

Automated Detection of Misinformation Sources on Social Media

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Valentin Rupprecht, BSc

Matrikelnummer 01525090

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Allan Hanbury
Mitwirkung: Univ.Prof. Dr. David Garcia Becerra
Dr. Jana Lasser
Dr. Hannah Metzler

Wien, 17. April 2022



Valentin Rupprecht



Allan Hanbury



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Automated Detection of Misinformation Sources on Social Media

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Valentin Rupprecht, BSc

Registration Number 01525090

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Allan Hanbury

Assistance: Univ.Prof. Dr. David Garcia Becerra

Dr. Jana Lasser

Dr. Hannah Metzler

Vienna, 17th April, 2022



Valentin Rupprecht



Allan Hanbury



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Valentin Rupprecht, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 17. April 2022



Valentin Rupprecht



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Zuerst möchte ich meinem Betreuer David Garcia für seine großartige Unterstützung danken. Du hast mein Interesse für Social Data Science geweckt und mich im Complexity Science Hub (CSH) willkommen geheißen. Das Wissen dich als meinen Betreuer zu haben und in einer relevanten und spannenden Disziplin zu forschen waren ausschlaggebend für meine Masterarbeit am CSH.

Besonders möchte ich auch Jana, Hannah und Max für ihre ständige Unterstützung und Feedback zu meinen Fortschritten danken. Ich schätze es sehr, dass ich euch immer um Rat fragen konnte, egal bei welchen Problemen. Ihr habt mir auch Materialien und technische Mittel zur Verfügung gestellt, ohne die meine Arbeit nicht möglich gewesen wäre.

Ich bin auch meinem Betreuer Allan Hanbury für seine Unterstützung sehr dankbar. Die Kommunikation mit dir war immer unkompliziert und du hast mir schnell hilfreiches Feedback gegeben. Vielen Dank auch an Alina, Anna und Segun für ihre Zeit und das hilfreiche Feedback. Ich schätze eure Bemühungen sehr.

Diese Diplomarbeit wäre auch nicht ohne die großartige Unterstützung von Freunden und Familie möglich gewesen. Ich bin dankbar, dass ich den Jack als Mitbewohner hatte. Mit dir haben selbst die harten Zeiten des Studiums viel Spaß gemacht. Ich will auch meinen besten Freunden in Wien, Fabi, Max, Scholzi und Sebi, danken. Ihr habt mir eine großartige Zeit während und außerhalb des Studiums beschert. Ich bin auch dem Olli, Raphael und Gabriel dankbar, dass wir gemeinsam die spannenden Seiten von Data Science entdeckt haben. Dank euch haben alle Vorlesungen und Projekte viel Spaß gemacht.

Zuletzt möchte ich meinen Eltern, Elke und Leo, sowie meiner Schwester Emily danken, dass sie mich die ganzen Jahre so gut unterstützt haben. Ihr habt mir mein Studium überhaupt ermöglicht. Ich bin auch der Susi sehr dankbar. Deine Meinung ist mir am wichtigsten.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

First, I want to thank my supervisor David Garcia for his great support and advice. You sparked my interest in Social Data Science and made me feel welcome at the Complexity Science Hub (CSH). Being able to count on your advice and researching in a relevant and exciting field were decisive reasons for my master thesis at the CSH.

Special thanks also to Jana, Hannah, and Max for their continuous reinforcement and feedback on my progress. I appreciate that I could always ask for your advice, regardless of the matter. You also provided me with material and computational resources, without which my thesis would not have been possible.

I am also grateful for the support of my advisor Allan Hanbury. The communication with you was always straightforward and you provided me with quick and helpful feedback. Many thanks also to Alina, Anna, and Segun for their time and valuable feedback. I really appreciate your effort.

I would have been unable to finish this master thesis without the support from my friends and family. I am grateful that I had Jack as my roommate. With you even the hard times of studying were a lot of fun. I also want to thank my best friends in Vienna, Fabi, Max, Scholzi, and Sebi. You guys gave me a great time during and beyond my studies. I am also thankful to Olli, Raphael, and Gabriel for discovering the exciting parts of Data Science together. Thanks to you, lectures and projects were always fun.

Finally, I want to thank my parents, Elke and Leo, and my sister Emily for their continuous support all these years. You made my studies possible in the first place. I am also very grateful to Susi. Your opinion is most important to me.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Falschinformationen und Fake News stellen eine große Herausforderung für unsere Gesellschaft dar. Diese haben die öffentliche Meinung gegenüber der COVID-19-Pandemie, der US-Präsidentenwahl 2016 oder der russischen Invasion in die Ukraine beeinflusst. Soziale Medien spielen als ein beliebtes Nachrichtenmedium eine wichtige Rolle bei der Verbreitung von Falschinformationen und erreichen somit eine große Anzahl an Menschen.

Nachrichtenartikel, die in sozialen Medien geteilt werden, werden zuvor in der Regel auf einer Website veröffentlicht, sogenannte Nachrichtenquellen. Da einige Nachrichtenquellen regelmäßig Fake News veröffentlichen, haben Experten begonnen die Vertrauenswürdigkeit von Nachrichtenquellen zu bewerten, indem sie deren Faktentreue, politische Einstellungen, Transparenz oder Sympathie für Verschwörungstheorien überprüfen. Dazu ist eine gründliche Analyse der Nachrichtenquelle und ihrer veröffentlichten Artikel erforderlich, was Zeit in Anspruch nimmt und somit nur einen Teil der tatsächlich existierenden Nachrichtenquellen abdecken kann, was vor allem Englischsprachige sind. Daher müssen automatisierte Lösungen erforscht werden, die in der Lage sind die große Zahl der vorhandenen Nachrichtenquellen zu beurteilen.

Obwohl es viel Forschung zur automatischen Erkennung von Fake News Artikeln gibt, existieren bisher wenig Lösungsansätze, die die Glaubwürdigkeit von Nachrichtenquellen klassifizieren. Da in den bisherigen Ansätzen Informationen aus sozialen Medien kaum berücksichtigt wurden, stellen wir ein auf maschinellem Lernen basierendes System vor, das die Glaubwürdigkeit einer Nachrichtenquelle anhand von Informationen aus Beiträgen in sozialen Medien bewertet, die einen Artikel dieser Quelle geteilt haben. Genauer gesagt sammeln wir Beiträge und Benutzerinformationen von Twitter und Facebook sowie die Artikel, die sie geteilt haben, und extrahieren Textmerkmale aus den Artikeln, den Beiträgen und der Selbstbeschreibungen der Nutzer mit Hilfe des vortrainierten Sprachmodells RoBERTa. Nachdem wir diese Informationen mit Metadaten von Nutzern und Beiträgen kombiniert haben, erhalten wir mit Hilfe eines LSTMs einen repräsentativen Vektor für die Nachrichtenquelle, welchen wir als Grundlage für die Klassifizierung verwenden. Die Auswertung unseres Modells zeigt, dass wir den Stand der Technik nicht verbessern konnten und die meisten der von uns erkannten Nachrichtenquellen für Falschinformation in Wirklichkeit korrekte Nachrichten veröffentlichten. Allerdings zeigte unser Modell eine vielversprechende Genauigkeit von fast 92% bei der Erkennung korrekter Nachrichtenquellen.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Misinformation and fake news pose major challenges for our society. False information has influenced public opinion regarding the COVID-19 pandemic, the US presidential election in 2016, or the Russian invasion of Ukraine. As a popular news medium, social media plays an important role in spreading false information and reaching a large audience.

News articles shared on social media are usually published on a website in the first place, so-called news sources. Since some news sources regularly publish fake news stories, human experts have started to assess the reliability of news sources by classifying their factual fidelity, political bias, transparency, or sympathy towards conspiracy theories. To do so, a thorough analysis of a news source and its published articles is necessary, which takes time and therefore can only cover a subset of news sources, primarily English speaking. Thus, automated approaches need to be researched, which are able to handle the vast number of existing news sources.

Although a lot of research regarding the automated detection of fake news articles exists, not much has been done yet in terms of the credibility classification of news sources. As existing work lacks information available on social media, we propose a framework based on Machine Learning that classifies the reliability of a news source by leveraging information from social media posts that shared an article of the given source. More specifically, we collect posts and user information from Twitter and Facebook as well as the articles referenced in the posts, and extract textual features from the articles, the posts, and the self-descriptions of users applying the pre-trained language model RoBERTa. After combining these features with metadata from users and posts, we aggregate them into a representational vector for the news source using an LSTM, which will be the input of our classifier. The evaluation of our model shows that we could not improve state-of-the-art approaches and most of our detected misinformation sources are actually publishing accurate news. However, our model showed a promising precision of almost 92% when detecting accurate news sources.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung	xi
Abstract	xiii
Contents	xv
1 Introduction	1
1.1 Problem Statement	2
1.2 Contributions	3
1.3 Thesis Overview	3
2 Preliminaries of Natural Language Processing	5
2.1 Transformers	5
2.2 Pre-trained Language Models	7
3 Related Work	11
3.1 Definition of False Information	11
3.2 Input Features	12
3.3 Categorization of Detection Approaches	13
3.4 Detecting False Information Articles	14
3.5 Detecting False Information Sources	19
3.6 Data Labeling	20
4 Data Acquisition and Labeling	23
4.1 Unification of News Source Ratings	23
4.2 Label Mapping Scheme	25
4.3 Collecting Data	27
5 Post Sequence Model	33
5.1 Model Overview	33
5.2 Feature Selection	34
5.3 Feature Embedding	35
5.4 Final Model	38
	xv

6	Model Performance	41
6.1	Parameter Tuning	41
6.2	Performance Validations	46
6.3	Additional Applications	50
7	Discussion and Conclusion	55
7.1	Limitations and Future Work	55
7.2	Summary	57
A	Fact-Checking Websites	59
A.1	Bufalopedia	59
A.2	Bufale	60
A.3	Butac	61
A.4	Buzzfeed News	62
A.5	Politifact	62
A.6	Columbia Journalism Review	63
A.7	Fake News Watch	63
A.8	Media Bias Fact Check	64
A.9	Melissa Zimdars – Washington Post	66
A.10	Newsguard	68
B	Supplementary Material	71
B.1	Additional Figures for Model Implementation	71
B.2	Evaluation Metrics	72
B.3	Permutation Test and Bootstrapping	72
B.4	Additional Figures for Performance Validation	73
	List of Figures	75
	List of Tables	79
	Bibliography	81

Introduction

Social media platforms play an important role as an information source and in daily news consumption. In 2021 about 48% of US adults consumed news on social media and about half of the users on Facebook and Twitter do it regularly [13]. Furthermore, more than half of the Tweets in 2021 published by US adults are about entertainment, government, or politics, but also health and international topics have become more popular [12]. Social media also enables users to share news articles published on external news sources with their followers and friends. This feature also aids the propagation of false information, which often attracts attention with misleading claims in the title, but comes with shorter, simpler article bodies containing repetitive information without argumentation [35]. That makes them easy to read and quick to share. Compared to true news, fake news is much more likely to be shared on social media, which spreads six times faster to reach the same number of people [87].

Recent events in the past years have also shown that false information on social media can cause real-world effects. An early example of a large-scale influence of false information on social media was the US presidential election in 2016. For instance, the 20 most popular fake news stories received almost 20% more engagements on Facebook than the 20 most popular election stories coming from major news websites [52]. In addition, people were more often exposed to articles coming from fake news sources in the weeks before the election day than afterwards, affecting people leaning to Republican partisanship more than people having a Democratic affinity [30]. A large appearance of false information distorts facts also during the COVID-19 pandemic, driving the perception of a part of society, also called “infodemic” [27]. Frequent topics covered by the infodemic are miracle cures, conspiracies about the provenance of the virus, or denying the risks caused by the virus [23]. Also false information about vaccines is frequently shared, which can cause mistrust or denial against vaccination [45]. Finally, the recent Russian invasion of Ukraine has started an information war between both countries also using false information [79,80].

Again social media plays a big role, where the public is deceived with false advances and footage about the war or agitated against Ukrainian refugees [31, 67].

1.1 Problem Statement

This significant influence false information can have on society makes it necessary to limit the spread of false information by informing users on social media about the reliability and factuality of news. An important indicator that journalists use when they check the truthfulness of some piece of information is the reliability of the source, since a source, which has already published false information, is likely to do it again [4]. Assessing the reliability at the news source level also saves a lot of time, compared to checking each news article separately. For this reason, several fact-checking websites have been established that estimate the reliability, factual fidelity, or political partisanship of news sources, such as Media Bias Fact Check [15] or NewsGuard [53]. The reliability classification of a news article using the reliability of its source is also much faster if the source is known. Debunking fake news at an early stage is very important to mitigate or even prevent the spread of false information on social media [94].

Since a manual approach assessing fake news articles is not scalable to the vast number of articles published, but an early detection of false information is desired, researchers have also proposed automated approaches based on Machine Learning, which detect fake news articles. However, little to no research has been done on automated reliability classification of news sources, although it would have some advantages. First, manual assessments of news sources by fact checkers get outdated quickly, e.g. because a news website was shut down or novel news sources have been established, which have not been assessed yet by human fact-checking experts. Secondly, well-established fact-checking experts mainly assess news sources from large countries speaking popular languages, like English or French. However, false information can be an issue in any country, where news is usually not written in English or no manual fact-checking is done. Therefore, an automated approach detecting false news sources independently of their language and at the time they appear could be beneficial for much more people.

