propaganda*, *hoaxes*, *conspiracy theories*, or *rumors*. The first three terms have been used interchangeably while the others define more specific types of deceptive information. Furthermore, there is some inconsistency about the definition of these terms which has led to confusion (Weeks and Gil de Zúñiga, 2019). Especially the term "fake news" has been irredeemably polarized in our current political and media climate and should therefore be avoided in academic writing (Vosoughi et al., 2018). We conform to this suggestion and use misinformation as an umbrella term to address all false or inaccurate information that is intentionally or unintentionally being disseminated. This general definition of the term misinformation makes it appropriate for our research as we investigate the detection of misinformation in general, and not the intent of the writer (e.g. disinformation), or a specific type of misinformation (e.g. conspiracy theories). Because previous research used different terminology we will refer to them with the term misinformation only when suitable, and use more specific terms if necessary.

2.2 The Influence of Misinformation

Misinformation is influencing peoples beliefs and opinions, and this can manifest itself in various ways. For example, the shootout that took place in a pizzeria as a result of online claims that the pizzeria for high-ranking officials of the Democratic Party was using human trafficking and child sexual abuse (Shu et al., 2017). Another example occurred in 2016, a tweet claimed that president Obama got injured by an explosion (Rapoza, 2017). This resulted in a finance stock crisis and wiped out

130 billion in stock value in only minutes after the tweet was posted. More recently, misinformation regarding the COVID-19 pandemic showed how false beliefs lead to health risks. For example, in Iran an article in which people proclaimed that drinking methanol would cure COVID-19 resulted in approximately 500 deaths (Delirrad and Mohammadi, 2020). These examples show what direct consequences misinformation can have. However, the majority of misinformation does not have direct consequences but influences people and societies in more profound ways on the long run. Although is difficult to prove what precise effect misinformation has, or will have, many researchers believe that misinformation leads to the disappearance of shared objective standards for truth. As a result, there has been a significant drop in trust in major institutions, organizations, leaders and many sources of information (*Edelman Trust Barometer* 2020).

To minimize the harm caused by misinformation we need to know how misinformation is affecting humans on an individual level. Due to the variety in types of misinformation, and because humans respond differently to misinformation, it is difficult to generalize. Though, recent psychological studies found some fundamental evidence that misinformation leads to distrust. It was found that the presence of misinformation results in people disbelieving related facts (Van der Linden et al., 2017; McCright et al., 2016) and, when framed as conspiracy theories, misinformation causes people to be less likely to accept official information from, for example, governments and institutions (Einstein and Glick, 2015; Jolley and Douglas, 2014). This indicates that misinformation influences humans in more profound ways than its initial influence, which is misinforming people. In an attempt to solve the growing distrust researchers have investigated how people respond to the correction of misinformation. This means that misinformation that is assumed to be true is later corrected. Unfortunately, most studies concluded that the correction of misinformation is rarely fully effective (Lewandowsky et al., 2017). In literature this phenomenon is named the "continued-influence effect" (Lewandowsky et al., 2012) and means that the damage caused by exposure to misinformation is hard to undo. Hence, the only way to prevent humans from being harmed by misinformation is to prevent humans from being exposed to misinformation. For this reason, we investigate in this study how misinformation can be detected immediately after it has been broadcast on an OSN.

2.3 Characteristics of Misinformation

Key platforms in the spread of misinformation are online social networks (OSNs) such as Twitter and Facebook. These platforms are used to amplify the spread of misinformation. Because of the major impact OSNs have in the modern media landscape the consensus is that they should actively combat misinformation. Therefore, OSNs have been investigated widely to find out what distinguishes the spread of misinformation from truthful information. The next sections describe some major findings about what characterizes the spread of misinformation and how these characteristics are used to detect misinformation.

2.3.1 Information Diffusion Patterns

A study by Vosoughi et al. (2018) focused on information diffusion patterns on Twitter. In this work, the spread of true and false news was studied by analyzing more than 4.5 million tweets which were posted between 2006 and 2017. To classify news as false they used six independent fact-checking organizations. This extremely large study drew a remarkable conclusion. In all categories of news (politics, urban legends, business, science & technology, terrorism & war, entertainment, and natural disasters) false news diffused significantly farther, faster, deeper, and more broadly than true news. Between categories, this effect was more pronounced for political news than for all other categories. This indicates that diffusion patterns of news stories are useful to distinguish misinformation from truthful information.

Because the information diffusion patterns in online social networks can be complex different features are used to represent these patterns. In early research on the detection of political abuse on Twitter using information diffusion networks, it was already found that simple network features were successful in identifying truthful information (Ratkiewicz et al., 2011). Examples of features they used are the number of nodes/edges, maximum in/out-degree, and mean size of connected components. Combining these features with a machine learning classifier resulted in a classification accuracy of 95.6 % on a test dataset. Nevertheless, this research dates back to 2011 and since then OSNs have changed in various ways, including the spread of information, the size of the network, and more. We therefore want to investigate if these network-related features are useful for misinformation detection.

2.3.2 Social Bots

The information diffusion patterns we described above are products of the users who spread this information. Therefore researchers have been identifying these users as this may be useful to discriminate misinformation from truthful information. Though real people also spread misinformation (intentionally or unintentionally) it was found that automatic spreaders, called *social bots*, play a major role in the dissemination of misinformation on OSNs (Ferrara et al., 2016b). Social bots try to mimic human behaviour to automatically produce content and interact with other humans. Due to the rapid development of artificial intelligence new social bots are being developed in very advanced ways.

A review paper by Ferrara et al. (Ferrara et al., 2016b) shows what an enormous influence social bots have on OSNs. By using automated ways to spread misinformation they affect stock markets (Hwang et al., 2012), promote terrorist propaganda (Berger and Morgan, 2015), produce extremist content (Ferrara et al., 2016a), and influence political campaigns (Bessi and Ferrara, 2016). Since social bots have been deployed extensively during the U.S. elections of 2016 and the Brexit, the behaviour of these 'political' bots is well studied. These studies revealed that social bots are mainly used to further polarize political discussions and enhance the spread of misinformation (Bessi and Ferrara, 2016; Bastos and Mercea, 2019). This is done using so-called *botnets* which are groups of bots that work together. For example, Shao et al. (2018b) analyzed 14 million tweets that were posted during and after the U.S. elections of 2016. What was found is that botnets applied strategies to amplify the spread of misinformation in an early stage, to increase the chance that an article goes "viral." Besides, they found that social bots often mention super-spreaders (users with many followers) in their posts to reach a big audience.

To find out which users are social bots various detection methods have been investigated. These can be categorized as *structured-based*, *crowdsourcing-based*, and *machine learning-based*. From these methods the most effective and popular one is *machine learning-based* detection (Karataş and Şahin, 2017). Though, with the development of more sophisticated social bots, the development of bot detection algorithms remains an ongoing research direction.

2.3.3 Linguistic Features

Previous research on linguistic features has been done using article content or social media content (e.g. tweets). In this work, we are solely interested in social media content and therefore we describe here which linguistic characteristics are related to social media posts that are related to misinformation.

In section 2.3.1, we already mentioned a study that analyzed the spread of false and true news using more than 4.5 million tweets (Vosoughi et al., 2018). This study also compared the emotional content of those tweets between true and false stories using eight different categories for emotion. These categories are *surprise*, *disgust*, *fear*, *anger*, *sadness*, *anticipation*, *joy*, and *trust*. For all tweets, an emotion score for each category was given. This was done by using a lexicon of 140,000 English words and 32,000 Twitter hashtags that all were related to one of these eight emotions. It was found that tweets from false news expressed greater *surprise* and *disgust* in their text, whereas tweets related to true news expressed greater *trust*, *anticipation*, *sadness*, and *joy*. These results are evidence that linguistic features, that indicate emotional responses, seem to be promising features to distinguish misinformation from truthful information. In this study, we will therefore investigate the discriminative power of such features for detecting misinformation.

To capture the linguistic features that distinguish misinformation from truthful information various natural language processing (NLP) techniques have been utilized. These can be categorized as handcrafted features or statistical features. Handcrafted features make use of regular expressions, lexicons, or other methods to extract linguistic information. In an early study on the credibility of information on Twitter (Castillo et al., 2011) a comprehensive set of handcrafted features was already found to detect tweets that contain misinformation with quite some precision, though, outperformed by a subset of propagation features. This set of linguistic features included sentiment words, hashtags, emoticons, orthography, and topic-related features. After this more research investigated the use of handcrafted features for misinformation detection. Recently, a study evaluated a set of linguistic features that were found to be successful in previous studies (Reis et al., 2019). The linguistic features consisted of syntax, lexical, psycholinguistic, semantic, and subjectivity features extracted from the text of a news article. It was found that all these feature groups have some discriminative power when distinguishing misinformation from truthful information. Therefore we want to find out if such features are also good for misinformation detection when extracted from tweets.

Since the characteristics of misinformation are still not yet fully understood, and depend on the type of misinformation, it is difficult and time-consuming to create hand-crafted features that work well in general (Ruchansky et al., 2017). To avoid this limitation statistical linguistic features are used that try to capture the semantics of a text by a numerical vector representation. These features are named embeddings and can represent a word, sentence, paragraph, or entire document. The most promising study that utilised embeddings to detect misinformation is by Volkova et al. (2017). In this work, the Doc2Vec algorithm (Le and Mikolov, 2014) was trained on a Twitter corpus that consisted 130,000 of suspicious and verified tweets. This algorithm generated 200-dimensional embeddings for every tweet in the corpus. Classifying these embeddings as suspicious or verified with logistic regression performed poorly (65% accuracy), however, when a recurrent or convolutional neural network was used as classifier the accuracy improved by more than 10 percent (78% and 76% accuracy, respectively). Moreover, when a comprehensive set of handcrafted linguistic features was added a accuracy of 93% was reached. This study reveals how handcrafted and statistical linguistic can supplement each other for the detection of misinformation which is why in this study we will also evaluate both types of features.

2.4 Detection Algorithms

As described in the previous section various characteristics are useful to identify misinformation, these include diffusion patterns, user characteristics, and linguistic cues. To use these characteristics for misinformation detection the task is formulated as a supervised binary classification problem. To solve this problem features are extracted and fed to machine learning classifiers. Depending on the type of features, different classifiers are being evaluated. Reis et al. (2019), for example, used a common approach for misinformation detection in which a set of features, represented by numerical values, are fed to a supervised machine learning classifier. The classifiers that have been evaluated are K-Nearest Neighbor, Naive Bayes, Random Forests, Support Vector Machine, and XGBoost. It was found that Random Forest and XGBoost performed best, however, this entirely depends on the features and data that have been used.

In addition to traditional machine learning classifiers, deep learning techniques are being utilized for misinformation detection. A reason for this is that deep learning methods can create latent feature spaces that cannot be captured using handcrafted features. Especially recurrent neural networks (RNNs) have been investigated to capture the temporal dynamics of the spread of misinformation. Ma et al. (2016) were the first to propose an RNN for the detection of misinformation on OSNs. This was done by modeling the spread of misinformation as variable-length time series data in which each time step contained multiple micro-blog posts that were represented by a linguistic feature called the term frequency–inverse document frequency (tf-idf). The RNN could effectively model the spread of rumors and outperformed other techniques that used handcrafted features.

Recently, some studies focused on the early detection of misinformation (Guo et al., 2019). These studies extended the work of Ma et al. (2016) by also using RNNs in combination with different features or mechanisms for early detection. For example, Chen et al. (2018) extracted linguistic features from a sequence of micro-blog posts, and combined these with an RNN in which a soft attention mechanism was integrated. This method successfully detects misinformation in an earlier stage. Other research added a convolutional neural network (CNN) to a commonly used RNN for representing propagation paths (Liu and Wu, 2018). The micro-blog posts were represented by simple user characteristics. It was shown that this model detects misinformation after 5 minutes with 92% accuracy, however, a limitation of these models is that the earliness of detection depends on the length of the propagation path (e.g. number of retweets). This means that only with abundant data at an early stage of dissemination these models are suitable for early detection. For example, Liu and Wu (2018) needed approximately 40 tweets in the first 5 minutes to detect misinformation with 92% accuracy. Since propagation paths vary in size this approach does not always detect misinformation after 5 minutes. We therefore aimed to design a detection method that is independent to the number of micro-blog posts.

Thus, misinformation detection algorithms differ in features, classifier, and the time of detection. Only a few studies investigated the effectiveness of different features groups for different times of detection. Kwon et al. (2017) evaluated a comprehensive set of linguistic, network, user, and temporal features for time windows from 3 to 56 days. They showed that the effectiveness of temporal and network features increases over time while that of linguistic features stayed the same. However, linguistic features outperformed all other feature groups for the smallest time window (3 days). Another interesting finding is that a combination of all features was optimal for the largest time window while for the smallest time window this model was outperformed by a combination of user and linguistic features. The results are evidence that optimal feature selection may depend on the targeted detection time.

A study similar to our current approach where misinformation detection is being investigated along with the relative contribution of different feature sets was carried out by Vosoughi et al. (2017). In this study, the detection accuracy was measured as a function of latency for temporal and non-temporal models when using linguistic, user, or propagation features. The results showed that when time passed the temporal model and propagation features became stronger while for real-time detection non-temporal and linguistic features slightly outperformed the others, though not very accurate. Their best model achieved a classification accuracy of 55% for near real-time detection and reached a maximum of 75% after some time.[1]

2.5 Evaluation Methods

The evaluation of misinformation detection algorithms applied to texts is similar to traditional classification tasks. A dataset of documents or micro-blogs is labeled as misinformation or truthful information. The detection algorithms are then trained to classify a document or message into one of these categories. By splitting the dataset into train and test sets the algorithms are evaluated using various metrics (e.g. accuracy, F1-scores, or AUC). However, a major problem is the collection of high-quality data for this problem (Asr and Taboada, 2018). Since detection algorithms are being developed for slightly different tasks, using different types of features, the collection of existing datasets is heterogeneous.

The work of Pierri and Ceri (2019) gives an overview of existing datasets for detecting misinformation. This overview is replicated in Table 2.5. Here we find that most datasets solely contain article content and therefore lack the ability to extract features from social context. Only the *Hoaxy* and *Rumors* datasets have been used

[1]No absolute detection times were presented in this study.

TABLE 2.1: Overview of existing datasets that contain misinformation articles (Pierri and Ceri, 2019).

	Content Features	**Social Context Features**	**Size**	**Labeling**	**Platform**	**Reference**
BuzzFeedNews	Article title and source	Engagement ratings	10^2	BuzzFeed	Facebook	(Silverman, 2016)
BuzzFeedWebis	Full Article	-	10^3	BuzzFeed	Facebook	(Potthast et al., 2017)
DeClare	Face-checking post	-	10^5	NewsTrust PolitiFact Snopes	-	(Popat et al., 2018)
FakeNewsAMT	Article text only	-	10^3	Manual GossiCop	-	(Pérez-Rosas et al., 2017)
FakeNewsNet	Full article	Users metadata	10^3	BuzzFeed Politifact	Twitter	(Shu et al., 2018)
Hoaxy	Full article	Diffusion network Temporal trends Bot score (for users)	>10^6	-	Twitter	(Shao et al., 2016)
Kaggle	Article text and metadata	-	10^4	BS Detector	-	(Risdal, 2017)
Liar	Short statement	-	10^4	PolitiFact	-	(Wang, 2017)
SemEval-2017 Task8	Full article Wikipedia articles	Threads (tweets, replies)	10^4	Manual	Twitter	(Derczynski et al., 2017)
Rumors	Fact-checking title	Diffusion network (Twitter) Original message, replies (Weibo)	10^4	Snopes Weibo	Twitter Sina Weibo	(Ma et al., 2016)

to construct diffusion networks of how misinformation articles have been disseminated on OSNs. The *Hoaxy* dataset consists of misinformation and fact-checking articles, and the tweets that posted a link to these articles on Twitter. Since fact-checking articles are a specific type of news articles this dataset has not been used for general misinformation detection but mostly to study the difference between diffusion networks of misinformation compared to fact-checking articles (e.g. Shao et al. (2018a)). The *Rumors* datasets consist of rumors (a story whose truth value is unverified or deliberately false), and non-rumors, and related posts on Twitter and Sino Weibo (a Chinese OSN). This dataset is collected by gathering the titles of news articles that were classified as rumor/non-rumor by the fact-checking website Snopes[2]. By extracting keywords from these titles the related micro-blog posts have been collected. The *Rumors* datasets are widely used for misinformation detection and because it contains temporal information about how a story is spread it is the most used dataset for early detection (e.g. Liu and Wu (2018)).

Unfortunately, the Twitter policy only allows you to publish tweet IDs which means that Twitter's API (Application Programming Interface) should be used to reconstruct these datasets. Since Twitter has started to actively remove suspicious accounts and tweets in 2018[3] it has become impossible to fully reconstruct the datasets. Moreover, because the production and dissemination of misinformation is constantly changing, detection algorithms should be evaluated using up-to-date data.

Summary

This chapter provides an overview of existing literature on the influence and detection of misinformation on OSNs. In summary, detecting misinformation in real-time is crucial to be effective, and existing detection methods rely on characteristics that discriminate misinformation from truthful information on OSNs. These characteristics are captured by extracting features and classified using machine learning methods. The most effective features rely on network, linguistic, or temporal information extracted from the micro-blog posts on OSNs. However, most existing detection methods use these features to detect misinformation after it already has been disseminated. Motivated by this shortcoming we designed a methodology to study the effectiveness of various features for early misinformation detection which will be outlined in the remainder of this thesis.

[2]https://www.snopes.com/
[3]https://www.nytimes.com/2018/07/11/technology/twitter-fake-followers.html

Chapter 3

Data

In order to develop and evaluate our methods for the automatic detection of misinformation we need a dataset that includes linguistic, network and temporal information about misinformation on OSNs. As described in the previous chapter existing misinformation datasets are scarce, homogeneous and often impossible to fully reconstruct. As a result, no publicly available dataset exist that fits the purposes of this research. Therefore, in this chapter, we give answer to the following research question:

SQ1. *How can we construct a dataset for political misinformation detection that includes linguistic, network and temporal information about news articles broadcast on an online social network?*

To answer this question we address the issues related to misinformation data and discuss, and motivate, how we dealt with these problems to create a novel dataset. Furthermore, to illustrate the quality of our data, a detailed description is given about how this dataset is constructed. Finally, to contribute to the community and encourage more research in this domain we have made the dataset available in the form of a data challenge for the International Conference on Military Information and Communication Systems (ICMCIS) 2020[1] which is hosted on the website Kaggle.[2]

3.1 Challenges

A major problem with misinformation detection is that it works like a game of cat-and-mouse. As detection techniques are evolving so are the methods to generate and disseminate misinformation. This has as consequence that misinformation data quickly becomes outdated. For example, previous studies have shown that social bots play a major role in the dissemination of misinformation on OSNs (Shao et al., 2018a) which makes them good indicators for detecting misinformation. However, with the development of social bots going at a rapid pace the bots nowadays act differently than before (Yang et al., 2019). Therefore, detection methods should be exposed to data with a significant presence of up-to-date social bots. This can only be achieved by creating a recent dataset. In this section we describe the challenges related to creating a misinformation dataset and how our approach tries to overcome these challenges using available resources.

[1] https://icmcis.eu/challenge/

[2] https://www.kaggle.com/c/icmcis2020

The desired dataset for our research includes information that enables us to extract linguistic, network and temporal features from the dissemination of misinformation and truthful news articles on an OSN. In this case the linguistic features are extracted from the linguistic content of the micro-blog posts that share this news article. The network features can then be extracted by reconstructing information diffusion networks from those micro-blog posts. Using the timestamps of the micro-blog posts time-varying diffusion networks can be used to extract temporal information. Therefore, we needed to find an appropriate OSN that provided the required data.

In line with the majority of research on early misinformation detection, Twitter became our platform of choice. The reason for this is that Twitter's data is easily accessible and includes all the data we require. Besides, Twitter has become a popular platform for social science and NLP research because it enables the collection of large quantities of trace data such as tweets and its meta-data. The brevity of the tweets (max 280 characters) also make it an easy choice for NLP tasks such as sentiment analysis. This has led to various large Twitter datasets that are utilised to create linguistic models. However, using Twitter as data provider also has some drawbacks. Maddock et al. (2015) showed that using Twitter's API to collect data results often in incomplete datasets due to the deletion of tweets. As mentioned early Twitter has recently become very active in removing so called suspicious tweets and accounts which especially affects those that share misinformation. Collecting Twitter data will therefore most probably result in incomplete data.