Possible applications could be that people use the framework as an optional plugin on social media that flags news from false information sources or by researchers from various fields working with news data to make statements about the subset of fake or misinforming news. These use cases require a model to detect fake news sources with high precision, such that it does not misclassify accurate news sources as fake news sources.

Existing research regarding the automated reliability classification of news sources primarily focused on the published content of the news sources and did not include information extracted from social media posts or their users [4, 5]. Opposed to that, existing approaches detecting false news articles also consider information extracted from social media, like post text, comments, user features, or following relationships [19, 71, 72, 93]. We argue that social media is an essential driver of spreading false information. It therefore remains an open problem in research to automatically classify

the credibility of a news source by leveraging features extracted from the content of its published articles, social media posts, and users spreading those articles. Therefore I will contribute to this research gap by answering the following research questions:

- What is an appropriate means to unify manual news source reliability ratings from various fact-checking organizations in order to collect a large and diverse dataset?
- What is an appropriate means of implementing a model predicting the reliability of a news website by using content and social context features of its articles?

1.2 Contributions

The main contributions of the thesis are the following:

- We collect various news source reliability ratings, which were manually compiled by a fact-checking organization, each using different categories or methods. We then unify those ratings onto a common scale to use them as a large and diverse ground-truth dataset.
- We collect social media posts and user data from Facebook and Twitter, which have shared an article published by a prior labeled news source. We combine the data to yield a single dataset containing posts and users from different social platforms.
- We implement a Machine Learning model that predicts the reliability of a news source¹. To this end, we link each news source to its articles, the posts that shared them, and the posting users. We extract features from all these entities and combine them into a single representational vector, which can then be used to predict the target label.

1.3 Thesis Overview

The thesis is structured as the following: In Chapter 2 we clarify preliminary knowledge regarding Natural Language Processing and neural language models. In Chapter 3 we summarize related work in the field of false information detection. We will focus mainly on possible features that can be extracted and show how they have been applied to detect fake news articles. We then present existing work approaching the reliability classification of news sources manually and automatically. In Chapter 4 we give details on how we relabeled existing news source ratings from different fact-checking experts onto a common scale. Further, we explain the collection of data, including news articles, social media posts, and their users, and explain which features we use. After that, in Chapter 5 we propose the architecture of our framework. We show how we extract features from

¹The labeled news sources and source code are available at https://github.com/vrupp/misinformation_sources

textual information and how we compile a single news source representation vector. In Chapter 6 we evaluate our model. We first summarize our findings after using different model configurations. After that, we report and validate our final results. In Chapter 7 we discuss our findings, illustrate possible limitations of our approach, which can be improved in future work, and end the thesis with a summary of our work. In Appendix A and B we provide supplementary material, e.g. details on the relabeling of news source ratings or additional tables and figures supporting our observations.

Preliminaries of Natural Language Processing

In this chapter we introduce preliminary knowledge regarding the field of Natural Language Processing (NLP), which is used in this work. In Section 2.1 we explain the basic concepts of the Transformer architecture and in Section 2.2 we present pre-trained language models using Transformers, which are used to solve many NLP tasks.

2.1 Transformers

Transformers are a neural architecture originally designed for machine translation, but now used in many language understanding tasks. Transformers map a sequence of word embeddings (x_1, \dots, x_n) into a sequence of output vectors (y_1, \dots, y_n) of the same length. Figure 2.1 shows the components of a single Transformer block. It consists of different layer components, like a standard feedforward layer, residual connections, which add the output of a layer to its bypassed input, or normalization layers, which normalize the input and pass it to a standard feedforward function. However, its core feature is the self-attention layer, which is responsible for extracting context information for each word in the sequence. The self-attention mechanism uses the dot product to compare words in a sequence with each other in order to derive a word's relevance. Each word takes three different roles in an attention process:

Query The word is compared to all of the other preceding words in the sequence.

Key The word is one of the preceding words of a query word, which it is compared to.

Value The word is used as value to compute the current focus of attention.

For each of these roles the self-attention layer contains a trainable weight matrix $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$, where d is the dimension of each word vector, i.e. $x_i \in \mathbb{R}^{1 \times d}$. When computing the attention for a word vector x_i , we need to compute the similarity to each preceding word x_j in the sequence, $j \leq i$, as

$$\text{score}(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

where $q_i = W^Q x_i$ and $k_j = W^K x_j$, both having dimension d_k . When using just a single self-attention layer, d_k is equal to d , but when using multihead attention, d_k can be different to d , as shown later. In addition to scale the score using the dimension d_k , it is further normalized with respect to the other words preceding a given word x_i by using the softmax function. These normalized scores are then used as weights for the own and preceding value vectors to compute the self-attention c_i of the input word x_i :

$$\begin{aligned} \alpha_{ij} &= \text{softmax}(\text{score}(x_i, x_j)), & \forall j \leq i \\ &= \frac{\exp(\text{score}(x_i, x_j))}{\sum_{k=1}^i \exp(\text{score}(x_i, x_k))} \\ c_i &= \sum_{j \leq i} \alpha_{ij} v_j, & v_i = W^V x_i \end{aligned}$$

These computations can be optimized into a sequence of matrix multiplications, such that we can compute the self-attention of all words x_1, \dots, x_n at once, as further explained in [40, Sec. 9.7].

The self-attention mechanism can be improved further to capture more relations between the words in a sequence. One way is the bidirectional self-attention, which computes the attention dependent on the entire input sequence, instead of only considering preceding words. This changes the computation of the softmax values and attention outputs as follows:

$$\begin{aligned} \alpha_{ij} &= \frac{\exp(\text{score}(x_i, x_j))}{\sum_{k=1}^n \exp(\text{score}(x_i, x_k))} \\ c_i &= \sum_{j=i}^n \alpha_{ij} v_j \end{aligned}$$

Another option that can be well combined to uni- or bidirectional self-attention is multi-head attention. This improvement allows to compute multiple attentions simultaneously, i.e. attention layer i , called head, uses its own set of parameters W_i^K, W_i^Q, W_i^V . Furthermore, the weight matrices can have a different dimension than the word embedding dimension d , where the weights of query and key share the same dimension d_k and the weights for value have dimension d_v , i.e. $W_i^K, W_i^Q \in \mathbb{R}^{d \times d_k}$ and $W_i^V \in \mathbb{R}^{d \times d_v}$. The outputs from each head are concatenated and projected back to the desired output dimension d by using another weight matrix $W^O \in \mathbb{R}^{hd_v \times d}$, where h is the number of heads. Figure 2.2 shows an example multihead attention layer using 4 heads [40, Sec. 9.7, 11.1].

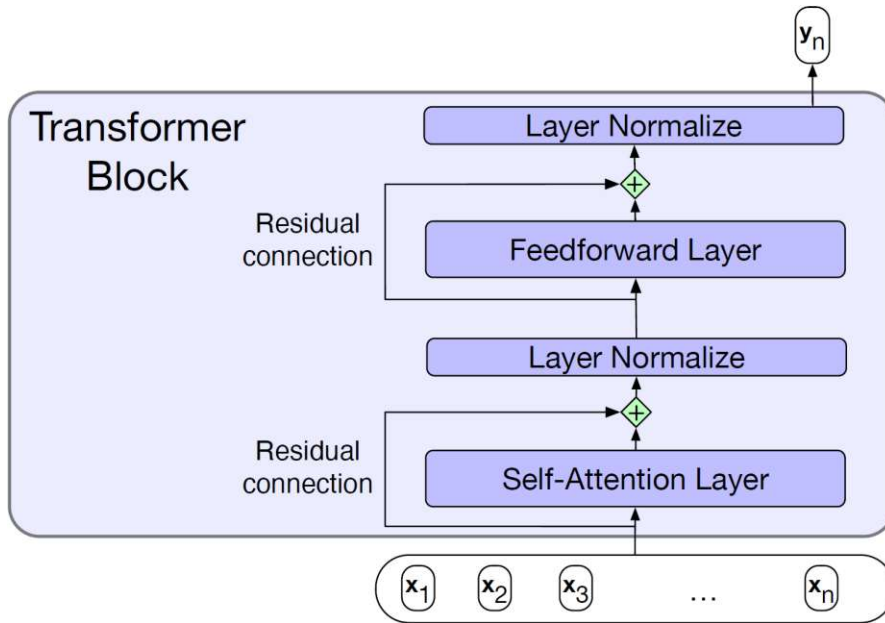


Figure 2.1: Transformer Block mapping embedding x_n to y_n using its preceding words [40, Sec. 9.7]

In order to also consider the position of a given word, Transformers combine the word embedding with a positional encoding. One simple way is to use absolute position embeddings, i.e. a learnable embedding vector for each position, which then is combined with the word embedding. The problem with this approach is that during training lower positions are much often updated than latter ones. This is solved by using a relative positional embedding computed by a static function, such that it reflects the closeness of two words being next to each other in a sequence. Let $PE \in \mathbb{R}^{n_{\max} \times d}$ be a matrix containing the positional embeddings in a rowwise manner, where n_{\max} is the maximum possible sequence length. The matrix is computed as follows:

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10,000^{2i/d}}\right)$$

$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10,000^{2i/d}}\right)$$

where pos is the position of the word and $1 \leq i \leq (d/2 - 1)$. Since the positional encoding has the same dimension d as the input word embedding, we can add both before forwarding it to the Transformer block [40, Sec. 9.7] [54, 83].

2.2 Pre-trained Language Models

As previously mentioned, Transformers were formerly developed for machine translation tasks, but are nowadays used for many NLP tasks, which are based on language modeling.

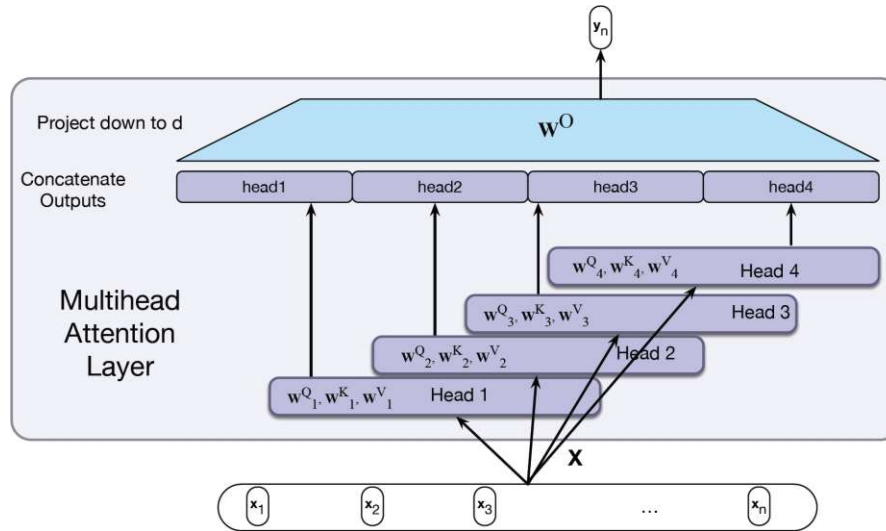


Figure 2.2: Multihead attention layer using 4 heads [40, Sec. 9.7]

In general, language modeling describes the task of language understanding, which is expressed as the assignment of a probability to a sequence of words, i.e. $P(w_1, \dots, w_n)$. This probability can be decomposed into the product of probabilities of each word using the chain rule: $P(w_1)P(w_2 | w_1) \dots P(w_n | w_{n-1}, \dots, w_1)$. Training a language model is a very general task, where one only needs a large text corpus, from which the model can learn statistics or infer relations between word tokens. A well trained language model can be used for more specialized tasks, like machine translation, text generation, or speech recognition. Transformers have proven to perform very well as a language model, which is why almost every pre-trained language model is now based on Transformers. To apply Transformers to the language modeling task, we need to add an additional layer to the output of the Transformer block, which projects the output into a vector of dimension equal to the size of the vocabulary, i.e. the number of possible words. By using the Softmax activation function this vector transforms into a probability distribution over the vocabulary. One can train this language model using a given text sequence by predicting the next word using all preceding words. The loss is then calculated over the whole sequence by averaging the individual losses over all words in the sequence [16, Ch. 5] [40, Ch. 3, Sec. 9.8] [44].

One of the first language models basing on Transformers that lead to tremendous progress in language understanding [44, 49, 96] is BERT [22]. BERT consists of 12 bidirectional Transformer blocks, each using 12 self-attention heads, and uses a dimension of $d = 768$. Its architecture is very general, such that it can be trained on many token- and sentence-level prediction tasks. For this it needs a specially formatted input and pre-training on a large text corpus. Each input sequence is tokenized and embedded using the existing WordPiece embedding [92], and extended by two additional tokens. A [CLS] token is added to the beginning of each sequence and its attention vector after 12 multihead

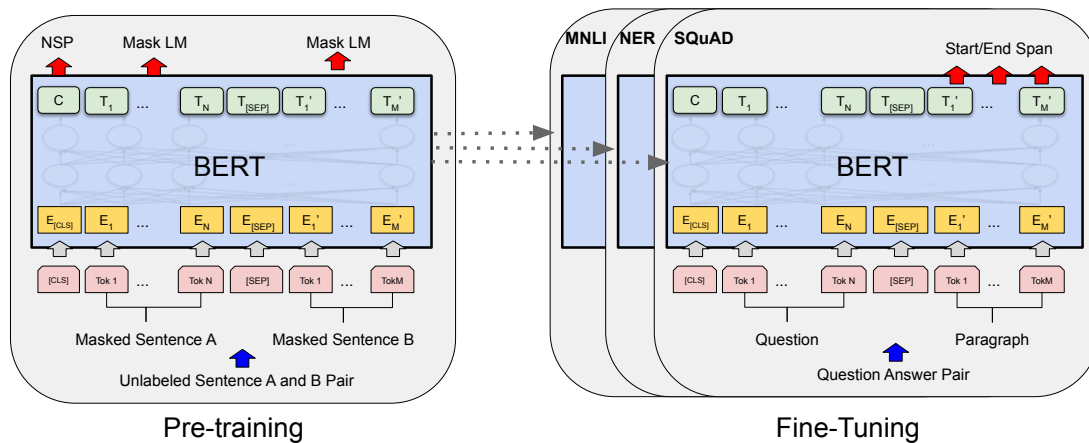


Figure 2.3: BERT Architecture during pre-training and fine-tuning

Transformer blocks is used as a representation vector for sequence classification. In case the input is a sequence pair, e.g. for question-answering tasks, both sequences A and B are concatenated to a single sequence separated by a [SEP] token. The final input embedding for BERT consists of the existing token embeddings and two additional embeddings, one positional and a second one indicating, whether the token belongs to sequence A or B, if existing.

The model is pre-trained using two tasks: Masked language modeling (MLM) and next sentence prediction (NSP). For both tasks the authors used document-level text corpora from the English Wikipedia and the BooksCorpus [97]. During the MLM task a certain percentage of tokens in a sequence are masked by replacing it with a special [MASK] token. The final attention vector of this masking token is then fed to a Softmax layer, which yields a probability distribution over the whole vocabulary. The second task NSP is needed for BERT to understand sentence relationships. For this a binary classification task is defined using the final attention vector of the [CLS] token as input to predict, whether sequence B truly follows sequence A. Although the task asks for predicting the next sentence, sequence A and B can consist of multiple sentences or parts of a single sentence. The classifier is trained using pairs of sequences from the text corpora, where 50% of the time sequence B actually follows sequence A and otherwise both sequences are randomly selected from the corpora. After the pre-training phase, BERT can be fine-tuned to any token- or sequence classification task by using the output attention vectors. For instance, similar to the NSP task, the output vector of the [CLS] token can be fed into an additional fully connected feed-forward layer including a Softmax function to perform any classification task, like sentiment analysis. Figure 2.3 shows the architecture of BERT and how it is used during the pre-training and fine-tuning phase.

As already mentioned, BERT was a major breakthrough in language modeling and therefore motivated researchers to build on BERT's technologies and further improve the idea. Figure 2.4 shows some of the related approaches to BERT and how they are

different from it. One of these improvements is RoBERTa [43]. RoBERTa shares the same architecture as BERT, but comes with an improved training procedure: First, RoBERTa was trained longer, with larger batch sizes, and more data. In addition to the text corpora BERT was trained on, RoBERTa was trained on English news articles and text extracted from websites shared on `reddit.com`. The raw size of the dataset is about 10 times as large as the subset BERT was trained on. Secondly, RoBERTa is not pre-trained using the NSP task. Thirdly, the model is trained on longer sequences than BERT. While the input for BERT always consists of a pair of sequences, the authors found that training with full contiguous natural sentences increases the performance of downstream tasks, like question-answering. Finally, RoBERTa improved the masking for the language modeling task. BERT already masks the data during preprocessing, which means that a sequence keeps the same mask for each training epoch. RoBERTa instead proposes dynamic masking, where each sequence is randomly masked before being fed to the model. Thus, in each epoch the same sequence could be masked differently, which also leads to better performance on downstream tasks.

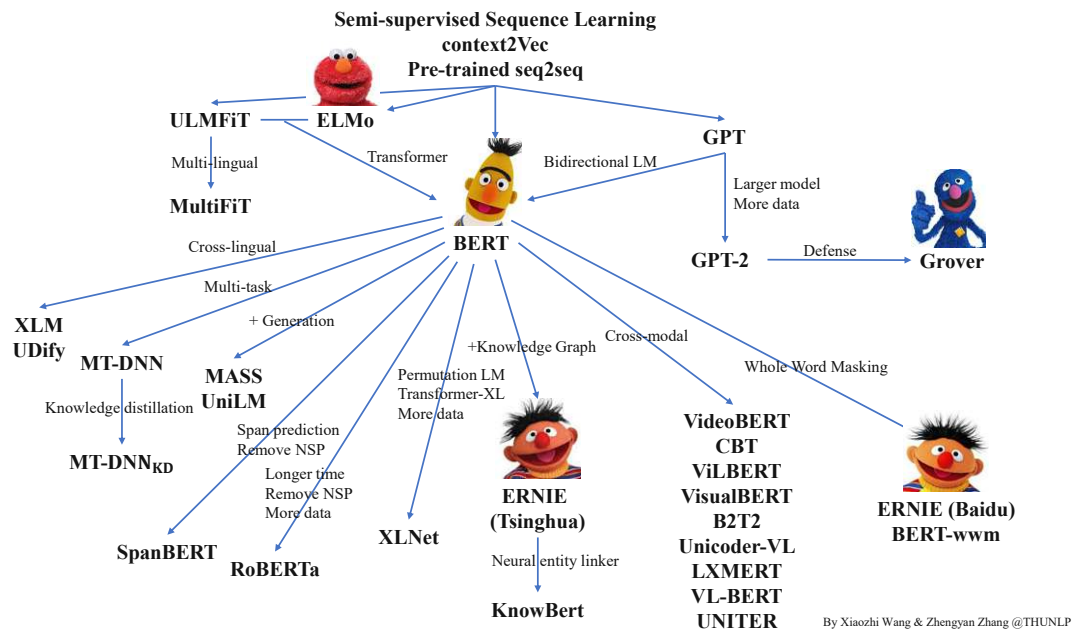


Figure 2.4: Family of pre-trained language models and their relation to BERT in 2019 [90]

Related Work

False information detection is a large research field with different perspectives, focuses, and approaches. This chapter summarizes existing work to give an overview of advances regarding false information detection. First, we give different explanations of false information and fake news. We then distinguish between several detection approaches and what types of features they are using. After that we focus on two main issues, the classification regarding false information of news articles and news sources, for which we outline existing approaches. Finally, we show different data collection and labeling techniques and compare their pros and cons.

3.1 Definition of False Information

Many different viewpoints and characterizations of false information exist. The most famous expression related to false information is *fake news*. This term is widely used for identifying false information on social media and arose during the U.S. Presidential Election in 2016 [7, 30, 42]. Naturally, fake news not only covers social media, but also traditional media channels. The term fake news is often defined as intentional and verifiable false information [7, 30, 42, 74, 95]. This definition consists of two general characteristics of false information, intention and verifiability. Fake news shares the basic concept of *disinformation*, which names intentional false information. As opposed to that, *misinformation* covers false news without the intention to deceive. A reason for misinformation can be the missing understanding or attention by the author [7, 42]. Thus, social media contains a lot of misinformation, since anyone can contribute to any topic, even though the user is not an expert. Also traditional media sources can share misinformation, e.g. by spreading views of celebrities which is considered newsworthy [23]. Apart from intention, false information can be classified based on its verifiability. False information can be verifiably false, when it contradicts, fabricates or conflates ground

truth information. Unverifiable false information can be expressed through an opinion, where no ground truth can be used to contradict a statement [7, 42].

Other aspects of false information rely on the type of intention or the potential harm it can cause. Bondielli et al. [7] introduce three types of intent of false information: *Serious fabrications*, where fake news are shared with malicious intent, *large scale hoaxes*, which produces false news by imitating trustful news, and *humorous fakes*, also known as *satire*, with the purpose to amuse the reader with fabricated stories. Similar to Bondielli, Zhou et al. [95] categorized existing concepts, like *cherry-picking*, *clickbait* and *rumor*, by their authenticity, intention, and whether they are news. Gallotti et al. [27] also generalized various different concepts of false information by mapping them to common scale of harm scores, which tells the potential contribution to manipulation and misinformation.

3.2 Input Features

When dealing with false information detection of news articles shared on social media multiple data sources can be exploited to support the decision process. Figure 3.1 gives an overview of possible features available. One group of features are news content features, which can be extracted solely from the news piece. This can include the source/author, headline, body text, and images or video [74]. One way for the extraction is using hand-crafted features. For instance, when composing features representing text, one can use syntactic features, like frequency of POS tags, lexical features, like term or n-gram frequencies, or semantic features, like sentiment or topic analysis. Another way of composing content features is extracting latent features by using deep learning mechanisms, like CNNs, RNNs, LSTMs, Transformer [7, 57, 95].

The second group are social context features, which are extracted from social media posts sharing a news article. These features represent information about the posting user, e.g. the number of followers or posts, the post itself, like stance or topic, and the network around the user and the post, which is built upon the following relationship or engagements. From the network one can extract graph features, like density or clustering coefficient, or diffusion patterns, for instance number and propagation time of retweets [7, 74].

Because of this diverseness of data sources, Shu et al. [74] characterized fake news detection on social media as unique challenge. On the one hand detecting fake news by only using content information is difficult, since it is supposed to mislead readers and mock true news. On the other hand context-based features, like user information and engagements, are very diverse and noisy, since fake news belong to "newly emerging, time-critical events". In an early stage of a post only few engagements or comments are available, which makes the false information detection less reliable [74, 75].

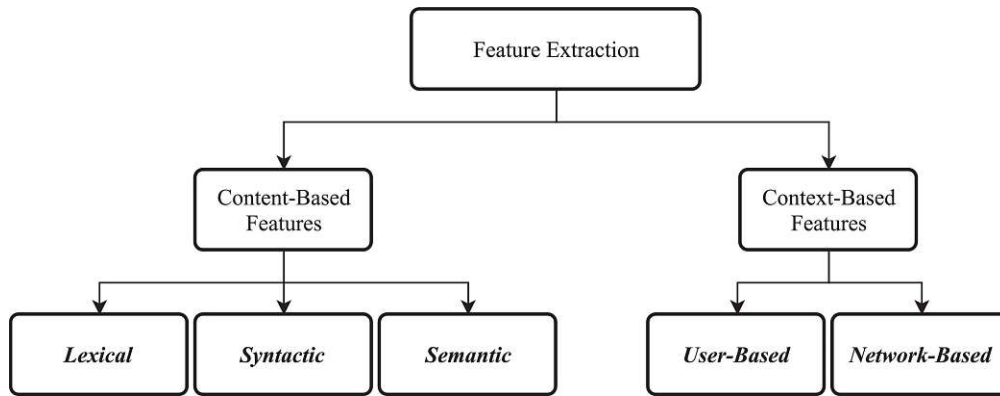


Figure 3.1: Available hand-crafted features for detecting false information [7]

3.3 Categorization of Detection Approaches

All forms of false information have basically three components: The content, the creator, and the publisher. With focus on social media one can identify a fourth entity, namely the propagating user. While the publisher represents the website that originally published the news article, the propagating user is responsible for sharing the article on social media platforms by posting its URL. For all four entities one can define a subtask, which aims for estimating its factuality or trustworthiness, e.g. detecting false information articles or unreliable publisher. Additionally, each subtask can be approached with the help of the other issues, e.g. a news article is unreliable, since it was shared mostly by unreliable social media users [95]. In the following sections we will only cover research detecting false news articles or rating the reliability of news sources, i.e. the publisher.

We begin with the most common issue, when dealing with false information, which is detecting false news articles. This problem is usually defined as a binary classification problem, deciding whether a news article is false information or not. It is also sometimes tackled as a multinomial classification problem, which assesses the credibility of an article (or similarly of a claim/a news source) in a more fine-grained manner [15, 47, 55]. In the literature, various detection methods exist, which can be categorized as follows [74, 95]:

Knowledge-based methods These approaches are also known as fact-checking and aim for validating claims from articles by trusted knowledge bases. This can be done manually by experts or crowd sourcing. Automatic techniques also exist, which extract facts from articles and validate them by a trusted knowledge base. The final label for an article is achieved by aggregating the validations of its claims.

Style-based methods Their goal is to extract style features from articles and assess the news intention for false information. This can be done by manually selecting features and classify them using Machine Learning or by using deep learning methods, which learn latent representations for text or images.

Stance-based methods Articles are classified according to user stances, which can be extracted from posts and comments belonging to a news piece. Those stances could be either explicit, e.g. via likes, or implicit, e.g. using text or network features.

Propagation-based methods Articles are classified as fake news according to their propagation in social networks. The propagation of an article can be expressed by a network, which are then used as input for classification models.

Source-based methods These methods classify news articles as false information based on the credibility of their sources. In this context, a source can represent a writer, publisher, or social media account spreading an article. For instance, an often used method for assessing the credibility of spreading accounts is detecting social bots, i.e. software scripts automatically spreading news articles.

When designing a false news detection approach, it does not have to fit a single method, but can also leverage a combination of multiple methods where possible, as done in so-called hybrid approaches [19, 66, 72].