After collecting the data with all the features, in order to use it to evaluate our detection method we will need to label the data. But classifying news articles as misinformation or not is a difficult and labour intensive task. The reason for this is that misinformation is being written to mislead and therefore hard to observe objectively. To exclude a dataset from being biased you want to have objective labels that reflect real world situations. In an attempt to objectively label articles as misinformation or trusted information often multiple labelers are used. This ensures that these labels are of higher quality. Unfortunately, this is very costly and therefore out of the scope of this study. We therefore used another approach that labels the source as misinformation instead of individual news articles. This approach enables us to create a big dataset without many resources. For this we used the existing misinformation identification systems Hoaxy (Shao et al., 2016) to identify misinformation and NewsAnalyser (Brena et al., 2019) to identify trusted information. Note that we collect 'trusted' instead of 'truthful' information since we do not verify the content but base our label on the trustworthiness of the source. Unfortunately, this also affects the quality of the data because content of these news articles is not verified. The constructed dataset in this study can therefore be considered as silver-truth. In remainder of this chapter we further describe the approach we took for creating a novel dataset to better illustrate the quality of our data. Section 3.2 describes how misinformation articles and related Twitter data is collected while Section 3.3 describes this for trusted information.

Furthermore, it has been shown that misinformation detection is topic-dependent (see Section 2.3). This means that the network, linguistic, and temporal features that are extracted from the Twitter data depend on the topic of the news article. Therefore, to better evaluate our detection models and because misinformation is over-represented in political news (Vosoughi et al., 2018) we decided to filter the news articles on politically related or not. The topic-classifier we build utilizes NLP methods for text classification and will be described in Section 3.4.

3.2 Collecting Misinformation

Hoaxy (Shao et al., 2016) is an open source platform[3] for the collection, detection and analysis of online news articles and the dissemination of these articles on Twitter. The platform can be used to collect data of both misinformation and fact-checking articles. To do this a pipeline is constructed in which web scraping, web syndication and Twitter APIs are linked. As input for this pipeline a comprehensive list of 120 low-credibility sources in the U.S. is provided that is compiled and published by reputable news and fact-checking organizations. This list of sources can be found in Appendix A.1. These sources are known for frequently publishing hoaxes, rumors, false news, and conspiracy theories, but may also publish accurate rapports. We label all this data as misinformation and are aware that this makes our data prone to mislabeled instances.

Hoaxy also tracks fact-checking websites, however, in this work only the data of low-credibility sources is used. Giving the list of sources Hoaxy collects data on news stories using RSS (Really Simple Syndication), which enables it to monitor multiple websites using a single news aggregator. To enrich this data the Scrapy framework[4] is used for creating a spider that crawls the link structure of the low-credibility sources.

Because Twitter is a micro-blog that allows only 140 characters per tweet the most common way to share news stories is to include a link to its web article. Therefore, to monitor how news stories are spread, Hoaxy extracts all tweets that share the URL that links to the low-credibility sources. To do this Hoaxy makes use of the *filter* endpoint of Twitter's streaming API[5]. Unfortunately, not all URLs that link to the same article are exactly the same. For example, different query parameters in the URL still link to the same article. Therefore Hoaxy relies on canonical URLs where possible, and applies a URL canonization method in other cases. This method works as follows:

Step 1. Transform URL text to lower case

Step 2. Remove protocol schema (e.g. 'http://')

Step 3. Remove any prefix instance of the strings 'www.' or 'm.'

Step 4. Remove all query parameters

The Twitter data extracted using these canonical URLs are then stored into a database. This pipeline is illustrated in Figure 3.1. As you may notice Hoaxy intends to incorporate a variety of social media platforms but for now only Twitter is monitored.

The data collected by Hoaxy can be obtained using its API.[6] We used this API to fetch all URLs and related tweets in the period from January 1, 2019 until July 31, 2019. To filter out only political articles we build a classifier, this classifier is described in section 3.4. However, to do this we also needed the article's content. For this we used the Python library *newspaper3k*.[7] Some articles were unavailable which means we had to remove them from our data. Finally, to make sure that we could analyze the

[3]Note that this platform is also used to construct the Hoaxy dataset that is mentioned in Section 2.5
[4]`scrapy.org`
[5]`https://developer.twitter.com/en/docs/tweets/filter-realtime/overview`
[6]`https://rapidapi.com/truthy/api/hoaxy`
[7]`https://github.com/codelucas/newspaper`

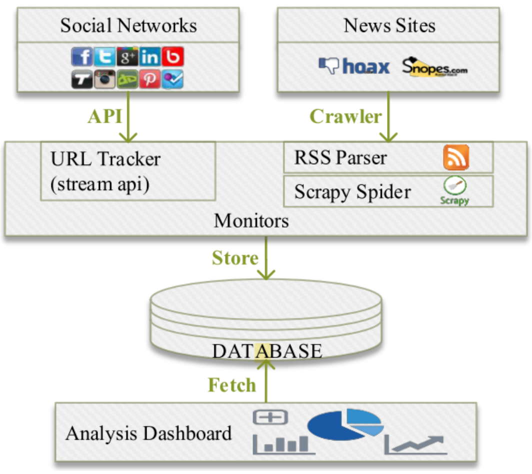


FIGURE 3.1: Architecture of the Hoaxy system (Shao et al., 2016).

dissemination behaviour of these articles on Twitter we removed all articles that had less than 20 tweets.

3.3 Collecting Trusted Information

For the collection of trusted news articles and its related Twitter content we used the open source software NewsAnalyzer (Brena et al., 2019). NewsAnalyzer is a software tool that makes possible the extraction of large collections of Twitter news-sharing users, their news tweets and the full data structure of the shared articles. In order to provide a list of news sources that publish reliable and trusted articles we rely on earlier research that used crowd-sourced judgments to define the quality of a news source (Pennycook and Rand, 2019). In this study the trustworthiness of various news sources from a republican, democratic and fact-checker perspective was being investigated. A combined score from all perspectives was provided to generate a list of most trusted news sources in the U.S. from which we used 9 as input for NewsAnalyzer. A list of the trusted sources we used can be found in Appendix A.2.

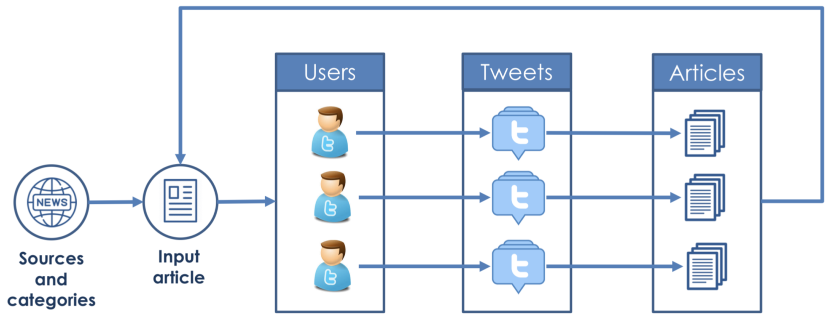


FIGURE 3.2: Architecture of the NewsAnalyzer system as proposed by Brena et al. (2019).

The NewsAnalyzer method consists of two steps as depicted in Figure 3.2. First the list of input sources is used to scrape an initial set of news articles utilizing Python's

Newspaper3k library and are then stored in a *MongoDB* database. After this a continuous loop is started in which the Twitter Search API[8] extracts all tweets that contain the same URL as the articles in the database. As with Hoaxy, only canonical URLs are used. The users who posted those tweets are further investigated by extracting their most recent tweets. If a tweet contains a URL to another news article of one of the provided sources this URL will be stored into the database. This means that the list of URLs is expanded with new URLs and those new URLs will be fed back to the beginning of the loop. At the start of this loop articles are categorized by a built-in classifier that extracts its topic from the URL. This is only done when the topic is mentioned in the URL, which is often the case for major news platforms. The following eight topics are used for this: *politics*, *science-tech*, *world*, *art-entertainment*, *business*, *local*, *style*, and *sports*. If the topic of an article is not mentioned in the URL a machine learning classifier is used to determine this topic. The original classifier uses only keywords as input features to categorize an article. However, as we aim to have only political articles we have build another topic classifier using different features which we describe in more detail in the next section (Section 3.4) and integrated this in the NewsAnalyzer system.

3.4 Topic Classifier

In this section, we describe how we utilised machine learning techniques to build a topic classifier that filters the collected articles on politically related or not.

3.4.1 Data and Data Preprocessing

The data used to train and test this classifier consists of the articles we scraped using NewsAnalyzer which already were categorized based on URL information. This URL information was available for seven out of the nine news sources we used and contains the topic to which an article belongs. NewsAnalyzer used the previously mentioned topics that consist of *politics* and 7 others which we adjusted to *politics* and *not politics*. This resulted in a dataset of 4804 news articles from which 2335 were labeled as *politics* and 2469 as *not politics*.

To clean the texts of the articles some simple preprocessing techniques are applied. First, an article's text is lowercased and digits, punctuation, and stop words are removed. The Porter stemming algorithm (Willett, 2006) is used to remove the common morphological and inflectional endings from words. Finally, we applied NLTK's RegexpTokenizer to identify words.

3.4.2 Feature Extraction

To represent the preprocessed articles by feature vectors we make use of the Doc2Vec algorithm (Le and Mikolov, 2014). This is an unsupervised machine learning technique that, as its name suggests, transforms variable-length documents into fixed-length numerical vectors, also named document embeddings. This method is based on the Word2Vec algorithm (Mikolov et al., 2013) that uses a shallow two-layered

[8] https://developer.twitter.com/en/docs/tweets/filter-realtime/overview

neural network to predict words or contexts (two variants exist) in a given document. After training this neural network the hidden layer is used to output word embeddings. Doc2Vec is based on the same technique but generalizes over all words in the document which enables it to output feature vectors that represent the entire document. An advantage of this method is that it inherits the semantics of the words and is therefore widely used for text classification tasks.

To better understand the inner-workings of the Doc2Vec algorithm we recommend to first read the description of the Word2Vec algorithm in Section 4.3.2 of this thesis. Afterwards, a detailed description of the Doc2Vec algorithm can be found in Appendix B.

For our topic classifier we used the implementation of Doc2Vec by Python's Gensim library.[9] To train the Doc2Vec model the default configurations provided by the Gensim library are used. Only the window size was increased from 5 to 10, indicating the maximum distance between the current and predicted word within a sentence. This was based on a study that compared a window size of 1 and 10 and found that a bigger context window creates more topic-oriented embeddings while a smaller context window creates syntax-oriented embeddings (Bansal et al., 2014). Furthermore, the vector size of the document embedding was set to 300 to create higher dimensionality and a sampling threshold of 1e-5 was used for down-sampling high-frequency words. Using these configurations we let the algorithm train for 100 epochs. The obtained Doc2Vec model was then used to infer 300-dimensional feature vectors for all articles which serve as input for our topic classifier.

3.4.3 Classifiers

Within the NewsAnalyzer model already three different supervised machine learning classifiers were integrated to categorize articles from which the topic was unknown. Although these classifiers were initially used for multi-class classification they work for binary classification problems as well. Additional to the build-in classifiers we implemented a multilayer perceptron (MLP) and compared their performances on the document embeddings that we extracted. In this section, we briefly describe the inner-workings of these classifiers.

Logistic Regression

Logistic regression is named to its core statistical function, the logistic function, which is used to map one or more independent variables to a binary dependent variable. In this case, the dependent variable represents the classes *politics* and *not politics*. The logistic function outputs a probability (p) between 0 and 1 and indicates the likeliness of the input being *politics* (when $p \leq 0.5$) or *not politics* (when $p > 0.5$). The logistic function can by described as follows:

$$p(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

where x is a linear combination of the input variables and represented by:

[9] https://radimrehurek.com/gensim/models/doc2vec.html

$$x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n \tag{3.2}$$

By estimating the coefficients θ_n the algorithm determines a decision boundary between the two classes. To optimize the decision boundary, so it better fits the data, the algorithm uses *stochastic gradient descent*. This method updates the coefficients during training by minimizing the following loss function:

$$Loss(p(x), y) = \begin{cases} -log(p(x)), & \text{if } y = 1 \\ -log(1 - p(x)), & \text{if } y = 0 \end{cases} \tag{3.3}$$

where y is the binary label of a sample in the data. By taking the partial derivative of the loss function a gradient can calculated. This gradient is used to find the direction that moves the loss function towards a local minimum. Furthermore, L2 regularization (Ng, 2004) is used to prevent the model from overfitting the data. This method adds a penalty to the loss function by computing the sum of the squared values of the coefficients. This forces the coefficients to be relatively small which makes the model more robust.

Support Vector Machine

A Support Vector Machine (SVM) aims to find a hyperplane, described by $\vec{w} * \vec{x} - b = 0$, that separates two classes in vector space. The optimal hyperplane is found by maximising the margin between the closest data point of each class and the hyperplane. The two lines parallel to the hyperplane that go through the closest data points are known as the support vectors. Thus, the largest margin is found by maximizing the distance between the two support vectors which is represented by:

$$\frac{1}{2}||\vec{w}||^2 \tag{3.4}$$

However, this hard-margin makes the SVM a linear classifier. To enable the SVM to also classify data that is not linearly separable the algorithm uses a *kernel trick* and a soft-margin. The soft-margin appoints a penalty to all the data points that lay on the wrong side of the decision boundary. The penalty given, is computed by using the *hinge loss*:

$$\ell(y) = \max(0, 1 - t \cdot y) \tag{3.5}$$

where t is the target class (-1 or 1), and y the predicted output value of $y = \vec{w} * \vec{x} - b$. The soft margin is then computed by adding the *hinge loss* to the margin distance:

$$\frac{1}{2}||\vec{w}||^2 + \lambda \left[\frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right) \right] \tag{3.6}$$

where λ determines the trade-off between maximizing the margin and the degree to which data points can lie on the wrong side of the hyper plane. This causes the hyperplane to be relaxed which is favorable for classes that are not linearly separable.

The *kernel trick* is a method that maps the data to a higher dimensional space. This is motivated by the fact that data that is not linearly separable in the input space can be linearly separated in a higher dimensional space. Therefore the linear SVM can be used in this higher dimensional space to find a non-linear decision boundary in the input space.

Random Forest

Random Forest (RF) is part of a group of classifiers that are based on decision trees. A decision tree can be illustrated as a flow-chart in which the root represent a binary class and the leafs with nodes represent the features. Taking a top down approach simple decision are being made at each node that are based on the feature values. This eventually leads to a predicted class at the root of the tree. To determine the best split for a given node the quality of the split is being estimated. In this study, a Gini impurity metric is used for this, which measures the number of correctly classified instances when randomly picking samples of the dataset, and classify these according to the class distribution. The Gini impurity metric is calculated as follows:

$$G = \sum_{i=1}^{C} p(i)(1 - p(i)) \tag{3.7}$$

where C is total number of classes and $p(i)$ is the probability of picking a sample from the data with class i. By weighting the Gini coefficient according the size of the branch a weighted Gini impurity score, named the Gini Gain can be computed. When training a decision tree the split with the highest Gini Gain constitutes the best split for the given dataset. A disadvantage of decision trees is that they easily overfit the training data, which is why RF has been proposed. This algorithm creates a large number of individual decision trees that operate as an ensemble. By predicting the output class for each decision tree the model outputs a final class by taking the class that has been predicted the most. The power of RF lays in the fact that these different trees are relatively uncorrelated with each other since this increases the chance of predicting the right class. To generate multiple relatively uncorrelated decision trees a method named *bagging* is performed. Bagging, also known as bootstrap aggregation, causes the decision trees to be slightly different by randomly choosing samples from the dataset, with replacement. Replacement will say that some samples appear more than once to ensure that the size of the dataset for each decision tree is equal. Moreover, RF uses a random subset of the input features for each decision tree to predict an output class. In this way, there is even more variation between the decision trees which results in lower correlation. The less correlated the decision trees are the better predictions the model makes.

Mulilayer Perceptron

A multilayer perceptron (MLP) is a type of artificial neural networks (ANNs) that, as it name suggests, consists of multiple layers of perceptrons. Where a traditional perceptron functions as a linear classifier, can an MLP distinguish data that is not linearly separable. In Figure 3.3 the architecture of a 3-layered MLP is illustrated. This network consist of an input layer, a hidden layer, an output layer, and the weights that connect the cells with each other. Note, that a bias cell is added to each layer, except the output layer, as this increases the flexibility of the model to fit the data. The input layer represents the feature vector, in this case a 300-dimensional document embedding, and by forwarding the feature values through the network the classifier makes a prediction. Afterwards, backpropagation is used update the weights which enables the algorithm to learn.

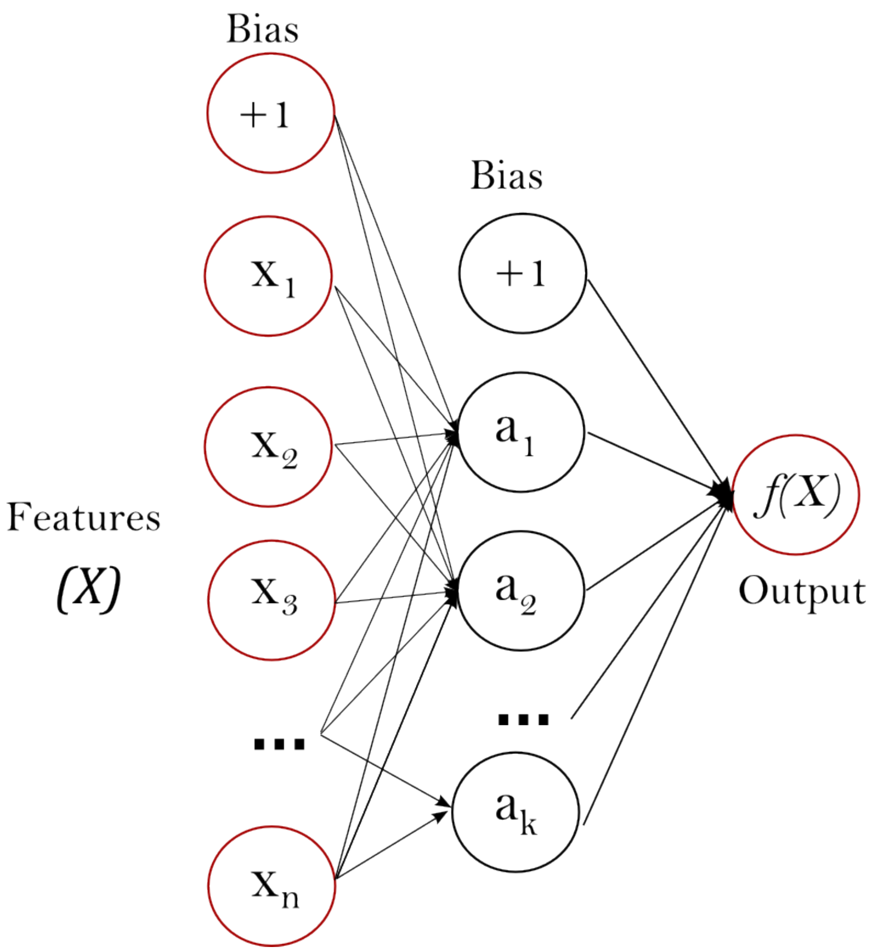


FIGURE 3.3: Architecture of a multilayer perceptron (Pedregosa et al., 2019).

Feed-forward propagation is done by taking the dot product of the input vector and the weights, and results in activation values for all the hidden cells. Then a non-linear activation function is applied to each hidden cell. This ensures that the network can approximate a non-linear decision boundary. Although various activation function exist we used a rectified linear unit also called ReLu function (Equation 3.8).

$$f(x) = max(0, x) \tag{3.8}$$

The same operations are performed to compute the output activations. These activations are then transposed to output probabilities in a range between 0 and 1 as follows:

$$p_O(k) = \frac{e^{a_O(k)}}{\sum_{k=1}^{n} e^{a_O(k)}} \tag{3.9}$$

where $a_O(k)$ is the activation of the output cell, n is the number of output cells and $p_O(k)$ is the output probability of that cell. In this case, the output layer has one output cell because we only have two classes (*politics* and *not politics*). This means that the input vector is classified as *politics* if $p_O(k) \leq 0.5$ and as *not politics* if $p_O(k) > 0.5$.

Back-propagation is applied to update the weights in the network by computing a gradient for each weight. The update rule for each weight works as follows:

$$W_{ji}(new) = W_{ji}(old) + \eta * \Delta_H(j) * a_H(j) \tag{3.10}$$

where W_{ji} is the weight between the cell of the actual layer and the cell of the previous layer, η is the learning rate, $a_H(j)$ the activation from the previous neuron and $\Delta_H(j)$ represents the gradient for each weight. For the output layer the gradient is calculated by subtracting the actual output probability from the desired output. For the hidden layer this is done by computing the derivative of the activation function and multiplying this by a summation of the output gradient ($\Delta_O(k)$) times the weights between the hidden cells and the output cell (W_{ik}):

$$\Delta_H(i) = f'(a(i)) * \sum_{k=1}^{n} (\Delta_O(k) * W_{ik}) \tag{3.11}$$

where $a(i)$ is the activation value of the actual hidden node and n is the number of nodes in the output layer.