A rather smaller research field is the assessment of reliability of news publishers. Existing approaches can be divided into manual and automatic assessments. Manual reliability rating can be conducted by a small group of experts or by crowdsourcing. Depending on the time, costs, and the expertise of the people checking news publisher, various methods and criteria are available, e.g. a review of the journalistic process, the publisher's financing or political interest, or its tendency to publish false information. Automatic reviews focus more on quantitative analyses by including various data sources belonging to a publisher and extracting useful data out of it. The data can be published articles, posts on social media, YouTube videos, etc. [4, 5].

3.4 Detecting False Information Articles

As already mentioned, false news detection on an article level is very well researched and much literature has been published regarding this issue. In the following we will outline state-of-the-art approaches for each category of detection methods mentioned in Section 3.3. In addition, we will also present existing hybrid approaches, which combine methods from multiple categories.

3.4.1 Knowledge-based Methods

An automated way of fact-checking has not existed for quite some time [95], but novel approaches were proposed recently. For instance Vo et al. [86] presented a neural network architecture with two attention layers capturing claim and document attention. Both layers consist of multiple heads to capture different semantic contributions that help fact-checking a claim. However, more solutions concerning manual fact-checking exist, for instance expert-based fact-checking sites like `FactCheck.org` [25] or `Snopes` [77]

debunking false information from various topics, recently especially concerning COVID-19 and vaccination. Also websites offering crowd-sources fact-checking, like Fiskkit [26], exist. Fiskkit enables users to read articles and annotate it on sentence level to help others identify true or false statements.

3.4.2 Style-based Methods

For style-based methods many more automatic approaches exist, using manually extracted features though. Ahmed et al. [2] extracted term frequencies from several n-grams and compared each feature set with six different supervised machine learning models. Pérez-Rosas et al. [60] also used hand-crafted text features to detect fake news, e.g. frequencies of unigrams, bigrams, and punctuation. They also incorporated readability features, like number of characters, number of complex words and long words. In the end they evaluated each feature set using an SVM classifier and cross-validation. Zhou et al. [94] proposed an even more extensive approach. Like Pérez-Rosas et al. they used features containing term frequencies and readability, but additionally they considered quality features, e.g. the diversity of terms used or the subjectivity of an article by counting biased words. Another attribute of their feature selection is the inclusion of clickbait-related attributes, of which one expresses the sensationalism of a news piece by measuring sentiment, punctuation usage, and similarity between headline and body text.

While those three approaches used hand-crafted content-based features, also latent content features were used in research, like in O'Brien et al. [56]. The authors use a pre-trained word2vec embedding, followed by a CNN layer with subsequent max-pooling layer to learn a new embedding, which is fed into a fully connected output layer. In their work they propose an explainable analysis of neural networks to identify words most relevant for the classification. Roy et al. [65] proposed an ensemble learner classifying statements by politicians into six reliability classes. The data consists of multiple attributes in textual form, e.g. the statement, the speaker's name and party etc., which is first embedded as pre-trained Google News vector and passed into a combination of a Bi-LSTM layer, a CNN with a max-pooling layer, and several dense layers. The outputs are first pairwise combined to reflect ten different relations, such as the one between speaker and party, and afterwards all together for a final single output. A very recent contribution using latent textual content features is FakeBERT [41]. The authors used BERT for sentence embedding, which is then fed into a CNN and max-pooling layer, connected to several dense layers computing the output prediction.

3.4.3 Stance-based Methods

Opposed to knowledge-based and style-based methods, detecting false information by investigating stances requires context-based features, like user comments or engagements. Since stances alone is a very little indicator for false information, most research focuses on combining stance features with other information, which is shown in Subsection 3.4.5.

However, there exists literature only focusing on the task of stance detection, or more generally, approval detection.

Sen et al. [70] give a good overview of the issue. In the general terms of approval detection, the authors distinguish between three tasks: Untargeted sentiment detection, targeted sentiment detection, and stance detection. The first task deals with capturing an overall sentiment of a text without considering a given target. A popular mechanism for determining sentiment is using lexicons of positive or negative words, which work well independent of the target. VADER [11] is one approach that used such well-established lexicons, like LIWC, and extended them with sentiment expressions often used in social media, such as emoticons, acronyms (e.g. "LOL"), or slang words (e.g. "nah"). Sen et al. showed that VADER performs very well in determining sentiments across various topics.

Compared to the first task, the latter two capture the sentiment/stance regarding a topic or entity, e.g. abortion or Donald Trump. Furthermore, stance detection analyses whether someone is in favor or against the given topic or entity, while the target does not need to be mentioned explicitly. Mohammad et al. [48] proposed a data set that can be used for untargeted sentiment and also stance analysis. In addition, they train an SVM classifier using features like word and character n-gram frequencies and sentiment lexicon features. This approach was also benchmarked by Sen et al. along with other stance detection approaches. Although these methods show great results for targets they were trained on, they do not generalize well when applied to unseen targets and are as good as untargeted sentiments detection methods [70]. Umer et al. [82] proposed a hybrid deep neural network approach, combining CNN and LSTM layers, and assess two different dimensionality reduction approaches, PCA and Chi-Square. After preprocessing and dimensionality reduction of the features, a pair of article headline and body text are fed into a CNN layer, followed by a max-pooling layer. Next, a dropout layer connects the CNN with an LSTM and dense layer, which classifies the stance. Finally, Darwish et al. [20] present an unsupervised stance detection mechanism that clusters users with similar stances regarding a topic. This method spares time for manual labeling, since only a few clusters instead of thousands of tweets or user have to be annotated with stances. However, this solution requires the tweets for clustering and annotation to be related to the same topic.

3.4.4 Propagation- and Source-based Methods

Propagation-based methods focus on the creation and propagation of an article, which can be expressed by a propagation network. Shu et al. [71] provide an overview of different network types used when detecting false information. They distinguish between homogeneous and heterogeneous networks, i.e. networks consisting only of a single type of nodes/edges vs. networks with multiple types of nodes/edges. For instance, a friendship network, which connects two users, when they are friends on the social platform, is a homogeneous network. On the other hand, an interaction network, which connects news articles with its publisher and sharing users, is a heterogeneous network. Another possible propagation graph is the News Cascade [95]. It is a tree-like directed graph, where the

nodes are users that shared a certain article and the arcs indicate from whom the user forwarded the article. The root of the tree therefore represents the user, who first shared the article. The key issue is now to extract features from those networks and find proper embeddings for them in order to perform classification tasks. Possible features are the breadth or depth of a News Cascade, the spreading speed, or the similarity between two graphs. Beside the detection, propagation-based methods also try to mitigate the propagation of false information. Using a network representation, one can identify the provenance of false information transmission or the persuaders, which are accountable for spreading the news most effectively.

Both, propagation- and source-based methods focus on the entities responsible for creating and sharing news on social media. While propagation-based methods rather consider quantitative features of the whole propagation process, source-based methods concentrate more on qualitative aspects of the publisher and the propagating user, e.g. their credibility [95]. To assess the credibility of a publisher, manual work has been done by Grinberg et al. [30], who analyzed the spread of fake news on Twitter during the US presidential election in 2016. For this they used a “list of news sources constructed by fact-checkers, journalists, and academics” and categorized them into black, red, and orange. Also Pennycook et al. [59] asked professionals to evaluate the reliability of news sources and compared their estimates with that of laypeople. Gallotti et al. [27] analyzed the spread of fake or misleading news during the COVID-19 pandemic. For this they collected class labels from various fact-checking websites and mapped them onto a harm score scale from 1 to 9, classified website domains shared on Twitter, and analyzed their spread through the network. Instead of classifying a news article by assessing the credibility of its source, false information can also be detected by identifying malicious propagating users, e.g. bots. One well-known bot classifier is Botometer [68], which extracts a wide range of features from Twitter, e.g. network, temporal or sentiment features. Green et al. [29] presented an analysis, where they connected social media accounts based in the US with voter registration entries and estimated the relative partisanship of articles and sources as the normalized ratio of shares in social media, i.e. whether it was rather shared by Republicans or vice versa. However, the authors emphasize that these estimates cannot be used as standalone partisanship scores, since a news source can be shared by a single partisan group, regardless of the published content.

3.4.5 Hybrid Methods

As already mentioned, many false information detection approaches combine several methods to profit from all advantages. Exploiting news content features enables researchers to detect false information at an early stage, since it does not depend on user engagements or propagation features. On the other hand, such auxiliary features from social media can improve detection algorithms. The way how false information articles are written is intended to mislead readers and thus also automatic detection, which is based solely on content, is difficult [74, 75]. This section presents hybrid approaches with the focus on how different features are combined.

Cui et al. [19] proposed a Sentiment-Aware Multi-modal Embedding called SAME. First, the framework extracts features from articles images, text, and source and learns for each modality an embedding by incorporating an adversarial mechanism equalizing the distribution of each modality. After that all three embeddings are concatenated and weighted with a sentiment measure consisting of sentiments between comments of two articles. Ruchansky et al. [66] designed a framework called CSI considering three characteristics of fake news, namely the article text, the user responses and the initiating group of users promoting it. One module of the proposed framework is a temporal model of user activity for a given articles, which receives temporal spacing and the text of an user's activity and results in a low dimensional temporal representation. A second module extracts an implicit user graph given shared engagements on articles and scores users according to their participation in source promotion groups. For each article the temporal embedding and user scores of people engaging with that article are concatenated and fed into a final neural layer. Shu et al. [72] presented an explainable fake news detection framework combining content and comment features. Both content and comments are fed into distinct encoding networks, whose outputs are modified by a sentence-comment co-attention mechanism. The attention mechanism helps to reinforce sentences or comments relevant for classification. The authors use both information sources simultaneously, because they are semantically associated to each other. After concatenating the resulting vectors a final neural layer predicts the label. Xie et al. [93] presented a Stance Extraction and Reasoning Network (SERN), which combines content-based features, like text and images, with stances taken from news content and user replies. The authors use BERT as automatic stance extraction technique, followed by a deep reasoning network that models interaction between users, which influence the individual stances, since news readers are exposed to several reactions from different user and form their final opinion based on a mixture of those information. The feature vector produced by the reasoning network is then concatenated with the output of second module combining text and images of the news piece and fed into a deep neural network, which predicts the final label. The special feature of the stance detection approach is that the stance extraction component automatically detects the stance without the need of labeling it. The output of the reasoning network is a hidden stance representation that is used by the proposed framework to detect fake news. Opposed to the previous approaches, which all use at some stage well researched neural network components, Shu et al. [76] proposed the TriFN framework, a custom learning algorithm with a self-defined update mechanism that aims to minimize an optimization function. The framework models the tri-relationship between publisher, news pieces and social network user. This optimization function consists of two non-negative matrix factorization functions obtaining embeddings for news content and user and two additional terms modeling user-news and publisher-news interactions. The user-news interaction term minimizes feature distances of high-credible users and true news and low-credible users and fake news, respectively. The publisher-news term obtains latent features of news publisher by combining features of all published news pieces.

3.5 Detecting False Information Sources

So far, we have seen how information about the credibility of a source can aid the detection of false news articles. Now we will summarize existing research regarding the estimation of those source credibility scores, which is often called “fact-checking” [15, 53, 61].

3.5.1 Manual Quality Assessment

Compared to the detection of false news articles, this is mostly approached by manual work, since assessing the reliability of single articles is tedious work. Bufale [8], Bufalopedia [9], and Butac [10] are three Italian websites that actually debunk false claims, but also maintain a “black list” of false information news sources, which are mainly distinguished by topics, e.g. conspiracy theories. They claim to have no political affiliation and that their work is financed by donations. It is also possible to report articles, which need to be checked or debunked. Also BuzzFeed News [50, 51] has compiled a list identifying 667 news sources having left or right partisanship. The main websites of each news source have clear or even openly stated preference with either liberal or conservative positions. The list does not contain any mainstream medium, even though some have liberal tendencies “due to their geographic location or staff makeup”. Like the three Italian fact-checking websites, also Politifact [61] actually debunks false claims or provides information about common false information scams that can easily fool people. In addition, they have published in 2017 a list of 330 false news sources, separated by their motivation to share false information, but with the common ground to publish false information intentionally. Another well known organization is Media Bias Fact Check [15], which claim to have estimated the bias of more than 4300 news sources. They are working independently and are financed by donations, ads on their website, and memberships. The bias of a news source is determined by a mix of objective measures and subjective analysis. While these news source ratings were all compiled by expert groups, Melissa Zimdars [62, 98] created a list by herself and with some help from her Facebook friends. As an assistant professor of communication and media she noticed that her students cite news sources generating strong attention. As a resource she created a list of news sources, not only containing clickbait or misleading news, but also false information, biased news sources, or conspiracy. Gallotti et al. [27] did not rate news sources by themselves, but rather collected existing news source reliability ratings from the above named fact-checkers and unified them onto a harm score scale from 1 to 9. These scores were then used to analyze the spread of Tweets referring false information sources during the COVID-19 pandemic. Unfortunately, the mapping from existing ratings to their harm score is not transparent. Finally, a rather different approach is taken by NewsGuard [53]. Instead of dividing news sources into different categories, they review, whether the news sources satisfy certain quality criteria. NewsGuard uses two different groups of criteria, one concerning the credibility (Table A.10), e.g. whether it has ever reported false information and not corrected it, and the other regarding transparency (Table A.11), e.g. whether a source discloses its financing. Experienced journalists are responsible for checking the criteria for every listed website and the results are reviewed by editors.

3.5.2 Automated Credibility Assessment

Although much more research has been done to detect fake news articles, some approaches regarding automatic reliability assessment of news source exist. Baly et al. [4] identified the field of predicting a website’s reliability as “under-explored”, although it is a well known issue. As already mentioned, many fact-checking organizations rating the credibility of news sources and websites exist. Also in journalism, it is an agreed requirement to question the credibility of an information source. Therefore, the authors collected over 1000 news sources from Media Bias Fact Check [15] and mapped each to two types of target labels: A 7-point “political bias” label, having values “extreme-left”, “left”, “center-left”, “center”, “center-right”, “right”, “extreme-right”, and a 3-point “factuality” label with its values “low”, “mixed”, “high”. For each source they crawled a small sample of their articles (10-100), its Wikipedia page and Twitter account (if existing), features representing the structure of the URL, and information about the Web traffic from Amazon’s technology *Alexa*. From their articles they extracted content features representing, among others, structure, topic, or bias. To yield a single representation vector per news source, the feature vectors of all articles of a source are averaged and concatenated with the remaining features, which are already at a news source level. The final label is then predicted using an SVM.

In follow-up research, Baly et al. [5] approached the problem with a different set of features, which are shown in Figure 3.2. Compared to the previous research, they dropped information about URL style and Web traffic and additionally incorporated (i) features extracted from YouTube videos posted by the source’s YouTube account, including acoustic and linguistic features as well as metrics like number of views, likes, dislikes, and comments, (ii) the self-description of Twitter users following the news source’s Twitter account, and (iii) the political spectrum of Facebook users who are interested in the news medium according to Facebook’s Marketing API. Again, in case the features are not already extracted per news source, but at a finer level of granularity, e.g. article features, they are averaged per news source. In contrast to the first research, they reduced the number of values for the political bias label to “left”, “center”, “right”. The resulting 5 types of features are again concatenated and used as input for an SVM classifier predicting the target label.

3.6 Data Labeling

As shown in the previous sections, multiple different kinds of false information exist and therefore there is no widely used definition of it. Also, detecting false information articles comes with additional challenges, like stance detection or source credibility assessment. For each challenge different features can be extracted and various approaches exist, each using other features. When using Machine Learning to tackle these challenges, one needs reliable ground truth labels for the available data. To date research uses datasets that have been compiled in different ways, each offering different benefits [7].

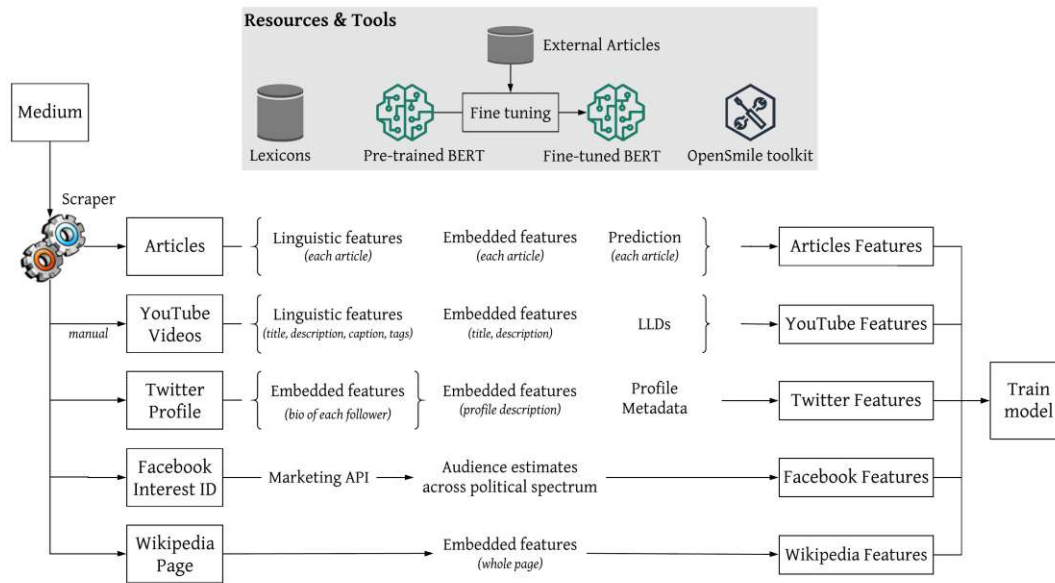


Figure 3.2: Framework by Baly et al. [5] to predict political bias and factuality of news sources. For each news source (medium) 5 entities are scraped, from which features are extracted. Features inside curly brackets $\{ \}$ are averaged per news source.

3.6.1 Manually Labeled Datasets

One way to label data is by assessing the ground truth manually, either by a few experts or by using crowd sourcing. An often mentioned dataset [7, 32, 65, 73, 74] for fact-checking of short statements or claims is LIAR, proposed by Wang et al. [89]. The dataset contains around 12 800 short statements, human labeled from `politifact.com`, which were relabeled as “false”, “half-true”, and “true”. Additionally, it is enriched by the speaker, the context, and a justification by the authors of the label. Another widely used dataset [19, 72, 76, 94] is FakeNewsNet by Shu et al. [73]. It consists of almost 23 200 news articles expert-labeled by Politifact and GossipCop. Additionally, it provides content, context and spatiotemporal features of the articles, like social media user locations and timestamps of posts and comments.

3.6.2 Weak Supervision

Since compiling a gold standard for short statements or articles is tedious work, much more manual labeling was done for news sources, as already shown in Section 3.5. Although crowd-sourced fact-checking, as done by Pennycook et al. [59], has better scalability in terms of adding more laypeople, it has management overhead and produces less credible and accurate results, since the individual biases of the people can falsify the classification [32, 55, 57, 74, 95]. Nevertheless, popular datasets providing a ground truth label for news articles exist and are used to classify news articles as false information.

3. RELATED WORK

The articles included in those datasets were not assessed individually, but were assigned to a target label according to weak supervision, which labels news articles the same as their sources. This mechanism reduces the amount of manual work significantly, because instead of thousands of articles one has to label only their sources [55].

One example dataset regarding weak supervision is NELA-GT-2018 [55] by Nørregaard et al. It consist of more than 700 000 news article from almost 200 labeled news source, which were compiled by fact-checker like Media Bias Fact Check [15] or NewsGuard [53]. Another example dataset is CREDBANK [47], which unlike the previous one does not contain news sources or articles, but around 1000 real events and 60 million tweets reacting to them. For each event the credibility was evaluated in terms of polarity (“Accurate”, “Inaccurate”, “Uncertain”) and degree of certainty (“Certainly”, “Probably”, “Uncertain”) by Amazon Mechanical Turk workers. Then tweets relating to those events were queried, whose credibility can be weakly labeled according to their related event [7, 73].

Data Acquisition and Labeling

In this chapter we describe the composition of our dataset. The data consists of news sources and labels classifying the credibility of the sources. For each news source we collect posts from social media that shared an article published by the source, the users who wrote the posts, and the content of the articles shared in the posts. We begin in Section 4.1 to unify the credibility labels of various fact-checker onto a common scale. In Section 4.2 we provide further details on how we mapped the existing labels to the unified scale. Finally, in Section 4.3 we describe how we collected posts, users, and articles for each news source.

4.1 Unification of News Source Ratings

In order to automatically assess the credibility of news sources we need ground truth data, for which we can identify common patterns and train a model on. In Section 3.5 we summarized the work of various fact-checking websites. Each of them has manually compiled news source labels by experts, who evaluate, among others, the journalistic review process and trustworthiness of articles. Also Baly et al. [4, 5] have used ratings by fact-checkers as ground truth labels, namely around 1000 sources from Media Bias Fact Check [15]. We argue that more training instances are necessary to make reliable statements about the model's performance. Unfortunately, it is not straightforward to use the labels of multiple fact-checkers, since each of them is using different ones. Therefore, the first task is to find a uniform labeling scale reflecting the reliability for news sources.

A similar task was already performed by Gallotti et al. [27]. For their analysis of fake news spreading during the COVID-19 pandemic they mapped reliability ratings for news sources from different fact-checking websites to a harm score. Unfortunately, it is not transparent what harm score the news websites received or how they were assigned. Besides the availability, the harm scores have other drawbacks for our purposes. First, the labeling scheme is too fine grained, which makes a clear separation between two classes

Score	Label	Description
1	False information	No or very little accuracy (e.g. fake news, conspiracy, satire)
2	Clickbait	News with some truthness or facts, but actually misleading (e.g. clickbait)
3	Biased	Mixed accuracy, hide/report half of the truth (e.g. bias, state)
4	Mainstream	Low biased or mainstream media (e.g. center-right/center-left bias)
5	Scientific	No biased or scientific news

Table 4.1: Accuracy scores

Score	Label	Description
1	No Transparency	Intentionally misleading or no information about editorial process (e.g. fake news, conspiracy)
2	Mixed Transparency	Sites with (partially) transparent intention, but can still be misunderstood because of the way articles are written (e.g. bias, clickbait, satire)
3	Transparent	Sites with transparent editorial or legal notice (e.g. mainstream, scientific news)

Table 4.2: Transparency scores

harder, e.g. harm scores 8 and 9 are very similar. Another issue is that a harm score gives only a limited representation about the reliability or factuality of a news website. For instance, satire has a reasonable harm score of 3, since it is mostly recognized as such and therefore causes less harm than clickbait or political news for instance. However, satire usually reports completely fabricated news based on little truth, which is not represented by the harm score. Finally, the scale has some logical errors, as score 5 and 6 appear to cause medium to medium high harm, but actually are unknown labels, which could lead to misinterpretation.

To come up with a more accurate score, we looked for similarities in labeling schemes of different fact-checking websites. We considered labels from Bufale [8], Bufalopedia [9], Butac [10], BuzzFeed News [50], Columbia Journalism Review [64], Fake News Watch [91], Media Bias Fact Check [15], NewsGuard [53], Politifact [61], and Zimdars from The Washington Post [62]. Most of these sources are also used by Galotti et al. [27]. We

found that many fact-checkers use various labels to describe different forms of false information, e.g. Butac uses labels like “pseudoscience” and “pseudomedicine”, while Zimdars uses “fake”, “conspiracy”, and “junksci” (junk science). The distinction of those labels lies in the topics the fact-checker focus on, e.g. alternative medicine versus general conspiracy theories. These distinctions are not required for our task, which is why they can be merged to one category. Other widely used categories are any kind of misleading news, clickbait, and biased news. Less used, but important for our analysis, are labels describing accurate news. From these observations we derived an ordinal accuracy score with 5 different labels, shown in Table 4.1, which relates to the credibility and tells how fact-based the news outlet reports. A detailed description of the mapping from fact-checker labels to the accuracy score is given in Section 4.2. As already mentioned for the Gallotti harm score, satire news sources are difficult to include into an ordinal reliability or accuracy scale, since they report false information, but not in a harmful way. Therefore we came up with an additional ordinal transparency score, shown in Table 4.2, which has 3 different values and evaluates how honest the source is in terms of factual fidelity, bias, or hoax.

4.2 Label Mapping Scheme

This section explains how news sources labeled by the fact-checking websites are assigned to an accuracy and transparency score. In Appendix A we listed the labeling criteria for all fact-checking websites and the assignment to an accuracy and transparency score. Some websites provide further descriptions for their labels and rules, which we preserved.

We developed two different mapping approaches based on the data from the fact-checkers. While NewsGuard [53] uses a set of quality criteria to assess the reliability of a news website, the remaining fact-checkers provide a categorization of news websites, where each website is labeled with exactly one category rating credibility, factuality, or sometimes risk potential.

Starting with the latter group of fact-checkers, we labeled each website with an accuracy and transparency score by mapping their existing label to our labels as the following: The three Italian websites Bufalopedia, Bufale, and Butac (Table A.1, A.2, and A.3) distinguish between several forms of false information and conspiracy blogs, which is why a lot of labels were assigned with accuracy 1. Also “false satire” and (regular) “satire” have very low accuracy, which is why they also receive a 1. Bufale’s “clickbait” gets per definition accuracy 2, likewise does Bufalopedia’s “misleading” label, since it is no fake news, but has low accuracy. The transparency of “clickbait” however is higher, than “misleading”, since behind “clickbait” are often newspapers with at least little journalistic standards, like imprint or name of the authors. That is of course not always the case and therefore it gets mixed transparency. Satire gets also mixed transparency, since it is honest with its missing truth in most of the cases, but can sometimes mock trusted news. However, “false satire” is clearly dishonest with telling false news, which is why it receives transparency 1. Next comes the partisan news analysis by BuzzFeed

News (Table A.4), whose categories receive accuracy 3 per definition and transparency 2, as some of them have at least little journalistic standards similar to “clickbait”. The following two fact-checking websites by Columbia Journalism Review and Fake News Watch (Table A.6 and A.7) show similar labels as the previous ones, which makes labeling self explanatory. This is not the case for Media Bias Fact Check (Table A.8). We rated “questionable sources” with accuracy and transparency 1, since it can contain conspiracy and fake news as well. Media Bias Fact Check distinguishes between 3 levels of biases, “least biased”, “left/right center bias”, and “left/right bias”. To keep this separation we assigned these labels with accuracy 5, 4, and 3, respectively. Additionally, the center bias classes have higher transparency, than (regular) bias, therefore they get the score 3. For Politifact (Table A.5) we labeled “Imposter site” like “false satire” and “Parody site” like “satire”. “Some fake stories” is labeled like “clickbait”, but with lower transparency, since it cannot be said, whether it is intentional. The last fact-checker Zimdars (Table A.9) uses a lot of labels seen at the previous ones, which makes mapping straight forward. Exceptions are “rumor”, which is equal to “clickbait” and “state”, which is equal to the biased labels. We excluded “hate” and “unknown”, since the factual fidelity of these labels are unclear.