By repeating this process of feed forwarding and back-propagation the model learns to approximate a function that maps a set of input features to an output class. However, if the learned function to closely fits the data on which it is trained the MLP is overfitting. This means that the classifier can not generalize and therefore lacks the ability to correctly classify unseen data. To prevent the MLP from overfitting we use a regularization technique named Dropout (Srivastava et al., 2014).

3.4.4 Experiments and Results

During experimentation the hyper-parameter configurations for Logistic Regression, Random Forest, and the Support Vector Machine were kept the same as in the original NewsAnalyzer model. Only for the MLP hyper-parameter tuning was performed by doing a random search on a 70/30 percent train/test split of the dataset. This resulted in a learning rate of 0.01, batch size of 32, dropout rate of 0.5 and 400 hidden cells. To compare all classifiers we have performed 10-fold cross validation using the aforementioned dataset of *politics* and *not politics* articles. In Table 3.1 the results are shown.

Classifier	Accuracy	F-score	Recall	Precision
Logistic Regression	0.73 ±0.04	0.76 ±0.03	0.76 ±0.03	0.77 ±0.03
Random Forest	0.78 ±0.04	0.80 ±0.02	0.80 ±0.02	0.81 ±0.02
Support Vector Machine	0.77 ±0.03	0.77 ±0.04	0.77 ±0.03	0.78 ±0.03
Multilayer Perceptron	0.94 ±0.07	0.94 ±0.07	0.94 ±0.07	0.94 ±0.07

TABLE 3.1: Average classification accuracy, F-score, recall and precision with their standard deviation.

From these results we can conclude that the multilayer perceptron (MLP) outperforms all other classifiers in this task. We therefore selected the MLP as our classifier and trained the model again using the entire dataset. The obtained topic classifier is then utilised to classify articles into the binary classification of either misinformation and trusted articles (those that were not yet categorized) as politically related or not.

3.5 Dataset

The dataset that we collected using the aforementioned methods consists of 1300 political-related misinformation and trusted news articles along with the tweets that disseminated these articles on Twitter. The dataset is equally balanced between classes which results in 650 articles per class. In Table 3.2 some statistics about the dataset are presented.

	Misinformation	Trusted
# Articles	650	650
# Total tweets	168088	78462
# retweets	151054	67401
# original/quote/reply tweets	17034	11061
# original tweets	4328	10433
# quote tweets	10651	41
# reply tweets	2055	587

TABLE 3.2: Statistics about the dataset.

We find that on average misinformation articles have been tweeted more often, most of them are retweets, while trusted articles are more tweeted using original tweets. This suggests that misinformation spreads farther which confirms previous analysis on misinformation data (Vosoughi et al., 2018; Shao et al., 2018a). Another interesting property of the data is that quoted and reply tweets are more used for misinformation than for trusted information. Quoted tweets are retweets in which you share an original tweet while adding a comment to it, and reply tweets are used to only comment on the original tweet. However, the use of these different types of tweets may imply various social behavior so it is difficult to conclude something out of this. In the table, the number of original, quote, and reply tweets are also displayed together because all of these tweets contain new linguistic information while retweets always share the same linguistic information as the original tweets. Therefore, retweets are especially useful to analyze the dissemination behavior of these articles on Twitter.

Finally, a disclaimer has to be made. As already discussed in section 3.1 Twitter is actively removing tweets and therefore, especially in case of misinformation, the dataset contains missing values.

Summary

This chapter addressed the challenges that arise when constructing a misinformation dataset and motivated the construction of a novel dataset. Using two existing tools (Hoaxy and NewsAnalyzer) it was shown how to construct a misinformation dataset that includes linguistic, network, and temporal information about news articles broadcast on Twitter. The labels of the articles are not manually verified but based on the credibility of the news sources. This means the proposed dataset can be considered silver standard. Furthermore, a topic classifier was created in which document embeddings were extracted from the texts of the articles and classified by utilizing a multilayer perceptron. This topic classifier is used to filter the collected articles on politically related or not. Ultimately, this resulted in a dataset of 1300 politics articles with related tweets that disseminated these articles on Twitter.

Chapter 4

Feature Engineering

To find out what are good features for misinformation detection and how to extract them from the proposed dataset —described in chapter 3— we formulated the following research question:

> **SQ2.** *What linguistic, network, and, temporal features discriminate misinformation from trusted information and how can we best extract them?*

In this chapter we demonstrate how Twitter diffusion networks are being reconstructed from the collected data and propose a novel method for capturing the temporality of these diffusion networks by using snapshots. Moreover, this enables us to evaluate the different features for early detection by using only parts of the Twitter diffusion networks. Based on earlier research[1] a group of network and linguistic features is being extracted from these diffusion networks. In this case, the network features include both spreading patterns and user information. Furthermore, we will give detailed descriptions of the linguistic features (both handcrafted and statistical) that we extracted from individual tweets and how we transformed these static features into temporal features by utilizing the Twitter diffusion networks.

4.1 Twitter Diffusion Networks

A Twitter diffusion network describes how a piece of information is being disseminated on Twitter through tweets. For the articles we collected we created such networks using Python's DyNetx[2] library to determine their dissemination. In these networks the nodes represent tweets and the edges show the relationship between an original tweet and a share (retweet, quoted tweet or reply tweet). Each node has a timestamp that corresponds to the time that has passed since the first tweet in the network was posted. By iterating over different timestamps we can now observe how the network evolves over time. Note that these networks consist solely of multiple star networks with a maximum cascade length of 1, as depicted in Figure 4.1.

In reality, users could retweet other retweets and therefore create longer cascades. Unfortunately, Twitter's API only provides limited data that points all retweets to the original tweet which makes it impossible to fully reconstruct the original diffusion networks. Though approaches have been proposed in which cascades are approximated based on tweet timestamps and friend-follower relationships it appears

[1]See Chapter 2 for more background literature.

[2]`https://dynetx.readthedocs.io/en/latest/`

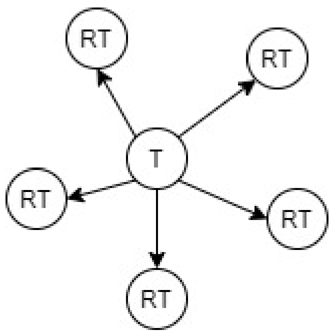
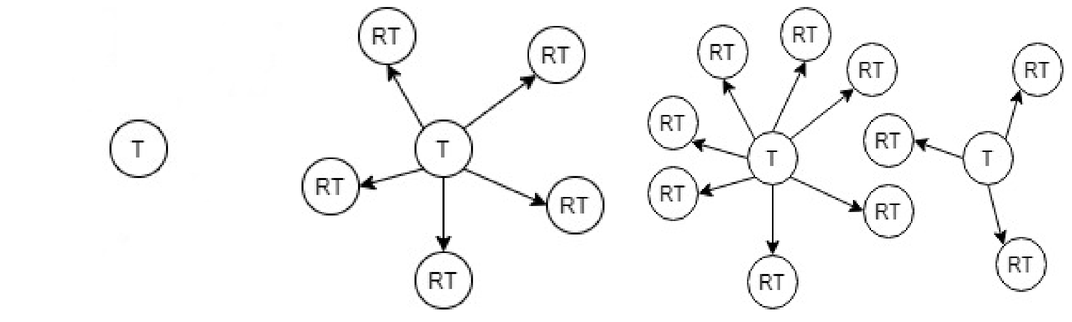


FIGURE 4.1: Example of a star network where T represents an original tweet while RT a retweet.

that this process is time-intensive (Vosoughi et al., 2017) and therefore not suitable for early detection.

By iterating over different timestamps we can now observe how an article diffuses over time. However, since the amount of tweets per article can vary a lot (between 20 and 5000 in our data) we transform these variable-length time series into fixed-length time series. This is enables us to compare the diffusion patterns of different news articles regardless of the size of the network. We do this by dividing the diffusion network into snapshots. Snapshots represent the state of the network at a particular point in time. For example, if the number of snapshots is four ($S = 4$) and the detection deadline is four hours ($T = 4$) than a snapshot represents the diffusion network after every hour. In Figure 4.2 an example is shown of how a diffusion network evaluates over time when using snapshots. As is illustrated in this example, an extra star network originates in Figure 4.2c when a new original tweet is posted and tweet is being retweeted. This means that the network of an article consists of multiple star networks from which the size is determined by the number of original tweets. In Because the time series data we created is highly dependent on the amount of snapshots (S) and the detection deadline (T) we varied with these variables during experimentation.



(A) Snapshot at $t = 0$ (B) Snapshot at $t = 1$ (C) Snapshot at $t = 2$

FIGURE 4.2: These figures illustrate how a Twitter diffusion network evolves over time when using snapshots.

4.2 Network Features

In this section, we describe the features we extracted from the Twitter diffusion networks. We categorized these network features according to the following categories:

diffusion patterns, *followers*, and *social bot* features. The features we describe are extracted per snapshot and therefore variate according to the number of snapshots and the detection deadline that has been chosen.

Diffusion Patterns

The diffusion patterns of a network describe how a network evolves over time. As described earlier the Twitter diffusion networks consist of star networks which have as a consequence that only simple network features could be extracted. The most basic diffusion patterns are therefore captured by tracking the number of nodes, increase in number of nodes, and the number of cascades. Furthermore, we extract Twitter-specific features which include the number of original tweets, number of shares (retweet, quoted tweet or reply tweet), number of likes, and the average number of likes per cascade. To adjust these features to the varying network sizes per article we also computed their relative values by dividing them with the network size of the correlating snapshot. This resulted in eleven features to represent the diffusion patterns.

Followers

On Twitter, information diffusion patterns are related to the users who spread this information. Users with a lot of followers, also called *superspreaders*, have a big audience and therefore tweets by these users reach more people. In Chapter 2 we already mentioned that social bots try to use superspreaders to let an article go viral (Shao et al., 2018b). To find out if superspreaders could indicate the presence of misinformation we extracted the followers count of all Twitter users in our data set. We then computed six different features using these followers counts.

Two features represent the total and relative amount of followers of all users over time. The relative amount is computed by dividing the total amount of followers by the number of users in the network. The reason for this is that the diffusion networks vary in size and this relative amount ensures the standardization of this feature. Two other features are extracted by counting the amount of "well-known users" (> 10,000 followers) and superspreaders (>100,000 followers) in the network. Existing literature did not define the number of followers that a user must have to talk about a superspreader. Therefore we invented this definition our self and added a new one we call "well-known users". The last two features are the relative amount of "well-known users" and superspreaders by using the previously mentioned standardization method.

Social Bots

As described in Chapter 2 the influence of social bots in the dissemination of misinformation on OSNs is enormous. Tracking social bots in the spread of news articles on Twitter could therefore indicate that these news articles contain misinformation. To track social bots we use a state-of-the-art bot detection algorithm called Botometer (Davis et al., 2016). This detection method is suggested to be 86 percent accurate (Varol et al., 2017) and is widely adopted in other research on misinformation detection on Twitter (Shao et al., 2018b; Vosoughi et al., 2018).

Botometer works as follows. A Twitter account is given to the Botometer API[3] which then extracts about 1,200 features using publicly available information and metadata extracted from interaction patterns and content. These features are grouped into six categories: *network*, *content*, *temporal*, *user*, *sentiment*, and *friends*. To classify an account as either a human or a social bot, Botometer makes use of seven pre-trained classifiers. One for each category of features and one that uses all features. These classifiers give a prediction score between 0 and 1, with 0 being most human-like and 1 being most bot-like.

To generate bot features we computed the Botometer scores for all unique Twitter accounts in our data set. We then averaged the Botometer scores across all the Twitter accounts that are present in a specific snapshot. We did this for all seven classifiers which enable us to analyze which category of features is optimal in detecting social bots that spread misinformation. In addition, we computed the average Botometer scores for Twitter users that posted original Tweets and for Twitter users that posted retweets. Finally, we used Botometer's Complete Automation Probability (CAP) score to count the number of users in the network that are most probably automated. When the CAP score of a user exceeded a threshold of 0.5 we classified this user as bot. We then divided the number of bots by the total number of users in the network to compute the percentage of social bots. All these features result in a total of 10 bot features per snapshot.

4.3 Linguistic Features

To represent the linguistic content that characterizes the misinformation and trusted tweets, we extracted two types of linguistic features: tweet embeddings and handcrafted features. In this section, we describe what preprocessing steps have been taken to clean up the texts of the tweets and how we extracted features from them. Unlike the network features, we extracted these features from individual tweets. By averaging a feature over all tweets we were able to compute a feature value per snapshot.

4.3.1 Preprocessing Tweets

Tweets often contain slang, typos, abbreviations, and other types of nonstandard language which results in noisy data. In this study, we examined various feature extraction methods to generate linguistic features. Some of these methods need the original text of the tweet while others benefit from less noisy data. Therefore we have divided the preprocessing part into two steps. After each step, the data is saved for further use.

In the first step, the URLs were removed from the tweets and, when present, retweet mentions were removed. A mention is carried out by using the @ sign immediately followed by someone's username and results in a notification of the tweet for the mentioned user. The retweet mention is a special mention that always refers to the original tweet. This preprocessing step is carried out to prevent the algorithm from being biased. Because the data is collected using the URLs of specific news sources (see Chapter 3) using them for classification could result in the classifier having a

[3]`https://botometer.iuni.iu.edu/#!/`

bias towards a specific news source. The same accounts for retweet mentions because tweets may mention the Twitter account of the news source that posted the article. Note that we do not remove mentions that do not refer to the original tweet that posted the article.

The second step consists of various preprocessing techniques that have been investigated in (Effrosynidis et al., 2017). In this research fifteen techniques are compared for sentiment analysis tasks on tweets. They showed that some of these techniques improved the results while others led to a decrease in performance for these specific tasks. We used the techniques that best suit the types of feature extraction methods we used. For example, because our feature extraction methods do not make use of slang we replaced slang with formal English. We did this by using a lexicon that translated slang to formal English words. Besides this, integers, punctuation, and hashtags were removed using regular expressions. Contractions were replaced by its complete form, causing contractions such as *won't* turn into *will not*. To correct typos and spelling errors we used Norvig's spelling corrector[4]. Finally, we removed all emoticons from the tweets.

The tweets texts saved after the first preprocessing step are indicated as *raw tweets* while after the second preprocessing step we call them *clean tweets*.

4.3.2 Tweet Embeddings

In recent years the development of advanced word embeddings has led to a boost in various NLP tasks such as syntactic parsing (Socher et al., 2013a) and sentiment analysis (Socher et al., 2013b). Word embedding techniques are used to map words from a vocabulary to vectors of real numbers. The main advantage of this technique is that words with similar meanings are located in close proximity to each other in vector space. To see if word embeddings are effective in discriminating misinformation tweets from trusted tweets we used the pre-trained Twitter Word2Vec model from Godin et al. (2015). This model applies the well-known Word2Vec algorithm of Mikolov et al. (2013) on a Twitter corpus of 400 million tweets to generate 400-dimensional word embeddings.

The original architecture of the Word2vec algorithm comes in two flavors: continuous bag of words (CBOW) and skip-gram. The CBOW model predicts a word given its context while the skip-gram model predicts its context given a word. The latter is used by Godin et al. (2015) and will be explained here. In Figure 4.3 an example of a skip-gram Word2Vec model is given. The model consists of a shallow neural network and receives as input a word, represented by a one-hot vector. A one-hot vector is a boolean vector whose length is equal to the size of the vocabulary. In this case, the vocabulary consists of 3,039,345 different words. The position of the input word is then used to set the corresponding vector index to one while all other values stay zero, as shown here:

(1) *Queen* →[0000001000000...]

(2) *King* →[1000000000000...]

(3) *Football* →[0010000000000...]

[4] http://norvig.com/spell-correct.html

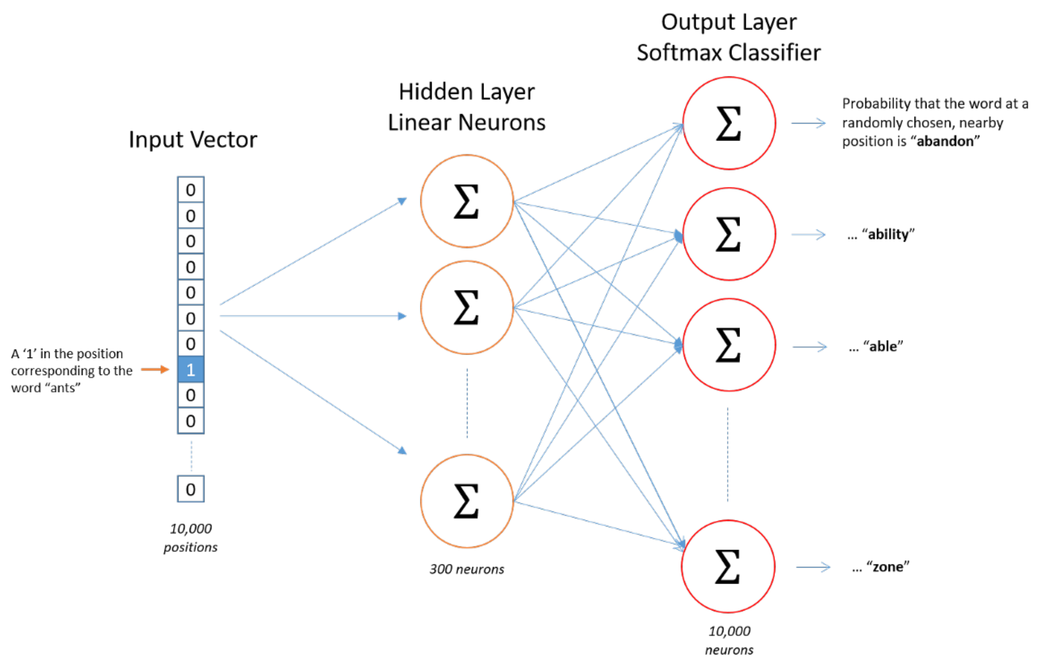


FIGURE 4.3: Architecture of skip-gram Word2Vec model (McCormick, 2016).

A problem with one-hot encoding is that the vector size becomes very large for large vocabularies. Moreover, it does not preserve any semantic meaning of the word since the distance between the one-hot encoding of two words is always the same. For instance, in the example above *Queen*, *King*, and *Football* are equally close to each other in vector space while not in semantic meaning. The Word2Vec model is used to transform this sparse representation into a distributed representation, also known as word embedding. Contrary to a one-hot vector a word embedding is a real-valued, dense, and low-dimensional vector, as shown here:

(4) *Queen* →[0.34, 0.51, 0.74]

(5) *King* →[0.36, 0.51, 0.73]

(6) *Football* →[0.76, 0.21, 0.43]

Using a distributed representation has as advantage that it can make related or similar words occur closer in vector space. This is shown in the example above, *Queen* and *King* have a similar meaning and therefore have a similar vector representation while that of *Football* is totally different. Besides, the vector size is of fixed-length, making it more efficient to store.

A transformation from sparse to distributed representation is achieved by using the skip-gram model. Giving a one-hot vector the model predicts the probability that a context word, which lays in a certain range from the input word in that tweet, is another word in the vocabulary. As shown in Figure 4.3 a neural network that applies a softmax function on the output layer is used for prediction. The softmax function converts the output vector to a probability vector that is again equal to the vocabulary size. This means that for every word in the vocabulary a probability is computed that represents the likeliness of the context word being that word. Because it is known what the actual context word is a loss function is used to compute the loss between the predicted and targeted output. This loss is then backpropagated through the network to update the weights.[5]

By iterating over all the tweets in the Twitter corpus the model learns to predict the dependency relationships between words. To represent these relationships, not the trained neural network itself is used but, instead, the weights of the hidden layer are used as word embeddings. This hidden layer consists of a matrix that is equal to the vocabulary size times the number of hidden nodes. As mentioned earlier, Godin et al. (2015) computed 400-dimensional feature vectors which means that their neural network contained 400 hidden nodes. It appears that this technique is especially good in capturing semantic information when trained on a very large corpus (Lilleberg et al., 2015) which is why a pre-trained Word2vec model is used in this study.

Using this pre-trained Word2Vec model with our own data we were able to construct tweet embeddings. This is done by first computing the word embedding for every word in a tweet. If a word did not occur in the pre-trained vocabulary this word was skipped. Ultimately, the *tweet embedding* was computed by averaging over all word embeddings in the corresponding tweet.

4.3.3 Handcrafted Features

Based on previous research that used linguistic features for misinformation detection and other NLP tasks we constructed a set of most promising handcrafted linguistic features. In this section, detailed descriptions of these features are given.

TextBlob

TextBlob[6] is a Python library that consist of various NLP tools. Previous efforts already showed the strength of the sentiment classifier of TextBlob for the detection of "Fake News" tweets (Krishnan and Chen, 2018; Reis et al., 2019). The sentiment classifier computes a subjectivity and polarity score for a text, based on the adjectives it contains. Adjectives are used to modify or change the semantics of a noun. In Examples 7 and 8, a sentence is shown with (indicated in bold) and without an adjective.

(7) The president of the United States is a **terrible** man.

(8) The president of the United States is a man.

This example shows that using the adjective "terrible" results in a more subjective and negative sentence. Applying the TextBlob sentiment classifier on example 7 will therefore result in a high subjectivity score and low polarity score. The subjective score of TextBlob is presented by a value between 0.0 and 1.0 while for the polarity score this value is between -1.0 and 1.0. These scores represent objectivity versus subjectivity and negativity versus positivity, respectively.

[5]See Section 3.4.3 for a more detailed description on neural networks.

[6]`https://textblob.readthedocs.io/en/dev/index.html`

Orthography

Orthography describes the set of conventions for written language. On Twitter, these conventions have become less strict causing the use of non-standard orthography. Particularly the use of capitalized letters, punctuation, spelling, and emojis is different from conventional texts. To evaluate if the orthography in tweets differs between misinformation and trusted articles we extracted several orthography features which we will describe here.

The frantic use of exclamation marks and capitalized letters is often brought in context with writers that want to cause an emotional reaction. In general, misinformation contains more content intended to evoke outrage. Therefore we hypothesize that misinformation tweets contain more exclamation marks and capital letters. Using regular expressions we extracted the exclamation marks, capital letters, and continuous capital letters, and represented them by total counts and percentages of occurrences. In addition, we looked at hashtags and mentions because these are often used to amplify the spread of a tweet. This may indicate that users who post a misinformation or trusted article are knowingly trying to reach a bigger audience. This strategy is often applied by bots to let a tweet go viral, as found by Shao et al. (2018b). We represented these features by a total count and a binary value that indicates the presence of at least one hashtag/mention. Finally, we extract the tweet length by counting the number of characters in a tweet. This is a feature that is proven to be effective for misinformation detection tasks when part of a larger feature set (Castillo et al., 2011; Buntain and Golbeck, 2017). However, because in this study the feature importance of individual features is investigated as well it will be interestingly to find out if the tweet length is really of discriminative power for misinformation detection.

The rise of social media has led to a significant increase in the use of emojis. Emojis are graphical symbols that represent facial expressions, emotions, objects, or ideograms. A comprehensive study by Novak et al. (2015) investigated the use of emojis on Twitter by analyzing the sentiment of these emojis. They provided a sentiment map of the 751 most used emojis on Twitter.[7] For each emoji, a sentiment score between -1.0 and 1.0 is given. We used this score as a feature whenever an emoji was present in the tweet and if not a neutral score of 0 was given. In the case of multiple emojis in the same tweet, we averaged the sentiment score. In addition to the sentiment score, a binary value was used to indicate if there is at least one emoji present.

Hedges, Subjectivity Cues and other Bias Markers

In a previous study on suspicious and verified news tweets, it was found that suspicious tweets contain significantly fewer hedges, subjective cues, and other bias markers than trusted news tweets (Volkova et al., 2017). Suspicious tweets were labeled as such when posted by one of the 174 suspicious Twitter accounts which they verified manually. To investigate if these features are also of discriminative power on our data we extract them as well.

Hedges Hedges are words that are used to introduce uncertainty about the proposition that follows. This means that the writer is not totally convinced about the

[7] `http://kt.ijs.si/data/Emoji_sentiment_ranking/index.html`

truth of the proposition. Below an example of a sentence with and without a hedge in it.

(9) We **may** have to close the shop.

(10) We have to close the shop.

In this example, the hedge (indicated in bold) ensures that the proposition has been made questionable. To bring this in context with misinformation it seems reasonable to think that hedges are more used in relation to misinformation since it could indicate that the truthfulness of the content is questionable. On the other hand, in scientific papers, which are seen as simple factual texts, hedging is used frequently to indicate the strength of a claim. This shows that a hedge also could imply that the writers well overthought the claim they made. Previous research indeed found that Twitter accounts of verified news sources use more hedges (Volkova et al., 2017). To find out how hedges are used in tweets that disseminate misinformation articles we used a list of hedges constructed by Hyland (2018). We extracted these hedges from tweets using regular expressions and represented them as binary value (present or not present) and a total count.

Subjectivity Cues Opinions are subjective expressions that describe people's sentiments, appraisals, or feelings toward entities, events, and their properties (Liu, 2010). Subjectivity cues in written language refer to the words that characterize these opinions. To find these cues in tweets we made use of the subjectivity lexicon of Riloff and Wiebe (2003) and another lexicon of positive and negative opinion words that are constructed by Liu et al. (2005).

The subjectivity lexicon consists of words that are weakly and strongly subjective. Strong subjective words are mainly used to express subjectivity while weak subjective words are used for both subjective and objective expressions. Positive and negative opinion words are also indicators of subjective language and therefore this lexicon could also be used to detect subjectivity in tweets. When manually analyzing both lexicons we observe high similarity. However, since Volkova et al. (2017) used both lexicons to extract subjectivity features we will do the same. We do this again by using regular expressions and use a binary representation if a word in the tweet occurs at least once in the lexicon and, besides, count the total number of occurrences.

Bias Markers In language, bias markers refer to the words that are flattering, vague or endorse a particular point of view. The previously described hedges and subjectivity cues are examples of bias markers. In this section, we describe four other bias markers (assertive, factive, implicative, and reportive verbs) that are useful for detecting biased language. This was found in a study by Recasens et al. (2013) in which a Wikipedia[8] dataset was used to evaluate the performance of these bias markers. While Wikipedia strives to keep its language unbiased, Twitter is a platform that contains a lot of biased language. This was confirmed by Volkova et al. (2017), who found that both suspicious and verified tweets contained a lot of bias markers. Nevertheless, suspicious tweets contained significantly more bias markers than verified tweets.

[8]https://www.wikipedia.org/

The aforementioned bias markers consist of factive, assertive, implicative, and reportive verbs. Factive verbs presuppose the truth of their complement cause (Kiparsky and Kiparsky, 1968). This means that factive verbs are used when the complement cause is assumed to be true. In Example 11 the verb "knows" implies that Ronald is sure about the truth of the complement and therefore "know" is a factive verb. Because the verb "believes" in Example 12 does not indicate that the complement clause is a fact so "believe" is not a factive verb.

(11) Ronald **knows** the earth is flat.

(12) Ronald believes the earth is flat

On the contrary, assertive verbs do not presuppose the truth of their but assert a level of certainty to the complement cause (Hooper, 1975). This level of certainty depends on the asserting verb that is used. In Examples 13 and 14 the use of assertive verbs is shown. These examples show that the verbs "say" and "claim" imply different levels of certainty.

(13) Frank **says** Russia will attack North Korea if they don't stop producing nuclear weapons.

(14) Frank **claims** Russia will attack North Korea if they don't stop producing nuclear weapons.

Implicative verbs imply —depending on the polarity of the main predicate— the truth or untruth of their complement (Karttunen, 1971). Although factive verbs also imply the truth of their complement clause, implicative verbs polarity of their complement clause changes when a negation is used (see Ex. 15 and 16) while factive verbs do not affect the truth of their complement cause under negation (see Ex. 17 and 18).

(15) Donald does **remember** that he stole a candy from the shop.

(16) Donald does **not remember** that he stole a candy from the shop.

(17) Ronald **knows** the earth is flat.

(18) Ronald **don't knows** the earth is flat.

Reporting verbs are also known as referring verbs since they are used to report or refer to others work. These verbs are frequently used in scientific writing to cite another writer's work. Using these verbs you can comment, agree or disagree, or evaluate someone else's text. This enables people react on the actual context of news article that is being disseminated. Below some examples of the use of reporting verbs.

(19) Some people **claim** that the the Republican Party manipulates voters.

(20) Members of the Republician Party **disagree** with this claim.

Affective Norms for English Words (ANEW)

A well-known assessment technique to directly measure valence, arousal and dominance associated with a person's affective emotion to stimuli is the Self-Assessment

Manikin (Lang, 1980). This technique is primarily used in psychological experiments and works as follows: a stimulus is presented to a person and this person has to express their emotions on this stimulus using the assessment form that is shown in Figure 4.4. The valence score is indicated by five manikins, in a range from happy to *unhappy*, from which the participants have to choose one that indicates their emotional state. The same accounts for arousal (ranged from *excited* to *calm*) and dominance (ranged from *controlled* to *in control*), and results is a three-valued score for emotion. Bradley and Lang (1999) used this technique to measure the emotional response to 1,034 English words. This work was later extended by Warriner et al. (2013) to create a dictionary of 13,915 English lemmas with related norms of valence, arousal, and dominance. In this study, the original rating system of the Self-Assessment Manikin was adjusted from 5-dimensional to 9-dimensional while keeping the original descriptions for range (happy, unhappy, etc.). The scores are therefore represented in the dictionary by a value between 1 and 9. These are average scores over multiple assessments that were collected during a large experiment. In this experiment between 15 and 50 assessments for valence, arousal, and dominance were registered for all 13,915 words.

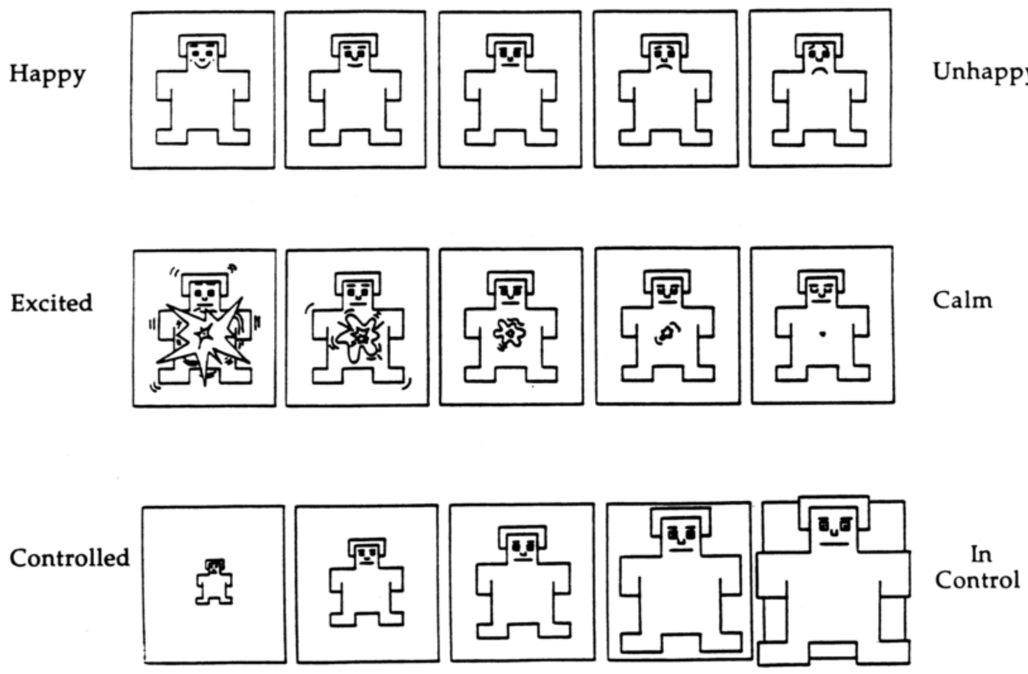


FIGURE 4.4: The Self-Assessment Manikin (SAM) form in which a valence (upper), arousal (middle), and dominance (lower) score can be given by choosing the right manikin (Bradley and Lang, 1994).

Although the standardization of emotional feelings is extremely difficult this dictionary has proven to be useful in various tasks, including social bot detection on Twitter (Varol et al., 2017). To investigate if the emotional response in tweets is different to misinformation articles than to trusted articles we used the aforementioned dictionary to extract valence, arousal, and dominance scores from tweets. Because the dictionary only contains lemmas we first applied lemmatization on the tweets texts using NLTK's part of speech (POS) tagger[9] and WordNetLemmatizer.[10] Additionally, we checked for negations in the previous three words and reversed the polarity of the scores if a negation was found. Finally, we averaged the valence, arousal, and dominance scores from all words in the tweet that were represented in the dictionary.

Verbs of Attribution

Verbs of attribution, also called attribution tags, are used to attribute information to others. This means that these verbs can be used to cite someone else's words, thoughts, or actions. Different verbs imply subtle differences about the citation. The examples below show such a difference when using different verbs of attribution (indicated in bold) in the same sentence.

(21) The New York Times **reveals** that president Donald Trump has secret alliances with Ukraine.

(22) The New York Times **speculates** that president Donald Trump has secret alliances with Ukraine.

In this example, both verbs of attribution (reveals and speculates) are used to cite a claim by *The New York Times*. However, the claim in the first example seems more trustworthy than the second due to the verb that is used. This suggests that verbs of attribution could be used to weaken or strengthen claims. We therefore hypothesize that verbs of attribution are more used in misinformation tweets to evoke vagueness around a related news story or to show agreement (or disagreement).

We extracted these verbs from tweets using a lexicon of attribution verbs.[11] This lexicon was composed to show alternatives for the most used attribution verb, which is "says". However, as we are interested in all attribution verbs we added "says" to the lexicon. We then constructed a binary feature that represents the occurrence of at least one verb of attribution and a real number that represents the total amount of attribution verbs.

Discourse Connectives

Discourse connectives are words or phrases that connect two coherent sentences and indicate the presence of discourse relations. Because discourse connectives are being considered the most reliable signals of coherent relations they are widely used in a variety of NLP tasks e.g. argumentation mining (Kirschner et al., 2015). With argumentation mining, the aim is to automatically identify arguments and argumentative relations in discourse. Since misinformation articles include false or inaccurate claims we hypothesize that tweets related to misinformation include more argumentative language to substantiate or disprove the claim that has been made. Argumentative language often includes attributing causality which is something trusted news sources are much more careful about. An example of argumentative language is illustrated in the following sentence:

[9] https://www.nltk.org/api/nltk.tag.html
[10] https://www.nltk.org/_modules/nltk/stem/wordnet.html
[11] https://www.centralia.edu/resources/docs/verbsatrib.pdf

(23) The deployment of a 5G mobile network is dangerous **because** it leads to health risk.

In this sentence, the word *because* is a discourse connective and indicates the argumentative relation between the two phrases. Therefore, we will identify the presence of discourse connectives to recognize argumentative language in tweets. For identification we utilised an existing list of discourse connectives[12] and used regular expressions to find these words in the tweets. Ultimately, the identified discourse connectives have been represented by a total count and binary indicator so they can serve as features for our detection model.

Summary

In Section 4.1 it was shown how Twitter diffusion networks can be reconstructed to represent the dissemination of news articles on Twitter. To capture the temporality of these diffusion networks a method is proposed in which time series data is constructed by taking snapshots of the network. This method ensures that our detection method is independent to number of micro-blog posts that are available at the time of detection. Utilizing the tweets that are included in the snapshots it was shown how to create temporal-network and temporal-linguistic features. The network features represent the diffusion patterns and user characteristics, such as the number of followers of a user or a bot score that indicates the likeliness of a user being a bot. The linguistic features consist of a comprehensive set of handcrafted features that try to capture various linguistic characteristics such as the orthography, subjectivity, sentiment, and biased language that is used in the tweets. Furthermore, a pre-trained Word2Vec model is used to create 400-dimensional Tweet embeddings. in Table 4.1 an overview is given of all the network and handcrafted linguistic features that have been extracted.

[12]https://www.sparklebox.co.uk/literacy/vocabulary/word-lists/connectives/.XbFoLugzZaR

TABLE 4.1: Overview of all network and handcrafted linguistic features.

Feature	Amount	Representation
NETWORK FEATURES		
Number of nodes	1	int
Increase in number of nodes	2	int
Number of original tweets	1	int
Number of shares	1	int
Number of likes	2	int
Number of cascades	2	int
Average likes per cascade	2	float
Number of followers	2	int
Number of well-known users	2	int
Number of superspreaders	2	int
Average botscores	7	float
Average botscore original tweets	1	float
Average botscore shares	1	float
Percentage of bots	1	float
HANDCRAFTED LINGUISTIC FEATURES		
Polarity score (TextBlob)	1	float
Subjectivity score (TextBlob)	1	float
Number of exclamation marks	1	int
Percentage exclamation marks	1	float
Number of capital letters	1	int
Number of continuous capital letters	1	int
Percentage capital letters	1	int
Hashtags	2	int, binary
Mentions	2	int, binary
Tweet length	1	int
Emojis sentiment score	1	float
Emojis	1	binary
Hedges	2	int, binary
Positive words	2	int, binary
Negative words	2	int, binary
Valence, arousal, dominance	3	float
Weak subjective words	2	int, binary
Strong subjective words	2	int, binary
Assertive verbs	2	int, binary
Factive verbs	2	int, binary
Implicative verbs	2	int, binary
Report verbs	1	int
Verbs of attribution	2	int, binary
Discourse connectives	2	int, binary

Chapter 5

Classification Model

As explained in the previous chapter the features are represented by time series concerning different snapshots of the diffusion network. Therefore, misinformation detection can be considered a sequence classification problem in this case. Given the fact that we want to measure the performance of different features the following research question was formulated:

> **SQ3.** *What classification model is able to incorporate all feature spaces while being able to evaluate each feature space independently from one and other?*

Based on earlier research that used time series data for misinformation detection (see Chapter 2) we decided to use a recurrent neural network as classifier. Additionally, several methods were integrated in the detection model to better fit the purposes of this study. In this chapter, we give a detailed description of the inner-workings of a recurrent neural network and furthermore describe, and motivate, the model choices that have been made.

5.1 Recurrent Neural Network

Recurrent neural networks (RNNs) belong to a set of machine learning algorithms called artificial neural networks (ANNs). These are self-learning algorithms that learn to recognize patterns and use this for prediction. The general working of a simple type of feedforward neural networks, the multilayer perceptron, is been explained in Section 3.4.3, where it was used as a topic classifier. Unlike traditional feedforward neural networks, RNNs are used for sequence prediction problems. RNNs use their internal state as a memory in which information from previous steps in the sequence is remembered. As such, an RNN can model the temporal dynamics of a sequence. This property has already proven to be effective for modeling the dissemination of misinformation on online social networks by Liu and Wu (2018) and Ma et al. (2016).

Figure 5.1 depicts the general architecture of an RNN. As is shown, the hidden cells have a feedback loop (W_H) that redirects the output as an input value. The recurrence of this hidden state is better illustrated by unfolding the RNN into multiple time steps. In doing so, we see that at each time step ($t = [1...n]$) the model receives an input vector (x_t) —that represent the feature values in each snapshot— and a hidden input from the previous time step (h_{t-1}). A linear combination of these input values and related weights is then used to compute the activation of the hidden cells which represents the information that is propagated through the network. This is

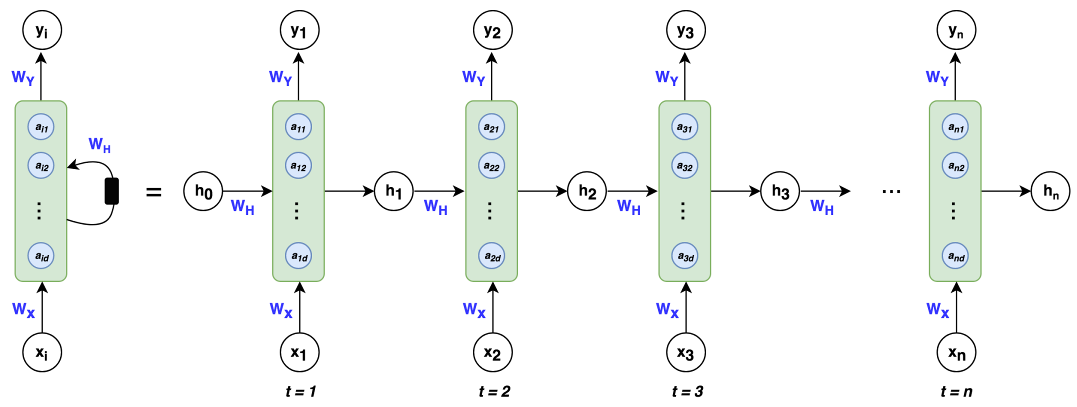


FIGURE 5.1: Architecture of a recurrent neural network (Khuong, 2019).

called forward propagation and is eventually used to generate a predicted output. Equation 5.1 shows how this activation is being computed for a single hidden cell:

$$a_t = W_H h_{t-1} + W_H x_t + b \tag{5.1}$$

Where W_X and W_H represent the input and hidden weights and b denotes the bias. This bias is a constant value that helps to shift the activation to a value that fits best for the given data. This is analogous to the constant value in a linear equation, which is used to effectively transpose the line with a constant value. The activation is computed for all hidden cells, from which the amount is decided by hyper-parameter d, and together represent the internal state or "memory" of the network (see the green block in Figure 5.1). The output of a hidden cell (h_t) depends on the type of cell that is being used. A vanilla RNN cell often uses a nonlinear function such as a hyperbolic tangent to generate the output:

$$h_t = \tanh(a_t) = \frac{e^{2a_t} - 1}{e^{2a_t} + 1} \tag{5.2}$$

The nonlinearity of this activation function is important because it enables the network to create a nonlinear decision boundary for classification. In this work, we use a different type of cell which will be explained in the next section. Finally, to predict the class of an input sequence a final output has to be computed. Because we have a binary classification problem we use a sigmoid function to generate the probability of the input sequence being misinformation (see eq. 5.3).

$$y_t = (W_Y h_t) = \frac{1}{1 + e^{-W_Y h_t}} \tag{5.3}$$

5.1.1 LSTM cell

Since a vanilla RNN cell does not work well with long-term decencies the Long Short-Term Memory (LSTM) cell was introduced by Hochreiter and Schmidhuber (1997). An LSTM uses gates (input, output, and forget) to regulate the information that goes through. These cells have shown to effectively learn what information to remember and whatnot, even for longer sequences. In figure 5.2 an example of an LSTM cell is illustrated that receives an input x_t and a hidden input (h_{t-1}). Additionally, the LSTM cell uses an internal state c_t which functions as a long-term memory that remembers information from earlier in the sequence.