Finally, we assigned accuracy and transparency labels to the websites from NewsGuard [53]. NewsGuard uses two groups of quality criteria, one assessing a website’s credibility and the other its transparency, which are shown in Table A.10 and A.11. Each of those criteria is assigned a numeric score, such that a website satisfying all criteria gets a score of 100. Websites with 60 or more points are labeled as “green”, otherwise as “red”. Websites that are either satire news sources or a news platforms that “primarily host user generated content” [53] are not rated using the criteria. We therefore excluded “Platform” websites from the analysis and mapped satire news sources, as done previously, to accuracy 1 and transparency 2. For the remaining websites we build a rule-based assignment depending on the criteria that a website satisfy, shown in Table A.12 and A.13. The rules use the IDs of the criteria as propositional variables, i.e. $c_i = 1$ if criterion c_i is satisfied, $c_i = 0$ otherwise, and same for t_i . The rules are applied in the given order to determine the accuracy and transparency score, because some of the rules are not exclusive, e.g. a website satisfying all c_i criteria would satisfy the rules leading to accuracy 5 and 3 as well.

Beginning with the credibility criteria, we map websites satisfying all criteria to accuracy 5, satisfying two criteria to 2, and satisfying at most one to 1. To receive an accuracy score above 2, a website has to satisfy c_1 , i.e. it does not repeatedly publish false content. In distinction to accuracy 3, a website rated as accuracy 4 needs criterion c_2 , i.e. responsible presentation of information, since the absence of this criterion is an indicator for biased or less accurate news. Additionally, accuracy 4 news sources can only miss one criterion to be rated as “Mainstream” news. The transparency score mapping is rather straight forward. Only news sources satisfying all transparency criteria are awarded with transparency 3. Satisfaction of three or two criteria is mapped to mixed, less than two to low transparency.

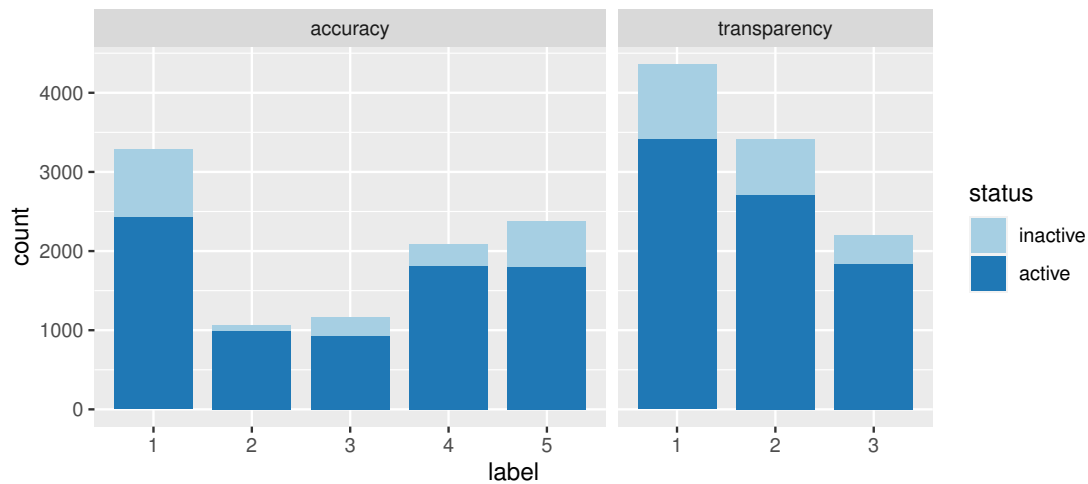


Figure 4.1: Number of news sources per accuracy and transparency label. Each bar additionally shows the share of inactive news sources, i.e. news sources not responding with HTTP status code 200.

After we labeled all websites with our two scores, we merged all lists of websites to a single list. Since some websites were considered by multiple fact checkers, they appeared more than once in the final list with possibly different scores. We removed those duplicates by keeping the lowest accuracy and transparency score. This results in 9971 news sources, of which 7957 are still active, i.e. returned an HTTP status code 200. The label distribution is shown in Figure 4.1.

4.3 Collecting Data

After labeling the news sources we need to collect data, which we can extract features from. We have outlined in Section 3.4 that existing approaches detecting fake news articles use a broad selection of features, including content and social context features. When it comes to the reliability classification of news sources, existing approaches have primarily extracted content-based features from a source’s articles or YouTube videos. In their second work, Baly et al. [5] include only little information extracted from social media by analyzing the self-description of follower of the news source’s Twitter account. There is a lack of information, though, since more than a quarter of their analyzed sources do not have a Twitter account and Twitter users may not have a biography. It is also badly scalable, since looking for the Twitter account of a news source includes manual work. Instead, we not only collect articles of a given news source, but also include information how and by whom these articles are shared on social media. To this end, we query posts from Twitter¹ and Facebook² sharing news articles of a given source. Both

¹<https://twitter.com/>

²<https://www.facebook.com/>

networks are regularly used to consume news [13]. From the collected posts we extract information about the post and the posting user to capture social network activities to news sources more comprehensively. Additionally, we collect the article shared in the post, which limits the set of articles we yield to relevant and “popular” ones, as they have been shared on social media. Since both platforms provide APIs we can automatically scrape posts sharing articles of a given source without manual work included [17, 38].

4.3.1 Query Social Media APIs

Facebook posts are accessible via the CrowdTangle API [17], which provides two interesting endpoints, namely `/posts` and `/links`. The `/posts` endpoint enables an user to query posts using a search term, which could be an URL as well. The `/links` endpoint however is specialized for retrieving posts matching a given web domain and is also capable of finding posts sharing shortened URLs that point to the requests one, which is why we queried this one. However, both endpoints only return posts, which were published by either Facebook pages or inside public Facebook groups. For each news source we queried posts in a 6 month time period, from April 1, 2021 to September 30, 2021. The API returns up to 1000 posts per query, which can be sorted by date, subscriber count of the posting user, or the total number of interactions. This means for news sources, whose articles are frequently shared on Facebook, that the 1000 posts we receive are highly clustered, i.e. having the same date, follower count, or number of interactions. Therefore we formulated for each news source six queries covering a 1 month

```
url = "https://api.crowdtangle.com/links"
for news_url in news_urls:
for month in range(4, 10):
params = {
    "link": news_url,
    "startDate": datetime(2021, month, 1).strftime("%Y-%m-%dT%H:%M:%S"),
    "endDate": datetime(2021, month + 1, 1).strftime("%Y-%m-%dT%H:%M:%S"),
    "count": 1000,
    "sortBy": "date",
    "includeSummary": "true"
}

response = requests.get(url, params=params)
```

Listing 4.1: Python pseudocode for querying Crowdangle’s `/links` endpoint, where `news_url` is the root URL of the news source and `startDate` / `endDate` is inclusive / exclusive. For each news source we construct a query retrieving Facebook posts from an one month period that shared some article of the news source. We repeat that for all months from April to September.

```

url = "https://api.twitter.com/2/tweets/counts/all"
for news_url in news_urls:
    params = {
        "query": f'lang:en -is:retweet url="{news_url}"',
        "granularity": "hour",
        "start_time": "2021-04-01T00:00:00Z",
        "end_time": "2021-10-01T00:00:00Z",
        "next_token": None
    }
    while True:
        response = requests.get(url, params=params)
        # Update params["next_token"] and loop until no next_token is in
        response

```

Listing 4.2: Python pseudocode for counting Tweets. For each news source we retrieve an hourly count of Tweets that shared some article of the source between April 1, and September 30, 2021.

period, from the first to the last day of the respective month. Since the Crowdtangle API does not provide an endpoint for estimating the number of posts returned by a query, we queried for each news source and month the maximum number of posts per query, i.e. 1000, which has the side effect that we could get an unbalanced dataset in terms of number of posts per label. The final query parameters are shown in Listing 4.1. In total, these queries resulted in 14,825,514 posts published by 756,318 unique Facebook accounts.

For querying Tweets we used the Twitter API [38], which provides the `/2/tweets/search/all` endpoint for querying Tweets matching certain criteria. With the `query` parameter we can set various constraints to restrict the set of returned Tweets, e.g. containing keywords or hashtags [36]. The most relevant query operator for us is `url:`, which enables us to query Tweets sharing external URLs, even shortened ones, matching the requested URL. By concatenating URLs using the `OR` operator we can query multiple news sources using one API query. Different from the Crowdtangle API, the Twitter API has however many limitations, which we had to overcome:

- Max. 500,000 Tweets in total
- Max. 1 API request per second
- Max. 100 Tweets returned per API query³
- Max. 1024 characters for the query parameter, which limits the number of news sources to simultaneously request Tweets for

³Actual limit is 500, but we requested `context_annotat`ions as well

By simply querying Tweets for each news source, we would yield at most $7957 \cdot 100 = 795,700$ Tweets with an unknown distribution in terms of number of Tweets per news source and per label. Fortunately, the Twitter API provides the `/2/tweets/counts/all` endpoint to estimate the total number of tweets returned for each query. In order to design queries, which respond 500,000 Tweets in total and have a balanced distribution over the news source labels, we counted the Tweets for each news source URL in the same time period as the Facebook posts, from April, 1 2021 to September, 30 2021, as sketched in Listing 4.2. With the `query` parameter we further restricted the result set to English Tweets that are no Retweets, since including Retweets could lead to duplicated Tweets in the dataset. Additionally, we set the `granularity` to `hour`, which counts Tweets for each 1 hour time period between `start_time` and `end_time`.

With those hourly Tweet counts we can design queries, each using a combination of URLs and a proper time period, that are guaranteed to return 100 Tweets. Therefore, we first formed sets of news source URLs U_1, \dots, U_n , such that the URLs $u_{i,1}, u_{i,2}, \dots \in U_i$ inside each set

- belong to news sources with the same accuracy and transparency label, and
- the resulting query `lang:en -is:retweet (url:"<u_i,1>" OR url:"<u_i,2>" OR ...)` has at most 1024 character, where `<u_i, j>` is replaced by the URL string $u_{i,j}$.

Next, we looked for the smallest time periods a query would return at least 100 Tweets. Let h_1, \dots, h_m be an ordered sequence of hourly timestamps, with $h_1 = 2021-04-01T00:00:00Z$, $h_m = 2021-10-01T00:00:00Z$, and $c(U_i, h_s, h_e)$ be the total number of tweets for the URLs U_i in the time interval $[h_s, h_e)$. We therefore form for each URL set U_i several non-overlapping time intervals $T_{i,j} = [h_{s,i,j}, h_{e,i,j})$, such that

- $1 \leq s < e \leq m$,
- $c(U_i, h_{s,i,j}, h_{e,i,j}) \geq 100$, and
- $c(U_i, h_{s,i,j}, h_{e-1,i,j}) < 100$.

An API query requesting URLs U_i during time $T_{i,j}$ is now guaranteed to return 100 Tweets, which minimizes the number of requests to the API and therefore the necessary time, since the API only allows for 1 request per second. Unfortunately, this query optimization was not applicable to the Crowdtangle API, since it does not provide an endpoint for estimating the number of posts returned.

With the URL sets and their time intervals we randomly sampled 4388 distinct pairs $(U_i, T_{i,j})$, such that each URL set is queried for at least three different time intervals, if possible, and the number of collected tweets has a balanced distribution in terms of the labels of the news sources the tweets shared. Those queries can then be used to construct

```

url = "https://api.twitter.com/2/tweets/search/all"
params = {
    "max_results": 100,
    "tweet.fields": "id,text,author_id,context_annotations,
conversation_id,created_at,entities,geo,possibly_sensitive,
public_metrics,referenced_tweets",
    "expansions": "author_id,geo.place_id",
    "user.fields": "id,name,username,created_at,description,location,
public_metrics,verified",
    "place.fields": "full_name,id,contained_within,country,
country_code,geo,name,place_type"
}
for query in queries:
    # query.keys() == ["urls", "start", "end"]
    url_clause = " OR ".join([f'url:"{url}"' for url in query["urls"]
])
    params["query"] = f"lang:en -is:retweet ({url_clause})"
    params["start_time"] = query["start"]
    params["end_time"] = query["end"]

    response = requests.get(url, params=params)

```

Listing 4.3: Python pseudocode for requesting Tweets with prebuilt queries. Each query contains a list of URLs and a start and end timestamp.

API requests, as shown in Listing 4.3, which resulted in 434,261 Tweets from 166,990 unique users.

4.3.2 Collect Articles and Data Cleaning

After querying posts from Twitter and Facebook we could retrieve 4,843,199 unique external URLs from the posts. We downloaded the content of those websites by using the Newspaper3k library [58]. The library was designed to interpret a website as a news article and extract useful information. This can simply be done by calling the `download` followed by the `parse` method. We ignored all websites, where the GET request did not respond with HTTP status 200. For all successful requests we receive as a result an `Article` object, from which we kept the following attributes for now: `title`, `text`, `meta_keywords`, `tags`, `authors`, `publish_date`, `meta_description`.