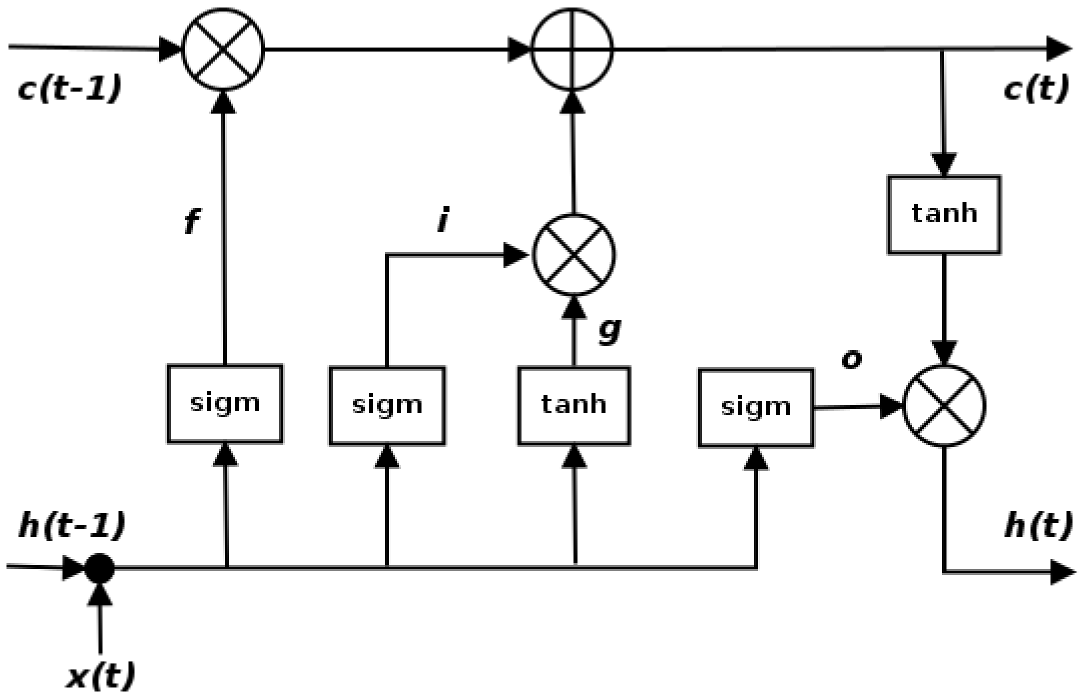


FIGURE 5.2: Diagram that illustrates the inner-workings of an LSTM cell (Gulli and Pal, 2017).

To compute the internal state three different gates are used. The first gate (f_t) is called the forget gate and consists of a sigmoid layer σ that uses the current input x_t and previous output (h_{t-1}) to compute what information to get rid of, see equation 5.4. In this equation W and b represent the weights and biases, respectively. The computed value is between 0 and 1 and defines how much information is forgotten.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.4}$$

The next gate is called the input gate and decides what information has to be stored in the cell state (c). This gate consists of two parts: one sigmoid layer that decides which input values should be inserted into the cell state (Eq. 5.5) and one tanh layer that computes new values that could be given to the cell state (Eq. 5.6).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.5}$$

$$c_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{5.6}$$

By combining old information that should be remembered (forget gate) with the new information that should be stored (input gate) we can update the cell state. Equation 5.7 shows how this is done by multiplying the previous cell state (c_{t-1}) with the output of the forget gate (f_t). Afterwards the new information ($i_t * c_t$) is added which results in an updated cell state. The cell state functions as the "memory" of the cell and is continuously updated after each time step.

$$c_t = f_t * C_{t-1} + i_t * c_t \tag{5.7}$$

Finally, to compute the output —that recurs as input in the next iteration— the cell uses the output gate. This gate again uses a sigmoid layer (Eq. 5.8) to determine which part of the cell state should be outputted and filters the cell state by applying a hyperbolic tangent function (Eq. 5.9).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.8}$$

$$h_t = o_t * \tanh(c_t) \tag{5.9}$$

5.1.2 Learning

In the previous sections, we described how activation is propagated through the network so it can predict which class (misinformation or trusted information) an article belongs to. To let the network learn from previous instances *backpropagation* is being applied. This technique improves the network by adjusting the weights according to the loss, which is the error between the predicted and targeted output. Due to the recursive nature of the weights and their effect on the loss, which spans over time, a slightly different variant of *backpropagation* called *backpropagation through time* (BPTT) is used. Using *gradient descent* a local minimum for the loss function is determined. In this process, the difference between the prediction and actual output is being minimized. Although different loss functions exist we use the most common one for binary classification problems, called binary cross-entropy:

$$L_t(y, \hat{y}) = -(y_t log(\hat{y}_t) + (1 - y) * log(1 - \hat{y}_t)) \tag{5.10}$$

Where y is the target label $[0, 1]$ and $\hat{y}_t$ the predicted probability $(0, 1)$. To compute the loss for an entire sequence the loss of every time step is accumulated:

$$L_{total}(y, \hat{y}) = -\sum_{t=1}^{T}(y_t log(\hat{y}_t) + (1 - y) * log(1 - \hat{y}_t)) \tag{5.11}$$

Now to improve the network the weights (W_X, W_H, W_Y) should be updated in a way that minimizes the total loss. This is done at the end of a sequence by using Equation 5.12.

$$W_i := W_i - \eta \frac{\partial L_{total}(y, \hat{y})}{\partial W_i} \tag{5.12}$$

In this equation $i = [X, Y, H]$ as a shorthand for the three weight matrices and η the learning rate between 0 and 1. The learning rate is a hyper-parameter that determines to what extend the weights should be adjusted. The higher the learning rate the faster the network will learn but this should be decided carefully because a too high learning rate can have as result that the algorithm will not convergence since a minimum is skipped. In the equation above the gradient is represented by $\frac{\partial L_{total}(y,\hat{y})}{\partial W_i}$ which means that a partial derivative of the loss function with respect to a weight matrix is being computed. Since the loss function is composite function the chain rule can be applied to do this. For the output weights W_Y the following equation is used:

$$\frac{\partial L_t(y,\hat{y})}{\partial W_Y} = \frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial W_Y} \tag{5.13}$$

Using the chain rule the gradient for W_Y can be computed by iterating over each time step:

$$\frac{\partial L_{total}(y,\hat{y})}{\partial W_Y} = \sum_{n=1}^{t}\frac{\partial L_t}{\partial W_Y} \tag{5.14}$$

Similar calculations should be done for the gradient of W_X and W_H but this appears to be more difficult. To propagate the loss back to the hidden layer the derivative of the loss function depends on the derivative of the loss function in the previous state as shown here:

$$\frac{\partial L_t(y,\hat{y})}{\partial W_X} = \frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial h_t}\frac{\partial h_t}{\partial h_k} \tag{5.15}$$

where k is equal to $t-1$. Note that, within h_t, h_k also includes W_X. This means that the chain rule should recursively be applied until h_0 is reached, or in other words, the error should be back propagated through time. This is done by using recursion:

$$\frac{\partial L_t(y,\hat{y})}{\partial W_X} = \sum_{k=0}^{t}\frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W_X} \tag{5.16}$$

To finally compute the gradient for W_X the loss again has to be accounted for each time step:

$$\frac{\partial L_{total}(y,\hat{y})}{\partial W_Y} = \sum_{n=1}^{t}\sum_{k=0}^{t}\frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W_X} \tag{5.17}$$

Since W_X and W_H are analogous to each other the same computations are applied to compute the gradient for W_H. Using the gradients the update rule of Equation 5.12 can be applied and the network is able to learn.

5.2 Model Architecture

The described recurrent neural network is a part of larger deep learning model that serves as our classifier. This model was implemented using Keras (Chollet, 2015) and is visualized in Figure 5.3. In this section, the other techniques that are part of this neural network are being described, which consists of an *attention mechanism*, *late fusion*, and *Dropout*.

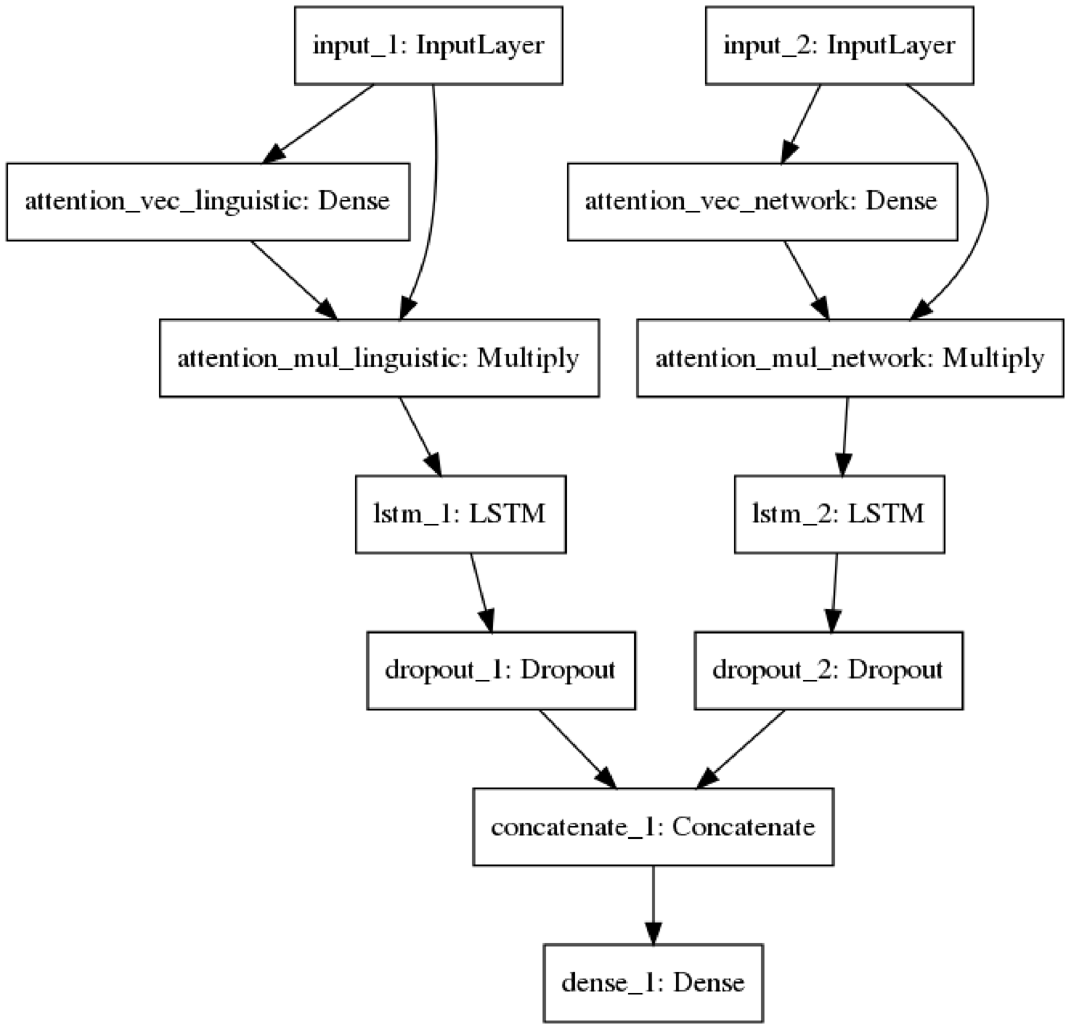

input_1: InputLayer
input_2: InputLayer
attention_vec_linguistic: Dense
attention_vec_network: Dense
attention_mul_linguistic: Multiply
attention_mul_network: Multiply
lstm_1: LSTM
lstm_2: LSTM
dropout_1: Dropout
dropout_2: Dropout
concatenate_1: Concatenate
dense_1: Dense

FIGURE 5.3: Model architecture.

5.2.1 Attention Mechanism

The function of an attention mechanism is to dynamically highlight relevant features in the input sequence. While originally developed to optimize an RNN in an encoder-decoder framework the mechanism has now been adopted in various neural architectures (Galassi et al., 2019). As a result, different implementations of the attention mechanism exist that depend on the architecture and data that is being used. In our case, we add an attention mechanism in between the input layer and the hidden LSTM layer to function as a dynamic feature weighting technique (Kohita et al., 2018). Thus, unlike conventional attention mechanisms for RNNs, that compute weights for various time steps, this attention layer learns to weight features depending on the input. The advantage of using this technique is two-fold. First, it learns the feature importance by linking input values to the target value (misinformation or trusted information). This means that the feature importance is context-dependent which results in different features being important for different misinformation articles. Secondly, it can give a deeper insight into which features are useful in general or for some specific cases of misinformation. This enables us to evaluate the contribution of specific linguistic and network features for the detection of misinformation articles.

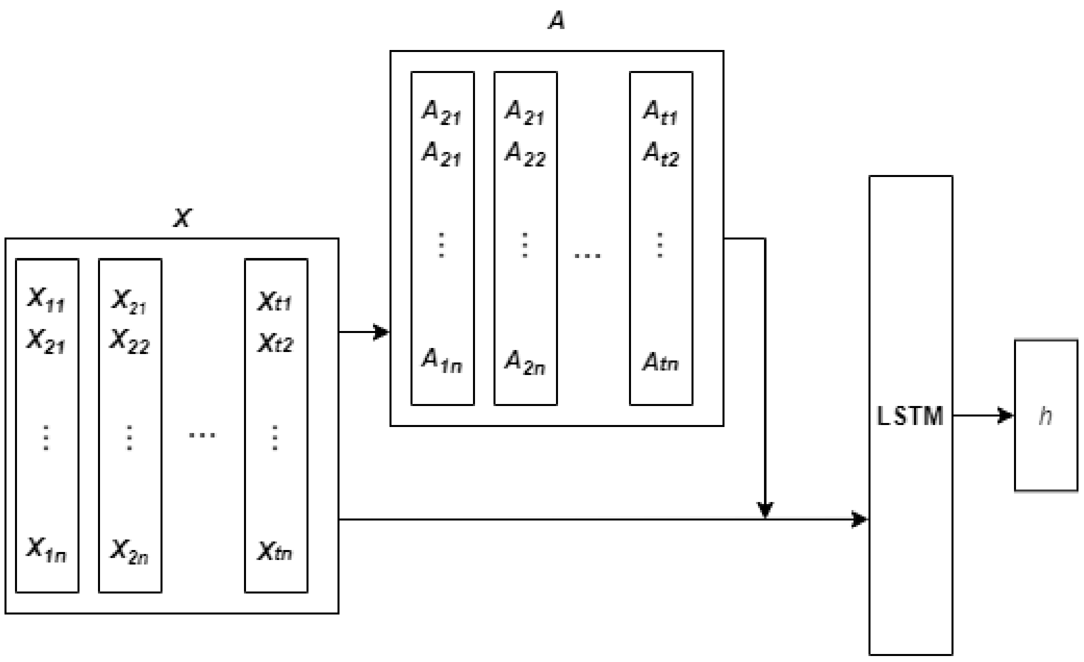


FIGURE 5.4: Architecture of the attention mechanism.

The implemented attention mechanism consists of a dense layer between the input layer and the hidden layer (LSTM), as illustrated in Figure 5.4. Since this mechanism determines the feature importance of each individual feature, the size of this layer is equal to the number of features. At each time step the layer functions as a simple feed-forward layer[1] that uses softmax as activation function. This softmax function normalizes the activation of the cells in this layer as follows:

$$a_n = \frac{e^{W_{An} \cdot x_n}}{\sum_{j=1}^{N} e^{W_{Aj} \cdot x_j}} \tag{5.18}$$

Applying this function for all input features, at every time step, creates an attention matrix (A). The weights regarding the attention layer W_A are updated similarly to the input weights W_X (see Section 5.1.2). Because this attention layer represents the feature importance at every time we multiply the input vector with the attention vector to create weighted features. Afterwards the weighted features are fed to the hidden layer as usual.

5.2.2 Late Fusion

Multi-task learning (MTL) is a subfield of machine learning in which multiple learning tasks are integrated into the same model. This field is inspired by human learning because when humans start learning new tasks they often apply knowledge that is acquired by learning related tasks. For example, when a baby learns to recognize faces it also uses this knowledge to recognize other objects. Based on this principle researchers have proposed various methods to perform multi-task learning in deep

[1] See Section 3.4.3 for more information on feed-forward layers.

neural networks. The most commonly used method to do this is by hard parameter sharing of the hidden layer, which was originally proposed by Caruana (1993). This method learns a combined representation from multiple input streams by concatenating the hidden layers with each other. It turns out that this technique is not only effective for multitask learning but also for single tasks when having multiple input streams. This is demonstrated in various tasks (Volkova et al., 2017; Karpathy et al., 2014) and often referred to as *late fusion*. The advantage of late fusion is that the model first focuses on the individual strength of the features in each input stream and afterwards fuses the semantic representations of those streams. In this study, we extracted two semantically different feature groups, namely network and linguistic features, and therefore we decided to use this method to combine these feature spaces. Besides, it enables us to easily measure the performance of one feature group by using only one input stream and discarding the concatenation layer. As is shown in Figure 5.3 both input streams have an attention mechanism. This means that the feature importance is only measured within one feature group.

5.2.3 Dropout

To prevent the network from overfitting a regularization method called *Dropout* (Srivastava et al., 2014) is being used. Overfitting occurs when the decision boundary that is learned by the network is too closely fit by a limited set of training data points. As a result, the algorithm is not able to generalize and performs poorly on unseen data. *Dropout* helps to prevent this from happening by randomly dropping cells, along with their connections, during training time. The number of cells that are being dropped is determined by a hyperparameter called the Dropout rate. This rate ranges from 0 to 1 and refers to the percentage of cells that is being dropped. In the proposed model, the *Dropout* layer is inserted between the hidden and concatenate layer (or output layer when using only one input stream), as is shown in 5.3. This means it will randomly drop cells of the hidden layer.

Summary

Collectively, this chapter presented a deep learning model in which both temporal-network and temporal-linguistic features can be integrated while also being able to evaluate these feature groups, and individual features, independently from one and other. First, a recurrent neural network is described in detail and enables the model to learn from the temporal information represented by time series data. Furthermore, an attention mechanism was integrated that serves as an internal feature weighting technique while it also increases the interpretability of the model. By evaluating the attention scores the feature importance of individual features can be measured during classification. Finally, a technique called *late fusion* was used to better combine the different feature spaces (network and linguistic) by using two input streams that have their own RNN while afterwards being fused together. Moreover, this enables us to evaluate one feature group by using only one input stream.

Chapter 6

Results & Discussion

To evaluate the performance of the proposed detect method and the effectiveness of different features we conducted a set of experiments. In this chapter, these experiments and their results will be discussed to eventually give answer to the following research questions.

SQ4. *Using the classification model from* **SQ3** *and the identified relevant features from* **SQ2** *to detect misinformation, what is the relative contribution of these feature spaces?*

SQ5. *How do the different feature spaces perform with respect to the earliness of detection?*

Furthermore, we describe the feature importance of the different features that have been used during these experiments based on the attention mechanism of our classification model.

6.1 Experimental Settings

The various feature groups have resulted in five different models represented by the following acronyms: LSTM-N (network features), LSTM-H (handcrafted linguistic features), LSTM-T (tweet embeddings), LSTM-L (all linguistic features), and LSTM-ALL (combines LSTM-N and LSTM-L). For experimentation, we divided the data set into a validation set (20%) and a train/test set (80%). The validation set was used for hyper-parameter optimization based on 10-fold cross-validation with a grid search. The optimal hyper-parameters for our models are shown in Table 6.1. To train the algorithm we applied stochastic gradient descent with the Adam update rule (Kingma and Ba, 2014) and Dropout (Srivastava et al., 2014) was used for regularization. The number of epochs was set to 100 and early stopping was applied when the validation loss saturated for 10 epochs.

	LSTM-N	LSTM-H	LSTM-T	LSTM-L	LSTM-ALL
Learning rate	0.01	0.001	0.001	0.001	0.001
Batch size	20	20	20	20	20
# LSTM cells	50	50	500	600	50 & 600
Dropout rate	0.1	0.1	0.1	0.1	0.1

TABLE 6.1: Model configurations obtained by doing a grid search.

To find out how different models perform with respect to the earliness of detection and the available temporal information we conducted different experiments. First, we investigated if our models were able to learn from temporal information by using snapshots of a diffusion network. We did this by choosing two detection deadlines (15 minutes and 4 hours) and varied the number of snapshots for these time windows. Secondly, we used the optimal number of snapshots to perform the rest of our experiments with detection deadlines between 1 minute and 10 days. For every configuration, we applied 10-fold cross-validation on the 80% train/test data set and computed the average accuracy, AUC, F1-score, precision, and recall plus their standard deviations. A Wilcoxon signed-rank test was performed to measure significance and we reject the null hypothesis when the p-value is lower than 0.05.

6.2 Model Performance

The results of all experiments conducted with the aforementioned configurations can be found in Appendix B. In this section we present and discuss the results that show how model performances differ with respect to the *number of snapshots* and *detection deadlines*. Furthermore, we compare different models according to the features that have been used.

6.2.1 Snapshots

The performance of all models with a detection deadline of 4 hours and varying amounts of snapshots is shown in Figure 6.1. We observe that the accuracy of most models decreases with a larger number of snapshots except for LSTM-N. The network features can take advantage of the temporal information and show an increase

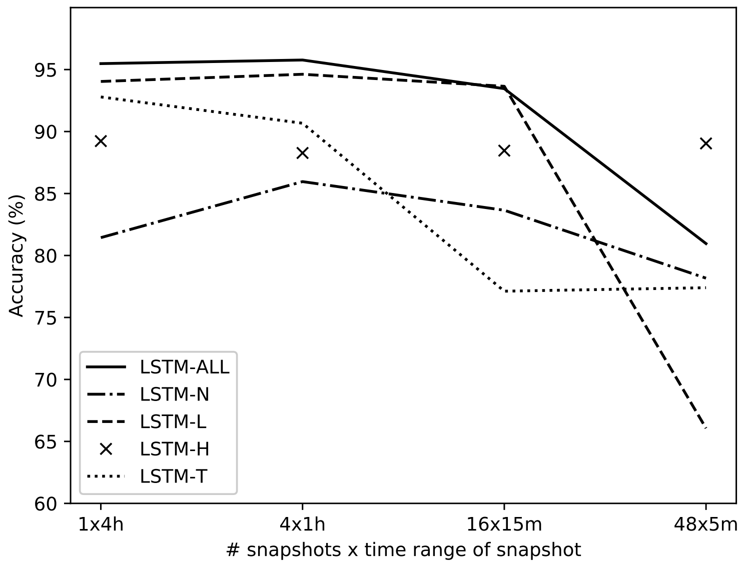


FIGURE 6.1: Model accuracy for varying snapshots and a detection deadline of 4 hours.

in accuracy when the number of snapshots is 4 instead of 1, although not significant ($Z = 13.0, p = 0.138$). For higher amounts of snapshots the performance of all models degrades strongly. We repeated this experiment for a detection deadline of 15 minutes and found similar results, as shown in Table 6.2. Together, the presented findings confirm that our models do not benefit from temporal information for the early detection of misinformation.

When comparing our results to the state-of-the-art that successfully combined temporal features with an RNN as classifier (Liu and Wu, 2018), it must be pointed out that we provided the classifier with different temporal data. As described, we created a tweet volume-independent detection method by introducing the use of snapshots while the model by Liu and Wu (2018) is tweet volume-dependent (see Section 2.4). An advantage of a tweet volume-independent model is that we do not rely on the number of tweets to define the detection deadline but instead use absolute detection deadlines. This also enables us to compare performances of different feature spaces based on the absolute detection deadline instead of the number of tweets. However, as the results suggest the use of snapshots causes the disappearance of valuable temporal information. This is most likely because we include multiple tweets per time step (snapshot) instead of having one tweet per time step. To conclude, the use of multiple snapshots to create temporal data does not add value to our detection model. We therefore performed the remaining experiments using only 1 snapshot.

6.2.2 Detection Deadlines

Figure 6.2 shows how the different models perform for ascending detection deadlines. We observe that each model improves when the detection deadline increases. This effect is observed more clearly for the network features than for the linguistic

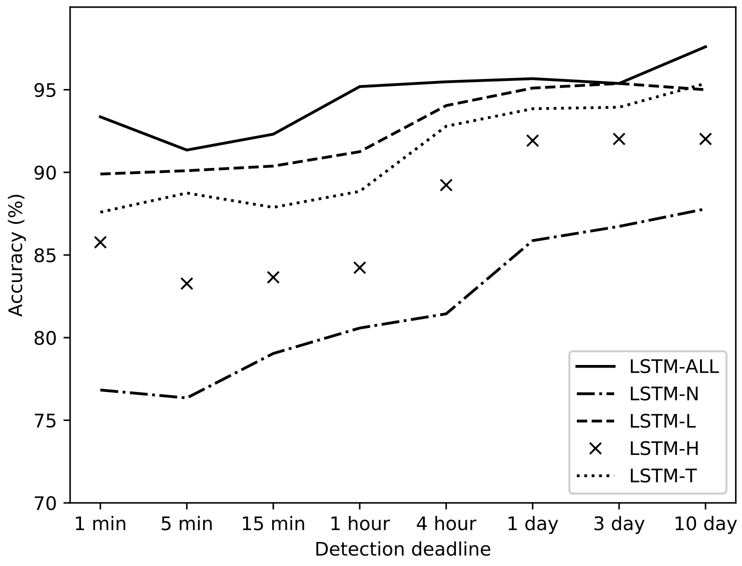


FIGURE 6.2: Model accuracy using 1 snapshot and varying detection deadlines.

features, a result also found by Vosoughi et al. (2017). This increased accuracy makes sense because more social context (e.g. tweets) becomes available as time passes, and with that more information that discriminates misinformation from trusted information. However, for our best detection model (LSTM-ALL) this improvement is minimal, observing an average increase in accuracy of 4.2 percent between a detection deadline of 1 minute and 10 days. For the LSTM-N model this increase in accuracy is much higher, namely 11 percent on average. Nevertheless, these differences might also be (partly) because the LSTM-N model performs poorly at a detection deadline of 1 minute and has, therefore, more room for improvement.

Another noticeable finding is that there is a drop in accuracy for LSTM-H when the detection deadline shifts from 1 minute to 5 minutes. A plausible explanation for this is that the handcrafted linguistic features become noisier when the detection deadline increases since we average the feature values over all the tweets in a snapshot. Why this decline is resurrected after a detection deadline of 4 hours is most likely because the added valuable information surpasses the poor feature presentation. This occurrence implies that our feature representations are not optimal for the handcrafted linguistic features.

6.2.3 Model Comparison

The classification accuracy of all experiments is presented in Table 6.2. We find that linguistic features (both handcrafted and tweet embeddings) outperform the network features for all detection deadlines. Especially for a detection deadline of 1 minute, the linguistic features perform much better which indicates that these features are more reliable for near real-time detection. These findings are consistent with other research that compared linguistic and network features for different detection deadlines (Kwon et al., 2017; Vosoughi et al., 2017).

Detection Deadline	S	LSTM-N	LSTM-H	LSTM-T	LSTM-L	LSTM-ALL
1 min	1	76.8 ± 5.6	85.8 ± 4.4	87.6 ± 4.3	89.9 ± 4.4	93.4 ± 2.5
5 min	1	76.4 ± 4.8	83.3 ± 5.5	88.8 ± 3.0	90.1 ± 4.7	91.4 ± 3.7
15 min	1	79.0 ± 5.6	83.7 ± 4.3	87.9 ± 3.5	90.4 ± 3.6	92.3 ± 3.8
	3	81.4 ± 6.5	85.1 ± 4.0	88.2 ± 3.0	90.4 ± 6.9	94.0 ± 3.6
	15	81.6 ± 4.5	84.6 ± 3.0	86.9 ± 2.9	91.6 ± 4.3	93.3 ± 2.4
1 hour	1	80.6 ± 3.8	84.2 ± 4.8	88.9 ± 3.5	91.3 ± 4.3	95.2 ± 2.5
4 hours	1	81.4 ± 4.4	89.2 ± 3.2	92.8 ± 2.3	94.0 ± 3.1	95.5 ± 2.2
	4	86.0 ± 4.2	88.3 ± 3.4	90.7 ± 3.0	94.6 ± 3.1	95.8 ± 2.8
	16	83.7 ± 4.3	88.5 ± 4.4	77.1 ± 14.5	93.7 ± 3.5	93.5 ± 2.1
	48	78.2 ± 7.5	89.0 ± 3.9	77.4 ± 15.7	66.1 ± 13.3	81.0 ± 6.5
1 day	1	85.9 ± 4.9	91.9 ± 3.2	93.9 ± 3.5	95.1 ± 2.5	95.7 ± 2.3
3 days	1	86.7 ± 5.0	92.0 ± 3.5	93.9 ± 2.3	95.4 ± 2.8	95.4 ± 2.9
10 days	1	87.8 ± 5.6	92.0 ± 3.1	95.4 ± 1.7	95.0 ± 2.8	97.6 ± 1.4
	60	86.8 ± 4.2	81.2 ± 10.3	78.4 ± 14.0	69.6 ± 11.6	92.0 ± 5.6

TABLE 6.2: Misinformation detection accuracy and their standard deviation by doing 10-fold cross validation for different detection deadlines and number of snapshots (S).

Between the different linguistic models we observe that the tweet embeddings are slightly better than the handcrafted features for a detection deadline of 1 minute, however, no significant difference was found ($Z = 14.0, p = 0.169$). When comparing all linguistic features (LSTM-L) with the tweet embeddings (LSTM-T) again no significant difference is found ($Z = 11.0$, $p = 0.171$). Nevertheless, by comparing all linguistic features (LSTM-L) with the handcrafted linguistic features (LSTM-H) a significant difference is found ($Z = 0.0, p = 0.012$). These results reveal that the handcrafted linguistic features and tweets embeddings can supplement each other to improve model performances which was also found by Volkova et al. (2017).

Ultimately, we find that a combination of linguistic and network features (LSTM-ALL) outperforms all other models for near real-time detection ($p < 0.05$). This model can classify articles as misinformation with an accuracy of 93.36 percent after 1 minute. For all other detection deadlines, the LSTM-ALL model also outperforms the others and reaches a maximum classification accuracy of 97.60 percent when the detection deadline is 10 days.

Comparing these results directly with other research on early misinformation detection is arbitrary because we evaluate our model on a novel dataset that is focused on political misinformation. Nevertheless, when looking at the state-of-the-art detection models that we described in Section 2.4 it is at least remarkable that our LSTM-ALL model can detect misinformation with extremely high accuracy in near real-time. For example, Vosoughi et al. (2017), who also compared linguistic, network, and temporal features, achieved very poor accuracy for near real-time detection (55%). On the other hand, Liu and Wu (2018) suggested that their model detects misinformation within 5 minutes with 92% accuracy. However, as mentioned earlier their model needs approximately 40 tweets to reach this accuracy while our model is tweet volume-independent, and reaches a detection accuracy of 93% after 1 minute.

6.3 Feature Importance

As described in Chapter 5 an attention mechanism has been integrated into our detection algorithm to function as a feature weighting technique. In this subsection, the activation values of this attention layer will be presented and we will take a look at the degree of attention that is received by each feature during classification. From these results, we can determine which individual features are most important for misinformation detection with respect to the earliness of detection. This is only done for the network and handcrafted linguistic features because the tweet embeddings are presented by a latent feature space and are therefore not directly interpretable.

The attention scores are based on the input values (features), attention weights, and the softmax function. Since the attention weights are initialized randomly and afterwards updated using stochastic gradient descent the final weights used during classification are non-deterministic. This means that the same model configurations may result in slightly different attention weights. Besides, the input features also vary per article. To minimize the effects of non-determinism and give a good indication about how important each feature is in general we report the cumulative attention scores over all test articles during the aforementioned 10-fold cross-validation. We perform a Wilcoxon Signed Rank test to measure if there are any significant differences in attention scores when the detection deadline varies.

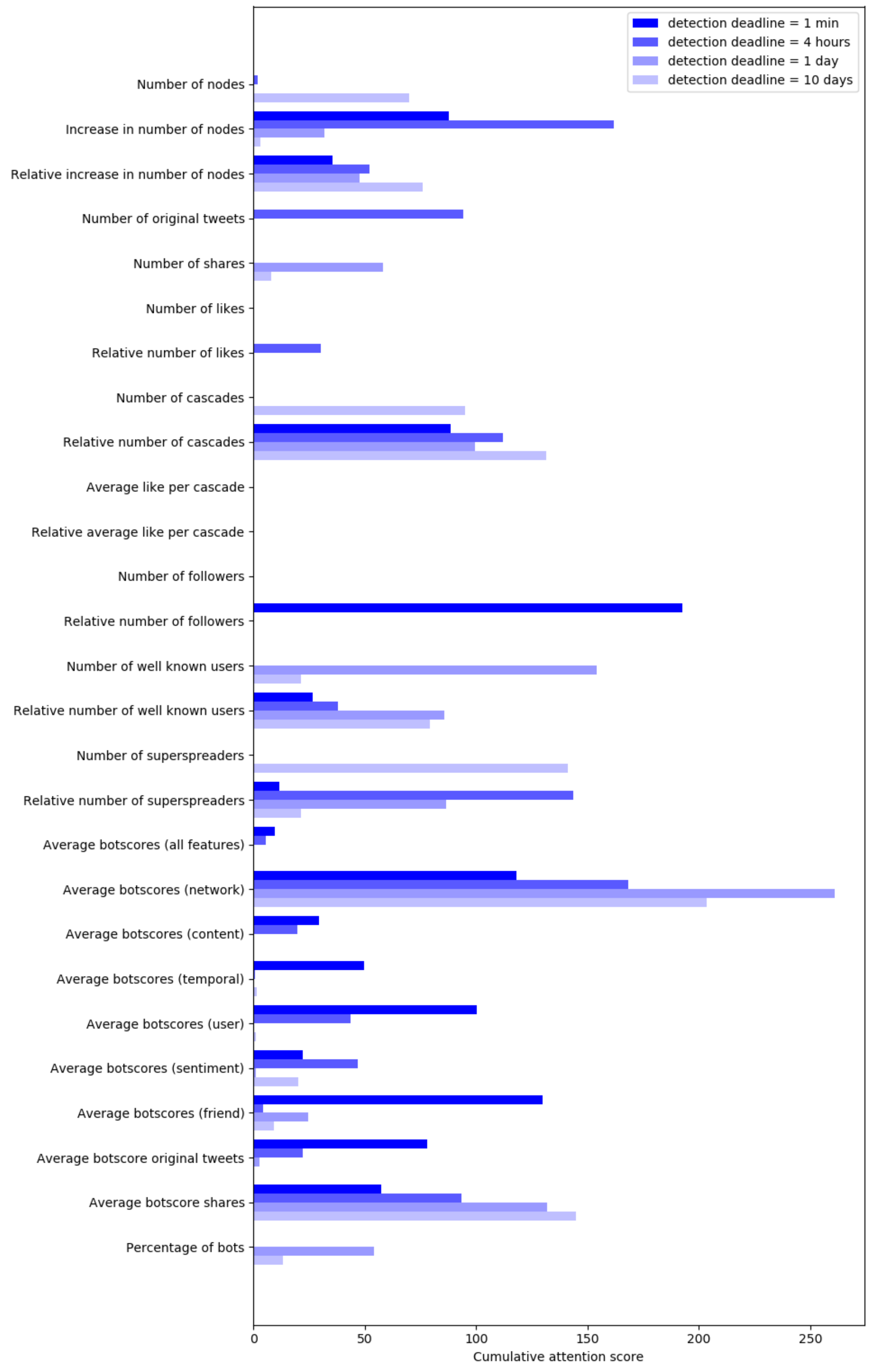


FIGURE 6.3: Average attention over a 10-fold cross validation using the LSTM-N model with 1 snapshot and different detection deadlines.

6.3.1 Network Features

In Figure 6.3 the cumulative attention scores of the network features for various detection deadlines are being shown. The results clearly show that there are considerable differences in importance between different features and between different detection deadlines. In this section, we discuss how the features performed in general and examine the most notable differences between features for different detection deadlines.

Starting with unimportant features we observe that the *number of likes* and *average likes per cascade* get no attention at all. Since these features are both related to the number of likes tweets and cascades of tweets receive we can conclude that likes do not have any discriminative power. On the other hand, *relative increase in number of nodes*, *relative number of cascades*, *relative number of well-known users*, *relative number of superspreaders*, *average botscore (network)*, and *average botscore shares* are all important for classification regardless of the detection deadline. Although, the degree of importance differs between detection deadlines.

Most notable is the *average botscore (network)*, as it has the highest attention score for most detection deadlines. This feature refers to the Botometer score that uses network-related features to compute the likeliness of a user being a bot. Interestingly is that for later detection deadlines this feature becomes even more important while for a detection deadline of 1 minute the other Botometer scores, that are based on different features such as *user* or *friends*, are more important. Noteworthy is that the *average botscore (all features)*, which uses all feature types (more than 1200 features) to compute a botscore, receives almost no attention at all, however, it is unclear why this happened. *Average botscore shares* is the average Botometer score over all shares (retweets, quote, and reply tweets) and becomes more important when the detection deadline increases. This makes sense because for near real-time detection only a few, or even no, shares have been posted. The *average botscore original tweets* represents the average Botometer score in the original tweets and receives more attention for early detection deadlines. This may indicate that bots are used to amplify the spread of misinformation in an early stage, a result also found in another study that analyzed misinformation data that was collected using Hoaxy (Shao et al., 2018b). In general, we can conclude that Botometer scores are useful features for detecting misinformation regardless of the time of detection.

The other features with decent attention scores across different detection deadlines are the previously mentioned *relative increase in number of nodes*, *relative number of cascades*, *relative number of well-known users*, and *relative number of superspreaders*. Especially the first two features show that the structure of a twitter diffusion network is beneficial for discriminating misinformation from trusted information. Surprisingly is that these features seem to be quiet robust across all detection deadlines. Performing Wilcoxon Signed Rank tests we find that for a detection deadline of 1 minute, 4 hours, and 1 day there is no significant difference ($p > 0.05$). Only for detection deadline of 10 days, there is significantly more attention going towards the *relative increase in number of nodes* and *relative number of cascades* ($p \leq 0.05$). Because diffusion networks expand when the detection deadline increases we expected that later detection would benefit from this as more structural information becomes available. Although this is slightly the case these results show that structural features are useful for near real-time detection as well. When comparing these two features with each other we find that the *relative number of cascades* outperforms the number of nodes in terms of attention. As for the *relative number of well-known users* and *relative number of superspreaders*, we observe that the feature importance depends more on the detection deadline. For example, with a detection deadline of 4 hours or 1 day, the *relative number of superspreaders* is far more important than with a detection deadline of 1 minute or 10 days. This example shows how the time of detection influences the performance of a feature during classification.

It is worth mentioning that we varied with the representation of a feature. We used both absolute and relative values as described in Chapter 4. Looking at the attention scores of these features we find that in most cases that the relative feature values lead to attention across all detection deadlines while absolute values highly vary in attention scores between detection deadlines. This indicates that relative feature values are more consistent with respect to the time of detection.

Finally, we found a remarkable attention score for the *relative number of followers*. For all detection deadlines the feature receives no attention except for a detection deadline of 1 minute, for this model the feature has the highest cumulative attention score. Together with the good performance of the *relative number of well-known users* and *relative number of superspreaders* it shows that the number of followers of a user has a good discriminative power.