For each news source we now have collected social media posts, the publishing user, and the shared news article belonging to that news source. In order to yield a valuable dataset, we performed the following data cleaning steps: We keep only articles for which we could extract an article title and body text. In addition we required each article to be shared at least by 5 posts on social media, no matter on which platform, and each news sources to have at least 5 articles. After this data cleaning step only 1961 news sources remained, but each having a solid base of articles and social media posts. The process

also changed the distribution of the news sources regarding their labels, as shown in Figure 4.2. Compared to Figure 4.1 we excluded many news sources with accuracy 1 or 2. Another remarkable change is the reversed order in the transparency score. All these observations can be explained that especially low quality, unreliable news sources survive only a short time on the web, until taking down their services and articles. Additionally, we required at least 5 articles per source, each being shared in at least 5 posts, which removed small and unpopular news sources from the dataset. In Figure 4.3 we show the total sum of distinct articles, posts, and users for each label. As we can see, the share of each entity is approximately the same for each label. The smallest share belongs to the articles, since articles are shared by multiple posts.

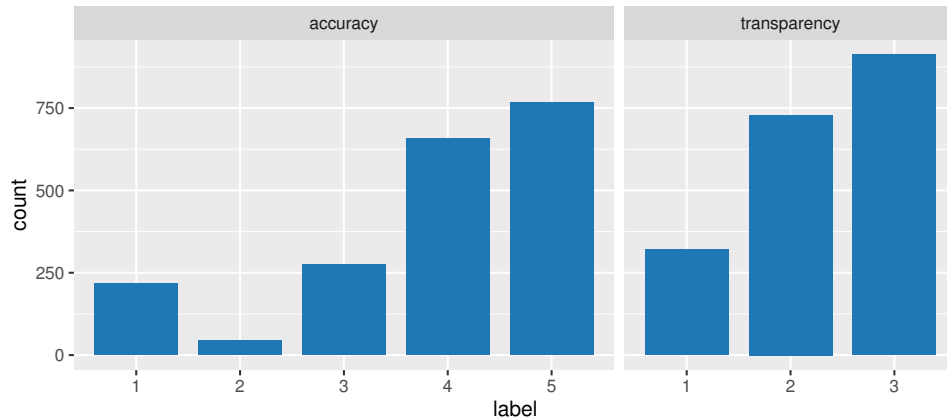


Figure 4.2: Number of news sources per accuracy and transparency label after data cleaning. Each news source is required to have at least 5 articles, each being shared in at least 5 posts on social media.

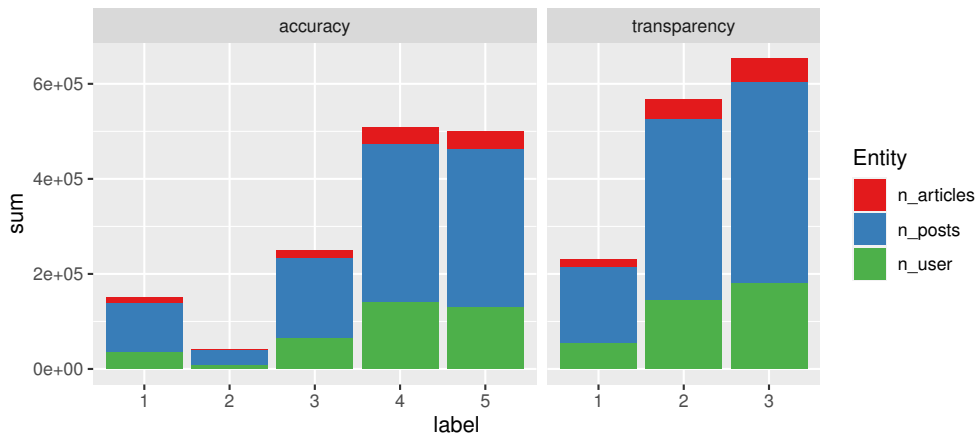


Figure 4.3: Number of distinct articles, posts, and users per accuracy and transparency label.

Post Sequence Model

In this chapter we propose a novel Machine Learning architecture, which predicts the reliability of news websites. For this purpose it takes as an input a set of news articles published by the website and a set of social media posts and their associated users, which shared those articles. Instead of averaging the features extracted from all three entities on a news source level, like is was done in previous research [4, 5], we will learn the representation of a news source, which would be able to weight the extracted features differently and thus be more flexible. Section 5.1 begins with the key idea and a broad overview of the model architecture. In Section 5.2 and 5.3 we describe the features we selected and how we embedded them into a numeric representation. At the end in Section 5.4 we combine all modules to the final model.

5.1 Model Overview

The main component of the model is an adaption of the sequence modeling technique presented by Shu et al. [71]. In this work the authors proposed a temporal representation of news articles by incorporating features from social media posts and their users that shared these articles. The resulting article embedding can then be used for classification tasks, like false information detection. For this purpose, they identify a diffusion network for each article, where the nodes represent social media users. Two users u_i, u_j are connected by an directed arc $u_i \rightarrow u_j$, if and only if u_j follows u_i and both shared the given article, but u_j did after u_i . This diffusion network embodies the idea that u_j was exposed to an article, since her follower u_i posted it, and as a result decided to propagate it further to her followers. The user post is seen as an engagement $e_i = \{s_i, u_i, t_i\}$, where user u_i shares an article in post s_i at time t_i . All engagements from the diffusion network are transformed into an ordered sequence e_1, \dots, e_n , where $t_1 \leq \dots \leq t_n$. From each engagement they extract several features, resulting in a final feature vector \mathbf{x}_i . Now they can encode the sequence $\mathbf{x}_1, \dots, \mathbf{x}_n$ using an RNN, as shown in Figure 5.1. Optionally,

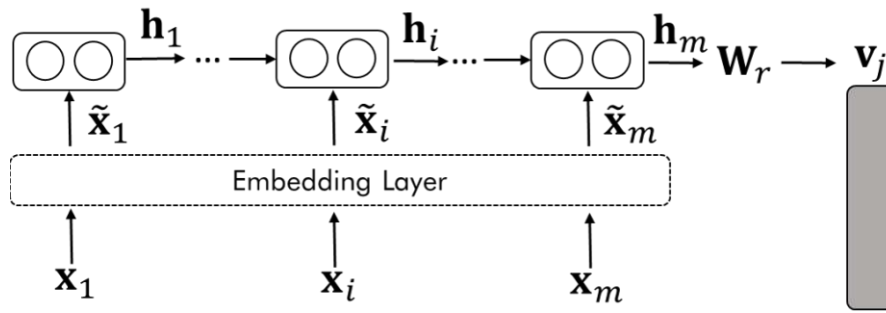


Figure 5.1: RNN framework for learning news temporal representations. Each \mathbf{x}_i represents the features associated with a post sharing an article on social media. By further embedding it into $\tilde{\mathbf{x}}_i$ and passing the sequence to an RNN, we yield an representational vector \mathbf{h}_m usable for further classification tasks. [71]

\mathbf{x}_i can be further embedded into $\tilde{\mathbf{x}}_i$ by using a fully connected neural layer. The hidden state \mathbf{h}_m of the last step can then be passed to another fully connected layer to perform a classification task.

We adapted this framework, such that we yield a temporal representation of a whole news source instead of a single article. For this we expanded the diffusion network to capture not a single article, but all articles from a given news source. We further add the article a_i to each engagement $e_i = \{a_i, s_i, u_i, t_i\}$, representing user u_i , who wrote post s_i at time t_i sharing a news article a_i , which was originally published by the respective news source. Note that it is possible that a user appears in multiple engagements, since a single user can share two articles, both published by the given news source. Similarly an article can appear in multiple engagements, if two users share the same article. In distinction to the original approach we did not enforce the follower condition. Thus, a news source can have two engagements e_i, e_j , where u_i and u_j are not following each other. This was necessary, since we collected posts from Facebook and Twitter, as shown in Section 4.3, between which no follower information can be established. This breaks down the diffusion network into a set of engagements, which can again be ordered by t_i , beginning with the earliest one. We can now encode this sequence in the same way as Shu et al. proposed, as already shown in Figure 5.1. Using the last hidden state \mathbf{h}_m we can perform a classification task for the whole news source instead of a single article. In the following section we show how we constructed the feature vector \mathbf{x}_i representing the engagement e_i .

5.2 Feature Selection

The first challenge we have to solve is to identify useful features from news articles, posts, and user profiles from Twitter and Facebook. Furthermore, we can only select social media features that appear on both platforms. For posts, we found four common properties, as shown in Table 5.1. The first column shows the given name of the extracted

feature and the second column the transformation of attributes from the Tweet [39] and Facebook post format [18]. Some features could be directly adopted from Facebook or Twitter attributes, e.g. `p|n_comments`, others had to be aggregated to a single feature, e.g. `p|n_likes`. For the latter applies that opposing to Twitter, Facebook users can not only react to a post with a “Like”, but also with various emotions, e.g. “Haha” or “Angry” [46]. Since an emotional reaction is a subclass of a “Like”, we took the sum of all Facebook post reactions and handled it as a Tweet like. For the number of shares, Twitter distinguishes between a “Retweet” and “Quote” (retweet with comment) [37], while Facebook counts the total number of shares, regardless of additional comments. Thus, we added Twitter’s retweet and quote counts. On the other hand, we had to exclude some attributes from our further analysis, for instance the creation time of the post, although they are available in Tweets and Facebook posts. Since the creation time is normalized to the UTC timezone, we have no information about the local time, e.g. whether the post was written at day- or nighttime.

The merging of attributes from Twitter users and Facebook accounts was rather straightforward. We kept three attributes appearing in both data formats, shown in Table 5.2. Again, we did not include the creation time of the social media profile.

Finally, we selected features from the parsed article websites. The Newspaper3k library [58] extracts a lot of information from those websites, e.g. images, movies, meta data, which can be a useful feature to detect false information, as for instance Cui et al. [19] or Baly et al. [5] already showed. We limited article features to textual information only, as shown in Table 5.3. Note that in general our approach can be extended to include other features as well, like image or video features. We did not include the publish date of the article, since for approx 33% of our collected articles a publish date could not be extracted.

5.3 Feature Embedding

After selecting relevant features, we need to embed them into a numerical representation, such that it can be used as an input to a neural architecture. The count features (`p|n_comments`, `p|n_likes`, `p|n_shares`, `u|n_followers`) are already numeric and the boolean `u|verified` feature can be easily transformed into a numeric form (`True` corresponds to 1, `False` to 0). The remaining features are all of textual form, which is why we used language models to encode them as numeric feature vectors. We decided to use RoBERTa [43], since it was primarily trained with text collected from websites and online news, as outlined in ??, which makes an application to our task inevitable. The encoding of the five textual features using RoBERTa is shown in Figure 5.2. First, we tokenized the raw text into its sentences by using `nltk.tokenize.sent_tokenize` from the NLTK library [63]. Empty documents result in a single empty string. We then encoded each sentence one by one using the pre-trained RoBERTa model. We took the model implementation and weights from the Hugging Face Transformers library [24]

¹Nested keys are separated by “.”

5. POST SEQUENCE MODEL

Feature	Tweet attributes (TW) and FB Post attributes (FB)	
p n_comments	TW	public_metrics.reply_count ¹
	FB	statistics.actual.commentCount
p n_likes	TW	public_metrics.like_count
	FB	angryCount + careCount + hahaCount + likeCount + loveCount + sadCount + thankfulCount + wowCount (each from statistics.actual)
p n_shares	TW	public_metrics.quote_count + public_metrics.retweet_count
	FB	statistics.actual.shareCount
p text	TW	text
	FB	message

Table 5.1: Combined features for Tweets and Facebook Posts. The right column show how those features are compiled using original platform features.

Feature	Twitter user attributes (TW) and FB account attributes (FB)	
u verified	TW	verified
	FB	verified
u description	TW	description
	FB	pageDescription
u n_followers	TW	public_metrics.followers_count
	FB	subscriberCount

Table 5.2: Combined features for Twitter users and Facebook accounts.

Article Features		
a title	a text	a meta_description

Table 5.3: Article features we keep. Formerly extracted by Newspaper3k [58].