6.3.2 Linguistic Features

In Figure 6.5 the cumulative attention scores for the handcrafted linguistic features are being shown. The results show that there are only a few features that receive the majority of the attention. Among these features there is the *percentage of capital letters* that is very important for all detection deadlines. To find out which class contains more capital letters we plotted the percentage of capital letters for all the tweets in the entire dataset in Figure 6.4. This figure clearly shows that misinformation tweets

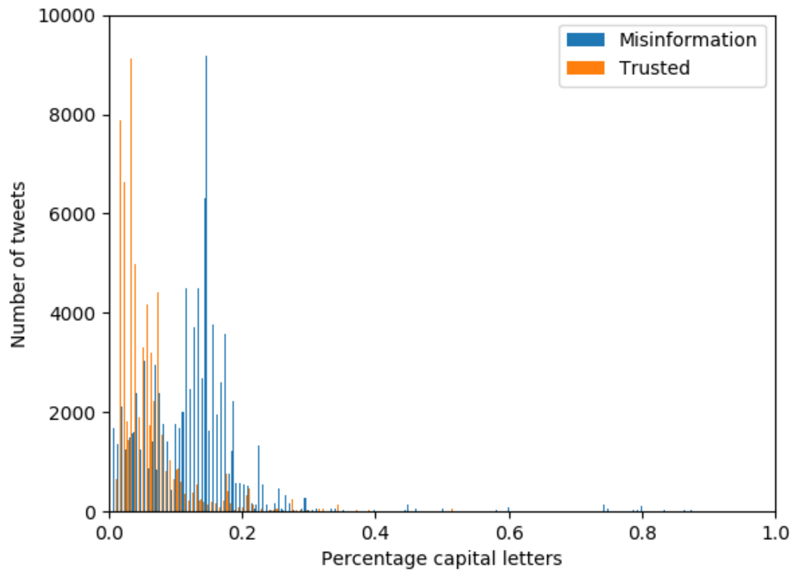


FIGURE 6.4: Percentage capital letters per tweet for the entire dataset.

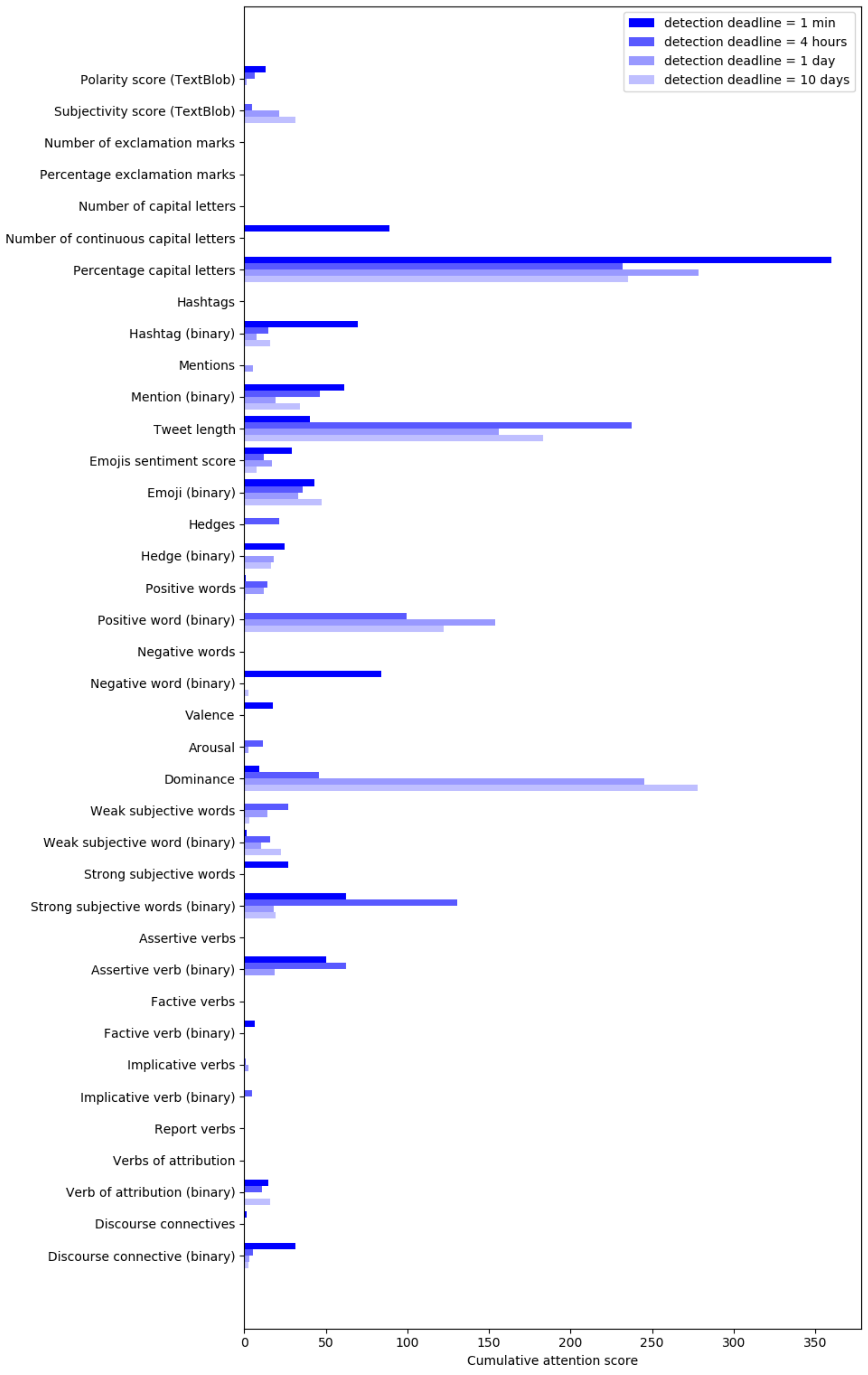


FIGURE 6.5: Average attention over a 10-fold cross validation using the LSTM-H model with 1 snapshot and different detection deadlines.

are generally composed of a higher percentage of capital letters than tweets that broadcast trusted news articles. The great use of capitalization may suggest that more people try to convey importance or urgency, emphasize a though, or grab attention when a news article contains misinformation.

The other peaks in attention are found for the *tweet length*, positive words (binary), and the *dominance score*. All these features have in common that they receive (almost) no attention for a detection deadline of 1 minute while high attention scores for later detection deadlines. For the tweet length this makes sense because more tweets become available when the detection deadline increases, as a result, the average tweet length that is used per snapshot contains less variation and is therefore better generalizable.

Apart from these orthographic features relate the *positive words (binary)* and the *dominance score* more to the semantic characteristics of the tweets. Interestingly to see is that *positive words (binary)* is not important for a detection deadline of 1 minute while *negative words (binary)* is. When observing later detection deadlines we see that exactly the opposite is true. This may indicate that the sentiment of the tweets changes over time. The *dominance score* refers to the dominance or power of the words in a tweet. These scores are based on psychological experiments that examined the extent to which words trigger emotional responses. It is therefore interesting to see that this feature is of major importance for detection deadlines of 1 and 10 day(s) as this indicates that especially after the news articles have been broadcast for some time there is a difference in the use of strong language between misinformation and trusted information.

Furthermore, we find that for the *emoji sentiment score* there is no significant difference in attention score between detection deadlines ($p > 0.05$). This means that regardless of the time of detection the emojis that have been used in the tweets are of discriminative power when separating misinformation from trusted information. Finally, we find that the *hashtag (binary)* and *mention (binary)* features receive significantly more attention for a detection deadline of 1 minute than for later detection deadlines ($p \leq 0.05$). These features indicate the presence of a hashtag or mention in a tweet. Using these hashtags and mentions a bigger audience can be reached and therefore bots use this as a strategy, as found by Shao et al. (2018b).[1] The good performance of these features might therefore be related to the presence of bots since we also found that the bot indicators are important network features.

[1]This study also analyzed misinformation data that was collected using Hoaxy.

Chapter 7

Conclusion & Future work

In this chapter, we recapitulate the relevance of this work, list our most important findings, and draw conclusions from this. Furthermore, we identify the shortcomings of this study and provide suggestions for future work.

7.1 Conclusion

With the increased usage of online social networks, misinformation has become a serious problem for societies worldwide. In this work, we argued that detecting misinformation in real-time, when it begins to spread, is crucial for solving this problem. However, due to the diversity, versatility, and complexity of misinformation, this is a challenging task. Misinformation detection is time-sensitive and topic-dependent, and therefore the main objective of this thesis was to find out which features are most effective for the early detection of political misinformation on online social networks.

Sections 2.5 and 3.1 comprised the challenges that occur when evaluating misinformation detection models. In summary, publicly available, ground-truth datasets are lacking, and constructing high-quality datasets is difficult because labeling misinformation is costly and datasets quickly become outdated. We therefore proposed a method for constructing topic-dependent misinformation datasets that based its labels on source credibility. By utilizing existing tools for tracking the spread of news articles on Twitter we constructed a political misinformation dataset without using many resources. Based on previous research, which showed that misinformation detection is topic-dependent, we believe that the topic classifier has contributed to the good performance of our detection model. To our knowledge, this is the first study that used a topic classifier as a first filter and therefore may be considered a promising aspect.

Utilizing the novel Twitter dataset a comprehensive set of linguistic and network features was extracted, and combined this with temporal information about diffusion networks. Unlike previous studies, temporal information was represented as tweet volume-independent time series by taking snapshots of the Twitter diffusion network. To evaluate and compare the performances of these feature groups, and individual features, a recurrent neural network was used as classifier. Moreover, an attention mechanism was integrated to function as an internal feature weighting technique. In addition to weighting features ensured this mechanism that the importance of different features during classification could be quantified. In this way, the decisions our model makes are better interpretable than existing detection models that utilize neural networks for classification. Important to mention is that the interpretability is directly related to the interpretability of the features that are being used, e.g. tweet embeddings are uninterpretable features (devoid of semantic content) and are therefore also not interpretable through the attention mechanism.

Experiments with the proposed detection model and dataset demonstrated that linguistic features (both handcrafted and tweet embeddings), extracted from tweets, are favorable compared to network features for near real-time detection of misinformation articles. This shows that the linguistic content of tweets contains discriminative information that is rightly captured by the features that we extracted, even in an early stage when only few tweets are available. Extensive knowledge about the contribution of individual handcrafted linguistic features for misinformation detection was provided through the attention mechanism. This showed that especially the percentage of capital letters in a tweet is a good indicator to discriminate misinformation from trusted information at an early stage of dissemination. Nonetheless, the main conclusion that can be drawn from our results is that the effectiveness of individual handcrafted linguistic features is highly dependent on the time of detection. The same accounts for most of the network features. However, in general, we can conclude that the Botometer scores, which indicate the likeliness of a user being a bot, are good features regardless of the time of detection. This is consistent with other research that showed how social bots play a major role in the dissemination of misinformation on online social networks. Furthermore, it was found that using multiple snapshots decreased performances and therefore we conclude that the temporality of the network is not rightly captured by using snapshots. This is probably because valuable temporal information is being lost when splitting the diffusion networks into arbitrary sizes. Nevertheless, using snapshots enabled the model to be tweet volume-independent, which means it does not depend on the number of available tweets, and therefore actual detection times could be measured.

In summary, this study casts a new light on the trade-off that exists between the effectiveness and efficiency of misinformation detection models. We showed how the detection time influences the performance of various linguistic and network features that are extracted from online social networks. With a strong focus on early misinformation detection, we proposed a model that achieves outstanding detection accuracy (93%) for detecting political misinformation articles on Twitter in near real-time. Besides, the proposed model provides a potential mechanism for making detection models more interpretable while concurrently integrating dynamic feature weighting. The proposed model is specially designed to detect potential misinformation, rather than debunk it. This could help fact-checkers to filter the huge amount of news articles that are being published on online social networks, after which another verification step is needed to determine the veracity of the news articles.

7.2 Future Work

A major problem in the development of misinformation detection methods is how to evaluate these methods correctly. In this study, we showed how a large topic-dependent dataset can be constructed based on source credibility. Nevertheless, because the labels are not manually verified, this dataset is considered silver standard.

This limits the reliability of our results. Moreover, it is arguable if even with manual verification it is possible to create ground truth labels because misinformation is hard to observe objectively. We therefore suggest that more research should focus on better evaluation methods for misinformation detection. For example, it would be helpful to deploy our proposed model on Twitter and evaluate its performance in good collaboration with fact-checkers. Another suggestion would be to test the performance of unsupervised learning methods since these methods do not require labeled data. Furthermore, it would help to develop detection models that are better explainable since this enables researchers to find out why the model makes certain decisions.

Future research should also consider the potential benefits of temporal information more carefully. This study showed how the effectiveness of various features is highly dependent on the time of detection. Interestingly would be to use temporal information to integrate this trade-off between effectiveness and efficiency into the detection model. Which means, determine the effectiveness of the detection model at every time step and only do a final prediction when the effectiveness/efficiency trade-off is as desired. This would help to find the optimal balance between the effectiveness and efficiency of the detection model for each news article dynamically.

Ultimately, to steer research towards developing real-time detection methods, we suggest studying the effectiveness of features that are extracted from the content of the article itself. Because this information is available in real-time, it is the only way to detect misinformation before people got exposed to it. Although it has been argued that detecting misinformation based on the text of the article is nontrivial (Shu et al., 2017), the increasing knowledge about misinformation due to academic research will help to better discriminate misinformation from trusted news articles. Moreover, since natural language processing techniques are rapidly advancing they might be able to find patterns that discriminate misinformation from trusted news articles better than humans can do.

Bibliography

Allcott, Hunt and Matthew Gentzkow (2017). "Social media and fake news in the 2016 election". In: *Journal of economic perspectives* 31.2, pp. 211–36.

Asr, Fatemeh Torabi and Maite Taboada (2018). "The data challenge in misinformation detection: Source reputation vs. content veracity". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pp. 10–15.

Bansal, Mohit et al. (2014). "Tailoring continuous word representations for dependency parsing". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 809–815.

Bastos, Marco T. and Dan Mercea (2019). "The Brexit botnet and user-generated hyperpartisan news". In: *Social Science Computer Review* 37.1, pp. 38–54.

Berger, Jonathon M. and Jonathon Morgan (2015). "The ISIS Twitter Census: Defining and describing the population of ISIS supporters on Twitter". In: *The Brookings Project on US Relations with the Islamic World* 3.20, pp. 4–1.

Bessi, Alessandro and Emilio Ferrara (2016). "Social bots distort the 2016 US Presidential election online discussion". In: *First Monday* 21.11-7.

Bradley, Margaret M and Peter J Lang (1994). "Measuring emotion: the self-assessment manikin and the semantic differential". In: *Journal of behavior therapy and experimental psychiatry* 25.1, pp. 49–59.

Bradley, Margaret M. and Peter J. Lang (1999). *Affective norms for English words (ANEW): Instruction manual and affective ratings*. Tech. rep. Technical report C-1, the center for research in psychophysiology ...

Brena, Giovanni et al. (2019). "News Sharing User Behaviour on Twitter: A Comprehensive Data Collection of News Articles and Social Interactions". In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 13. 01, pp. 592–597.

Buntain, Cody and Jennifer Golbeck (2017). "Automatically identifying fake news in popular Twitter threads". In: *2017 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, pp. 208–215.

Caruana, Rich (1993). "Multitask Learning: A Knowledge-Based Source of Inductive Bias ICML". In: *Google Scholar Google Scholar Digital Library Digital Library*.

Castillo, Carlos et al. (2011). "Information credibility on twitter". In: *Proceedings of the 20th international conference on World wide web*, pp. 675–684.

Chen, Tong et al. (2018). "Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pp. 40–52.

Chollet, François et al. (2015). *keras*.

Davis, Clayton Allen et al. (2016). "Botornot: A system to evaluate social bots". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 273–274.

Delirrad, Mohammad and Ali Banagozar Mohammadi (2020). "New methanol poisoning outbreaks in iran following COVID-19 pandemic". In: *Alcohol and Alcoholism (Oxford, Oxfordshire)*.

Derczynski, Leon et al. (2017). “SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours”. In: *arXiv preprint arXiv:1704.05972*.

Edelman Trust Barometer (2020). URL: https://www.edelman.com/trustbarometer.

Effrosynidis, Dimitrios et al. (2017). “A comparison of pre-processing techniques for twitter sentiment analysis”. In: *International Conference on Theory and Practice of Digital Libraries*. Springer, pp. 394–406.

Einstein, Katherine Levine and David M. Glick (2015). “Do I think BLS data are BS? The consequences of conspiracy theories”. In: *Political Behavior* 37.3, pp. 679–701.

Ferrara, Emilio et al. (2016a). “Predicting online extremism, content adopters, and interaction reciprocity”. In: *International conference on social informatics*. Springer, pp. 22–39.

Ferrara, Emilio et al. (2016b). “The rise of social bots”. In: *Communications of the ACM* 59.7, pp. 96–104.

Galassi, Andrea et al. (2019). “Attention, please! a critical review of neural attention models in natural language processing”. In: *arXiv preprint arXiv:1902.02181*.

Godin, Fréderic et al. (2015). “Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations”. In: *Proceedings of the workshop on noisy user-generated text*, pp. 146–153.

Gulli, Antonio and Sujit Pal (2017). *Deep learning with Keras*. Packt Publishing Ltd.

Guo, Bin et al. (2019). “The Future of Misinformation Detection: New Perspectives and Trends”. In: *arXiv preprint arXiv:1909.03654*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Hooper, Joan B. (1975). “On assertive predicates”. In: *Syntax and Semantics volume 4*. Brill, pp. 91–124.

Howard, Philip N. and Bence Kollanyi (2016). “Bots, #StrongerIn, and #Brexit: computational propaganda during the UK-EU referendum”. In: *Available at SSRN 2798311*.

Hwang, Tim et al. (2012). “Socialbots: Voices from the fronts”. In: *interactions* 19.2, pp. 38–45.

Hyland, Ken (2018). *Metadiscourse: Exploring interaction in writing*. Bloomsbury Publishing.

Jolley, Daniel and Karen M. Douglas (2014). “The social consequences of conspiracism: Exposure to conspiracy theories decreases intentions to engage in politics and to reduce one’s carbon footprint”. In: *British Journal of Psychology* 105.1, pp. 35–56.

Karataş, Arzum and Serap Şahin (2017). “A review on social bot detection techniques and research directions”. In: *Proc. Int. Security and Cryptology Conference Turkey*, pp. 156–161.

Karpathy, Andrej et al. (2014). “Large-scale video classification with convolutional neural networks”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732.

Karttunen, Lauri (1971). “Implicative verbs”. In: *Language*, pp. 340–358.

Khuong, Ben (2019). *The Basics of Recurrent Neural Networks (RNNs)*. URL: https://medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f.

Kingma, Diederik P. and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Kiparsky, Paul and Carol Kiparsky (1968). *Fact*. Linguistics Club, Indiana University.

Kirschner, Christian et al. (2015). “Linking the thoughts: Analysis of argumentation structures in scientific publications”. In: *Proceedings of the 2nd Workshop on Argumentation Mining*, pp. 1–11.

Kohita, Ryosuke et al. (2018). “Dynamic feature selection with attention in incremental parsing”. In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 785–794.

Krishnan, Saranya and Min Chen (2018). “Identifying Tweets with Fake News”. In: *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, pp. 460–464.

Kwon, Sejeong et al. (2017). “Rumor detection over varying time windows”. In: *PloS one* 12.1.

Lang, Peter (1980). “Behavioral treatment and bio-behavioral assessment: Computer applications”. In: *Technology in mental health care delivery systems*, pp. 119–137.

Le, Quoc and Tomas Mikolov (2014). “Distributed representations of sentences and documents”. In: *International conference on machine learning*, pp. 1188–1196.

Lewandowsky, Stephan et al. (2012). “Misinformation and its correction: Continued influence and successful debiasing”. In: *Psychological science in the public interest* 13.3, pp. 106–131.

Lewandowsky, Stephan et al. (2017). “Beyond misinformation: Understanding and coping with the “post-truth” era”. In: *Journal of Applied Research in Memory and Cognition* 6.4, pp. 353–369.

Lilleberg, Joseph et al. (2015). “Support vector machines and word2vec for text classification with semantic features”. In: *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, pp. 136–140.

Liu, Bing et al. (2005). “Opinion observer: analyzing and comparing opinions on the web”. In: *Proceedings of the 14th international conference on World Wide Web*. ACM, pp. 342–351.

Liu, Bing et al. (2010). “Sentiment analysis and subjectivity.” In: *Handbook of natural language processing* 2.2010, pp. 627–666.

Liu, Yang and Yi-Fang Brook Wu (2018). “Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ma, Jing et al. (2016). “Detecting rumors from microblogs with recurrent neural networks”. In:

Maddock, Jim et al. (2015). “Using historical twitter data for research: Ethical challenges of tweet deletions”. In: *CSCW 2015 Workshop on Ethics for Studying Sociotechnical Systems in a Big Data World. ACM*.

McCormick, C. (2016). *Word2Vec Tutorial - The Skip-Gram Model*. `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`.

McCright, Aaron M. et al. (2016). “Examining the effectiveness of climate change frames in the face of a climate change denial counter-frame”. In: *Topics in cognitive science* 8.1, pp. 76–97.

Mikolov, Tomas et al. (2013). “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781*.

Ng, Andrew Y (2004). “Feature selection, L 1 vs. L 2 regularization, and rotational invariance”. In: *Proceedings of the twenty-first international conference on Machine learning*, p. 78.

Novak, Petra Kralj et al. (2015). “Sentiment of emojis”. In: *PloS one* 10.12, e0144296.

Ortiz-Ospina, Esteban (2019). “The rise of social media”. In: *Our World in Data*.

Pedregosa, Fabian et al. (2019). *Neural network models (supervised)*. URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#neural-network-models-supervised.

Pennycook, Gordon and David G. Rand (2019). "Fighting misinformation on social media using crowdsourced judgments of news source quality". In: *Proceedings of the National Academy of Sciences* 116.7, pp. 2521–2526.

Pérez-Rosas, Verónica et al. (2017). "Automatic detection of fake news". In: *arXiv preprint arXiv:1708.07104*.

Pierri, Francesco and Stefano Ceri (2019). "False News On Social Media: A Data-Driven Survey". In: *arXiv preprint arXiv:1902.07539*.

Popat, Kashyap et al. (2018). "DeClarE: Debunking fake news and false claims using evidence-aware deep learning". In: *arXiv preprint arXiv:1809.06416*.

Potthast, Martin et al. (2017). "A stylometric inquiry into hyperpartisan and fake news". In: *arXiv preprint arXiv:1702.05638*.

Rapoza, Kenneth (2017). "Can 'fake news' impact the stock market?" In: *by Forbes*.

Ratkiewicz, Jacob et al. (2011). "Detecting and tracking political abuse in social media". In: *Fifth international AAAI conference on weblogs and social media*.

Recasens, Marta et al. (2013). "Linguistic models for analyzing and detecting biased language". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1650–1659.

Reis, Julio C.S. et al. (2019). "Supervised Learning for Fake News Detection". In: *IEEE Intelligent Systems* 34.2, pp. 76–81.

Riloff, Ellen and Janyce Wiebe (2003). "Learning extraction patterns for subjective expressions". In: *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pp. 105–112.

Risdal, Megan (2017). *Fake news dataset*. URL: https://www.kaggle.com/mrisdal/fake-news.

Ruchansky, Natali et al. (2017). "Csi: A hybrid deep model for fake news detection". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 797–806.

Shao, Chengcheng et al. (2016). "Hoaxy: A Platform for Tracking Online Misinformation". In: DOI: 10.1145/2872518.2890098. arXiv: 1603.01511. URL: http://arxiv.org/abs/1603.01511http://dx.doi.org/10.1145/2872518.2890098.

Shao, Chengcheng et al. (2018a). "Anatomy of an online misinformation network". In: *PloS one* 13.4, e0196087.

Shao, Chengcheng et al. (2018b). "The spread of low-credibility content by social bots". In: *Nature communications* 9.1, p. 4787.

Shu, Kai et al. (2017). "Fake news detection on social media: A data mining perspective". In: *ACM SIGKDD Explorations Newsletter* 19.1, pp. 22–36.

Shu, Kai et al. (2018). "Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media". In: *arXiv preprint arXiv:1809.01286*.

Silverman, Craig (2016). "This analysis shows how viral fake election news stories outperformed real news on Facebook". In: *BuzzFeed news* 16.

Socher, Richard et al. (2013a). "Parsing with compositional vector grammars". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 455–465.

Socher, Richard et al. (2013b). "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Van de Guchte, Lennart et al. (2020). "Near real-time detection of misinformation on online social networks". In: *Multidisciplinary International Symposium on Disinformation in Open Online Media*. Springer.

Van der Linden, Sander et al. (2017). "Inoculating the public against misinformation about climate change". In: *Global Challenges* 1.2, p. 1600008.

Varol, Onur et al. (2017). "Online human-bot interactions: Detection, estimation, and characterization". In: *Eleventh international AAAI conference on web and social media*.

Volkova, Svitlana et al. (2017). "Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 647–653.

Vosoughi, Soroush et al. (2017). "Rumor gauge: Predicting the veracity of rumors on Twitter". In: *ACM transactions on knowledge discovery from data (TKDD)* 11.4, pp. 1–36.

Vosoughi, Soroush et al. (2018). "The spread of true and false news online". In: *Science* 359.6380, pp. 1146–1151.

Wang, William Yang (2017). "" liar, liar pants on fire": A new benchmark dataset for fake news detection". In: *arXiv preprint arXiv:1705.00648*.

Warriner, Amy Beth et al. (2013). "Norms of valence, arousal, and dominance for 13,915 English lemmas". In: *Behavior research methods* 45.4, pp. 1191–1207.

Weeks, Brian E. and Homero Gil de Zúñiga (2019). "What's Next? Six Observations for the Future of Political Misinformation Research". In: *American Behavioral Scientist*, p. 0002764219878236.

Willett, Peter (2006). "The Porter stemming algorithm: then and now". In: *Program* 40.3, pp. 219–223.

Yang, Kai-Cheng et al. (2019). "Arming the public with artificial intelligence to counter social bots". In: *Human Behavior and Emerging Technologies* 1.1, pp. 48–61.

Appendix A

News Sources

A.1 Low Credibility Sources

21stcenturywire.com
70news.wordpress.com
abcnews.com.co
activistpost.com
addictinginfo.org
americannews.com
americannewsx.com
amplifyingglass.com
anonews.co
beforeitsnews.com
bigamericannews.com
bipartisanreport.com
bluenationreview.com
breitbart.com
burrardstreetjournal.com
callthecops.net
christiantimes.com
christwire.org
chronicle.su
civictribune.com
clickhole.com
coasttocoastam.com
collective-evolution.com
consciouslifenews.com
conservativeoutfitters.com
countdowntozerotime.com
counterpsyops.com
creambmp.com
dailybuzzlive.com
dailycurrant.com
dailynewsbin.com
dcclothesline.com
demyx.com
denverguardian.com
derfmagazine.com
disclose.tv

duffelblog.com
duhprogressive.com
empireherald.com
empirenews.net
empiresports.co
en.mediamass.net
endingthefed.com
enduringvision.com
flyheight.com
fprnradio.com
freewoodpost.com
geoengineeringwatch.org
globalassociatednews.com
globalresearch.ca
gomerblog.com
govtslaves.info
gulagbound.com
hangthebankers.com
humansarefree.com
huzlers.com
ifyouonlynews.com
infowars.com
intellihub.com
itaglive.com
jonesreport.com
lewrockwell.com
liberalamerica.org
libertymovementradio.com
libertytalk.fm
libertyvideos.org
lightlybraisedturnip.com
nationalreport.net
naturalnews.com
ncscooper.com
newsbiscuit.com
newslo.com
newsmutiny.com
newswire-24.com
nodisinfo.com
now8news.com
nowtheendbegins.com
occupydemocrats.com
other98.com
pakalertpress.com
politicalblindspot.com
politicalears.com
politicops.com
politicususa.com
prisonplanet.com
react365.com
realfarmacy.com

realnewsrightnow.com
redflagnews.com
redstate.com
rilenews.com
rockcitytimes.com
satiratribune.com
stuppid.com
theblaze.com
thebostontribune.com
thedailysheeple.com
thedcgazette.com
thefreethoughtproject.com
thelapine.ca
thenewsnerd.com
theonion.com
theracketreport.com
therundownlive.com
thespoof.com
theuspatriot.com
truthfrequencyradio.com
twitchy.com
unconfirmedsources.com
USAToday.com.co
usuncut.com
veteranstoday.com
wakingupwisconsin.com
weeklyworldnews.com
wideawakeamerica.com
winningdemocrats.com
witscience.org
wnd.com
worldnewsdailyreport.com
worldtruth.tv
yournewswire.com

A.2 Trusted Sources

CBS News
CNN
USA Today
ABC News
The Washington Post
The New York Times
Fox News
NBC News
Huffington Post

Appendix B

Algorithms

B.1 Doc2Vec

In Section 4.3.2 of this thesis the Word2Vec algorithm (Mikolov et al., 2013), on which the Doc2Vec algorithm (Le and Mikolov, 2014) is heavily based, has already been described. As with Word2Vec the algorithm comes in two flavours: skip-gram and continues back of words (CBOW). Since a CBOW implementation of the Doc2Vec algorithm is used in Section 3.4.2 of this thesis, this version will be described here.

The Doc2Vec algorithm uses a neural network to predict a word in the document given its context. The input consists of a randomly taken sample of consecutive words from the document and takes one word out which it has to predict. However, additional to Word2Vec the model adds an extra *paragraph id* as input which is illustrated in Figure B.1. This simple extension of the Word2Vec model enables the model to learn an paragraph vector for each sample that is taken from the document. Finally, a document embedding is computed by averaging or concatenating all the paragraph vectors it has learned. In this study an averaged document embedding was used because a concatenated document embedding results in a very high dimensional feature space.

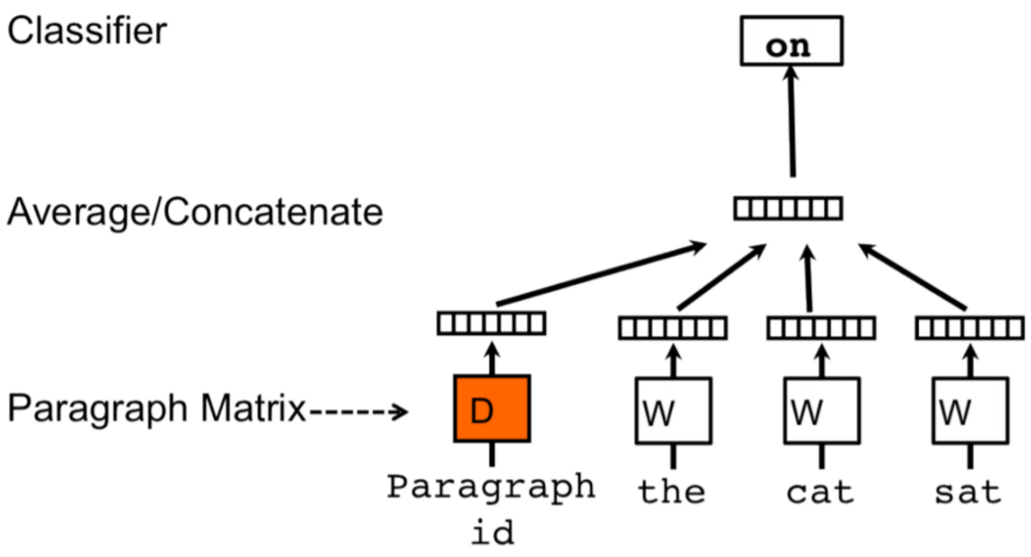


FIGURE B.1: Architecture of the Doc2Vec algorithm (Shperber, 2017).

Appendix C

Results

C.1 Model Performance

Detection Deadline	S	AUC-ROC	F1-score	Precision	Recall	Accuracy
1 min	1	98.56 ± 1.33	93.61 ± 2.30	91.50 ± 5.48	96.15 ± 3.10	93.36 ± 2.52
5 min	1	98.00 ± 1.79	91.19 ± 4.02	92.30 ± 5.34	90.77 ± 7.73	91.35 ± 3.65
15 min	1	98.03 ± 1.45	92.41 ± 3.72	91.79 ± 5.68	93.46 ± 5.52	92.31 ± 3.80
	3	99.13 ± 0.89	94.56 ± 2.56	94.41 ± 4.61	95.00 ± 4.05	94.52 ± 2.58
	15	98.47 ±1.17	93.16 ±2.54	94.39 ±3.87	92.31 ±5.37	93.27 ±2.39
1 hour	1	98.93±0.86	95.24 ±2.40	95.05 ±4.20	95.58 ±2.73	95.19 ±2.47
4 hour	1	99.50 ±0.47	95.47 ±2.28	95.28 ±2.44	95.77±3.73	95.48 ±2.24
	4	99.44 ± 0.43	95.90 ± 2.57	94.30 ± 4.46	97.69 ± 2.07	95.77 ± 2.76
	16	99.18 ±0.60	93.34 ±2.31	94.68 ±4.01	92.50 ±6.11	93.46 ±2.10
	48	90.84 ±4.40	80.01 ±7.78	83.93 ±8.55	78.27 ±13.07	80.96 ±6.46
1 day	1	99.61 ± 0.36	95.69 ± 2.25	95.77 ± 4.57	95.96 ± 4.42	95.67 ± 2.29
3 day	1	99.70 ±0.37	95.25 ±3.37	95.70 ±3.49	94.62 ±7.61	95.38 ±2.94
10 days	1	99.83 ±0.34	97.62 ±1.39	97.26 ±3.00	98.08 ±1.72	97.60 ±1.44
	60	97.44 ±2.72	92.24 ±5.44	90.86 ±7.22	94.04 ±6.11	92.02 ±5.57

TABLE C.1: Different metrics for the evaluation of the LSTM-ALL model using different time ranges and number of snapshots (S).

Detection Deadline	S	AUC-ROC	F1-score	Precision	Recall	Accuracy
1 min	1	85.27 ±4.33	76.58 ±4.76	78.46 ±7.61	75.38 ±5.56	76.83 ±5.55
5 min	1	85.64 ±4.90	76.37 ±5.33	76.09 ±4.56	77.12 ±8.40	76.35 ±4.79
15 min	1	88.50 ±4.25	78.17 ±7.02	80.97 ±5.88	76.15 ±9.59	79.04 ±5.64
	3	89.96 ± 5.71	80.81 ± 7.24	82.78 ± 6.74	79.81 ± 10.92	81.35 ± 6.45
	15	91.11 ± 5.12	82.28 ± 4.35	79.63 ± 4.59	85.38 ± 5.97	81.63 ± 4.50
1 hour	1	89.55 ±3.03	79.99 ±4.39	82.23 ±4.17	78.27 ±7.40	80.58 ±3.79
4 hour	1	89.70 ±3.08	80.69 ±4.38	84.72 ±7.16	77.50 ±5.71	81.44 ±4.37
	4	94.31 ± 2.97	85.67 ± 4.12	87.97 ± 6.13	83.85 ± 5.39	85.96 ± 4.17
	16	93.21 ±2.85	83.16 ±5.08	85.77 ±6.28	81.73 ±92.33	83.65 ±4.30
	48	89.16 ±4.90	79.59 ±4.79	77.14 ±8.40	83.46 ±7.10	78.17 ±7.52
1 day	1	93.45 ± 3.82	85.84 ± 5.41	85.58 ± 5.39	86.73 ± 8.83	85.87 ± 4.87
3 day	1	93.47 ±4.30	86.33 ±5.48	88.29 ±4.28	84.62 ±7.25	86.73 ±4.97
10 days	1	94.84 ± 3.37	87.50 ± 5.54	90.31 ± 7.29	85.19 ± 6.08	87.79 ± 5.64
	60	94.21 ± 3.31	86.79 ± 4.56	86.73 ± 4.83	87.31 ± 7.51	86.83 ± 4.15

TABLE C.2: Different metrics for the evaluation of the LSTM-N model using different time ranges and number of snapshots (S).

Detection Deadline	S	AUC-ROC	F1-score	Precision	Recall	Accuracy
1 min	1	96.61 ± 2.25	89.94 ± 4.48	89.26 ± 5.22	91.35 ± 8.48	89.90 ± 4.35
5 min	1	96.76 ± 2.49	90.44 ± 4.43	88.58 ± 7.01	92.88 ± 5.71	90.10 ± 4.67
15 min	1	96.84 ±1.89	90.46 ±3.57	89.72 ±3.94	91.35 ±4.73	90.38 ±3.57
	3	98.55 ± 1.23	94.16 ± 3.61	92.72 ± 5.14	95.96 ± 5.12	94.04 ± 3.64
	15	97.97 ± 1.79	91.84 ± 4.09	90.48 ± 6.38	93.85 ± 6.42	91.63 ± 4.26
1 hour	1	97.80 ±2.02	91.54 ±4.12	89.59 ±6.60	94.04 ±5.54	91.25 ±4.29
4 hour	1	98.94 ±0.97	94.15 ±2.84	93.61 ±5.47	95.00 ±3.57	94.04 ±3.07
	4	99.12 ± 0.82	94.47 ± 3.48	95.71 ± 3.52	93.65 ± 6.78	94.62 ± 3.14
	16	98.90 ±1.02	93.59 ±3.68	93.80 ±4.34	93.85 ±6.87	93.65 ±3.47
	48	82.06 ±8.01	66.15 ±23.52	59.41 ±23.33	77.69 ±28.04	66.06 ±13.31
1 day	1	98.88 ±1.37	95.19 ±2.55	93.58 ±3.99	97.12 ±4.41	95.10 ±2.52
3 day	1	99.25 ±0.87	95.34 ±2.83	95.94 ±3.85	95.00 ±4.88	95.38 ±2.75
10 days	1	99.54 ±0.54	95.12 ±2.75	93.06 ±4.78	97.69 ±5.15	95.00 ±2.75
	60	73.88 ±12.52	75.87 ±9.17	64.88 ±11.52	94.23 ±13.16	69.62 ±11.62

TABLE C.3: Different metrics for the evaluation of the LSTM-L model using different time ranges and number of snapshots (S).

Detection Deadline	S	AUC-ROC	F1-score	Precision	Recall	Accuracy
1 min	1	92.24 ± 3.39	85.81 ± 4.22	85.87 ± 5.36	85.96 ± 5.02	85.77 ± 4.40
5 min	1	90.62 ± 4.39	83.06 ± 5.69	84.05 ± 5.90	82.31 ± 6.98	83.27 ± 5.53
15 min	1	91.88 ±3.43	83.70 ±4.17	83.68 ±4.99	83.85 ±4.40	83.65 ±4.32
	3	92.97 ± 3.09	84.75 ± 4.32	86.40 ± 3.95	83.46 ± 6.73	85.10 ± 3.95
	15	92.54 ± 3.23	84.76 ± 2.66	84.36 ± 4.21	85.38 ± 3.77	84.62 ± 3.01
1 hour	1	93.11 ±4.15	84.59 ±4.51	83.20 ±6.16	86.35 ±5.47	84.23 ±4.75
4 hour	1	95.66 ±1.84	89.18 ±3.31	89.41 ±3.02	89.04 ±4.55	89.23 ±3.18
	4	95.71 ± 2.67	87.99 ± 3.45	90.01 ± 3.44	86.15 ± 4.62	88.27 ± 3.35
	16	96.22 ±2.14	88.37 ±4.54	89.20 ±5.25	87.88 ±6.02	88.46 ±4.39
	48	95.98 ±2.63	89.08 ±3.65	89.75 ±6.57	88.85 ±4.54	89.04 ±3.92
1 day	1	96.99 ±2.06	91.90 ±3.19	92.26 ±4.09	91.73 ±4.64	91.92 ±3.17
3 day	1	97.01 ±2.16	92.04 ±3.50	91.68 ±3.67	92.50 ±4.51	92.02 ±3.50
10 days	1	97.02 ±1.75	92.07 ±3.08	91.40 ±3.49	92.88 ±4.39	92.02 ±3.10
	60	91.78 ±4.75	82.34 ±7.19	82.02 ±12.28	84.81±9.44	81.15 ±10.26

TABLE C.4: Different metrics for the evaluation of the LSTM-H model using different time ranges and number of snapshots (S).

Detection Deadline	S	AUC-ROC	F1-score	Precision	Recall	Accuracy
1 min	1	95.92 ± 2.46	87.64 ± 4.42	87.95 ± 7.41	88.65 ± 9.71	87.59 ± 4.26
5 min	1	96.56 ± 1.92	88.70 ± 3.29	89.40 ± 6.61	89.04 ± 8.02	88.75 ± 2.95
15 min	1	96.04 ± 1.79	88.53 ± 2.87	85.19 ± 6.25	92.69 ± 4.37	87.88 ± 3.45
	3	96.65 ±1.82	88.29 ±3.08	87.99 ±7.15	90.00 ±9.18	88.17 ±2.95
	15	95.20 ± 1.70	87.79 ± 2.27	83.20 ± 5.33	93.46 ± 4.57	86.92 ± 2.92
1 hour	1	97.27 ±1.46	88.81 ±4.03	88.77 ±5.65	89.81 ±8.90	88.85 ±3.45
4 hour	1	98.07 ±1.02	92.84 ±2.51	91.56 ±3.39	94.62 ±6.37	92.79 ±2.25
	4	98.19 ± 0.93	91.03 ± 2.58	89.13 ± 6.90	93.85 ± 5.82	90.67 ± 3.01
	16	93.23 ±3.16	81.99 ±8.98	73.92 ±15.41	95.58 ±5.84	77.12 ±14.49
	48	91.60 ±4.38	75.50 ±26.26	70.32 ±27.01	84.04 ±28.43	77.40 ±15.71
1 day	1	99.03 ±0.73	93.76 ±3.83	94.12 ±4.92	94.04 ±7.57	93.85 ±3.47
3 day	1	98.70 ±0.78	94.12 ±2.18	91.99 ±3.93	96.54 ±3.30	93.94 ±2.32
10 days	1	99.34 ±0.48	95.32 ±1.82	96.32 ±3.26	94.62 ±4.70	95.38 ±1.71
	60	89.35 ±7.97	75.15 ±26.18	71.54 ±26.57	81.35 ±28.94	78.37 ±13.95

TABLE C.5: Different metrics for the evaluation of the LSTM-T model using different time ranges and number of snapshots (S).