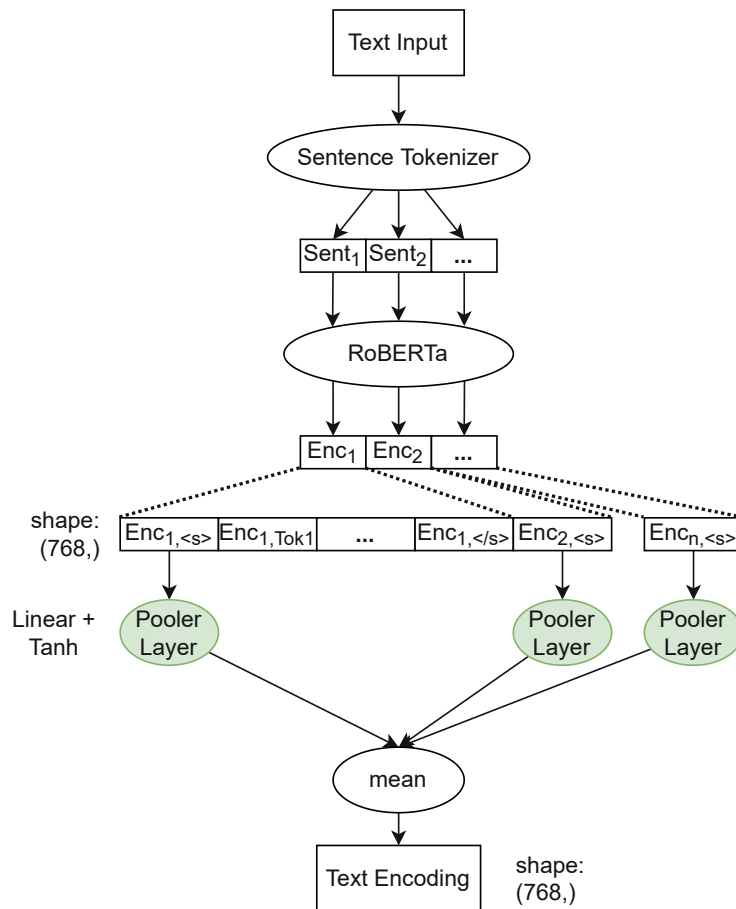


Figure 5.2: Text Encoder module transforming text into numerical features. After tokenizing the input text into its sentences, those are embedded using RoBERTa, but only the first token embedding will be further processed using the Pooler Layer, respectively. The trainable elements are marked green.

with its default configuration, shown in Listing B.1. The library further tokenizes each sentence and encloses it by a starting `<s>` and ending `</s>` token. These tokens are then embedded into a numerical identifier, where empty sentences will result in the start and closing token only. All sentences resulting in more than 512 tokens are truncated, which results in a maximum number of 514 tokens per sentence. Thus, the resulting encoding of each sentence has a minimum shape of (2, 768) and maximum of (514, 768), i.e. each token is transformed into 768 floating point numbers. Now we can leverage the encoding of the special inserted `<s>` token, also referred to as [CLS] token [22, 43], which can be used as representational vector of the whole sentence for further sentence classification tasks [93]. We therefore feed the `<s>` token encoding into a fully connected layer with a tanh activation function, called “Pooler Layer”. Unlike classical fine-tuning, we will only train the weights of the Pooler Layer, not the weights of the transformer layers.

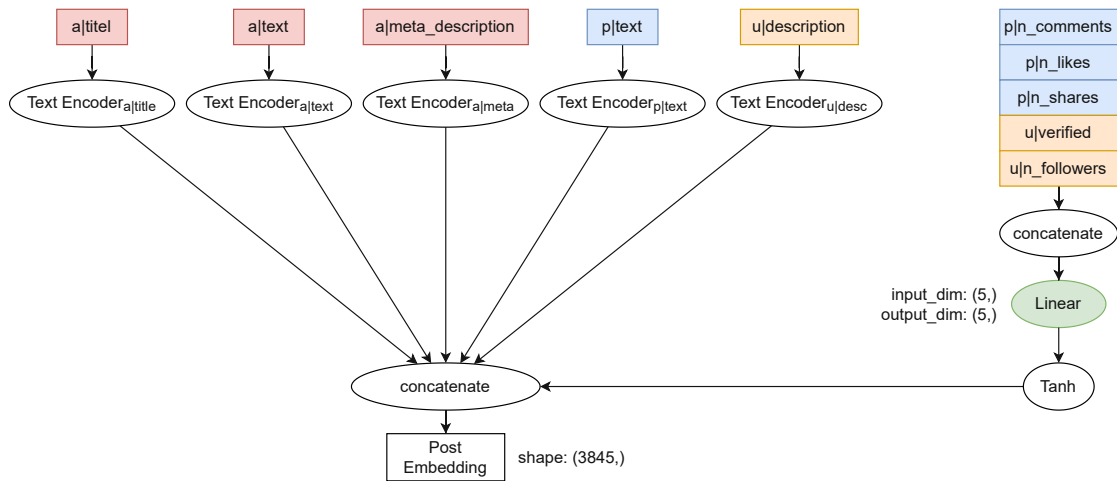


Figure 5.3: Post Embedder concatenating all available features. Features from articles are marked red, those from posts blue, and from users yellow. Trainable components are marked green.

Note that we train five separate Pooler Layers, one for each text feature. Since our text documents consist of a different number of sentences, we need to aggregate the sentence encodings to a single document encoding. This is done by simply taking the average of all Pooler outputs. Another way could be to use an RNN or LSTM, which takes the encoding of each sentence in the document one after another and output the last hidden state vector. In either way, we receive for each text document a floating point vector of shape (768,).

We can now combine the encoded textual features with the remaining numeric features as shown in Figure 5.3. Similarly to the recommendation by Shu et al. [71] we further embed the features with a fully connected layer. We only embed the raw numeric features, since the textual features were already processed. The five text encoding vectors (each of shape (768,)) are concatenated together with the embedded output from numeric features (shape (5,)), which results in a vector of shape (3845,).

5.4 Final Model

The final model putting all presented submodules together is shown in Figure 5.4. It uses a small dataset as an example, consisting of three engagements $\{a_1, p_1, u_1, t_1\}$, $\{a_1, p_2, u_2, t_2\}$, $\{a_2, p_3, u_2, t_3\}$, where $t_1 \leq t_2 \leq t_3$ and t_i is the creation time of post p_i . The submodules are encapsulated by the Post Embedding module, which embeds the engagement into a numeric format. As described in Section 5.1, we applied the approach by Shu et al. [71] and feed the sequence of embedded posts, ordered by its creation date, into a recurrent neural network. We decided to use an LSTM unit instead, since RNN's suffer from the vanishing gradient problem [34]. Another possibility would be to use

a GRU module, which also handles longer sequences better. In Figure 5.4 we chose a hidden dimension of (1000,) for the LSTM, which causes a dimensionality reduction and can possibly discard unimportant features [1].

The LSTM, and similarly RNN or GRU, gives us multiple opportunities to derive a representation for the post sequence. One option was already described in [71], where simply the last hidden state represents the whole sequence, since it is dependent on all previous hidden states. Another possibility would be to aggregate all hidden states, e.g. by taking an element-wise mean, or to concatenate multiple aggregations, e.g. element-wise mean and maximum [40, Sec. 9.4.2]. In either way, we feed this representational vector into a final classification layer, i.e. a fully connected layer followed by a Softmax activation function. The output shape is equal to the number of distinct classes, such that the Softmax activation function gives us the predicted probability of each target class. In Figure 5.4 we assumed that the target label is the accuracy score, which has five distinct classes.

5. POST SEQUENCE MODEL

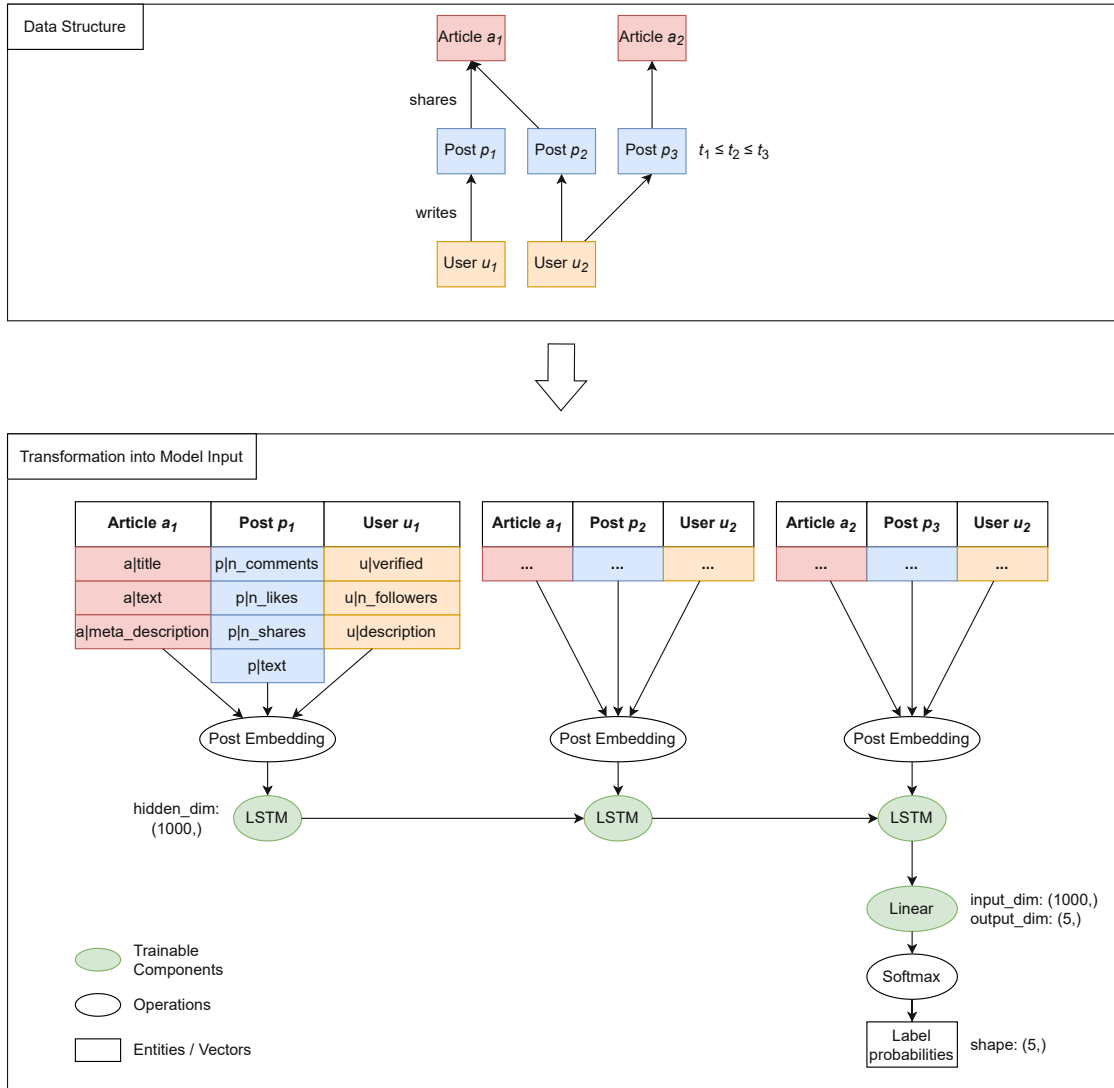


Figure 5.4: Post sequence model classifying news sources. The upper part shows a sample data structure consisting of 3 engagements. In the lower part features for every entity are extracted and embedded using the Post Embedder.

Model Performance

We have described in Chapter 4 the acquisition and labeling of our dataset and in Chapter 5 the architecture of our neural model. In short, we have compiled a dataset of 1961 news sources, each having a sequence of engagements $e_i = \{a_i, s_i, u_i, t_i\}$, i.e. user u_i shared article a_i in post s_i at time t_i . We feed this sequence into our model, which extracts for each engagement features from text using language models, combines them with social context information, encodes the whole sequence using a recurrent neural network, and predicts a source reliability rating. In this Chapter we will evaluate and review the performance of the model. For this purpose we will first summarize our findings from tuning the model's hyperparameters, such that the model is optimized according given metrics and criteria. After the parameter tuning we will use the best model configuration to evaluate and verify the results.

6.1 Parameter Tuning

6.1.1 Setup

Prior to the model's evaluation we have to find a good configuration, such that the model is able to perform best. In Table 6.1 we have defined hyperparameters driving the training of the model and changing the models behavior. One way of finding the best combination of parameter values is by using a grid search, which is given a set of possible values for each parameter and evaluates all combinations against the same metrics. This however can quickly blow up the number of possible combinations. A grid search tuning 5 hyperparameters, each having 10 different values, would train and evaluate $10^5 = 100,000$ different models, which takes a lot of time [1, Sec. 3.3]. Since one training epoch of our model lasts about 1 hour, we need to limit the number of possible combinations. For this purpose we have evaluated possible values of the parameters one after another and combined only promising options with each other.

Hyperparameter	Description
<code>oversampling</code>	Whether to oversample training instances from the minority class(es), such that all classes have exactly the same number of training instances.
<code>n_epochs</code>	Number of training epochs, i.e. the number of times to train on the whole training set.
<code>batch_size</code>	Batch size of the training data, i.e. the number of instances to calculate the loss on and after that the model weights are adjusted.
<code>loss_fn</code>	Name of the loss function. Only “CrossEntropyLoss” was used.
<code>optimizer.name</code>	The name of the optimizer function, which updates the model’s weights. Available options are “SGD” and “Adam”.
<code>optimizer.lr</code>	The learning rate of the optimizer function, if supported.
<code>optimizer.momentum</code>	The momentum of the optimizer function, if supported.
<code>psm.type</code>	The type of the recurrent post sequence model (how the sequence of posts should be modeled). Available options are “RNN”, “LSTM”, or “GRU”.
<code>psm.hidden_size</code>	Size of the hidden state in the recurrent model.
<code>psm.output</code>	Name of the aggregation function that determines the sequence encoding from the hidden states of the recurrent network. Available options are “last_state”, “mean”, “max”, or “mean+max”.
<code>te.type</code>	Name of the aggregation function that determines the text document embedding from the CLS token (sentence) encodings. Available options are “mean” or “LSTM”.
<code>te.embedding_size</code>	Size of the text embedding. Only applies, if “te.type” is “LSTM” (equates the hidden size of the LSTM).

Table 6.1: Available hyperparameter and their description. Depending on their value they can change the model’s architecture or training behavior.

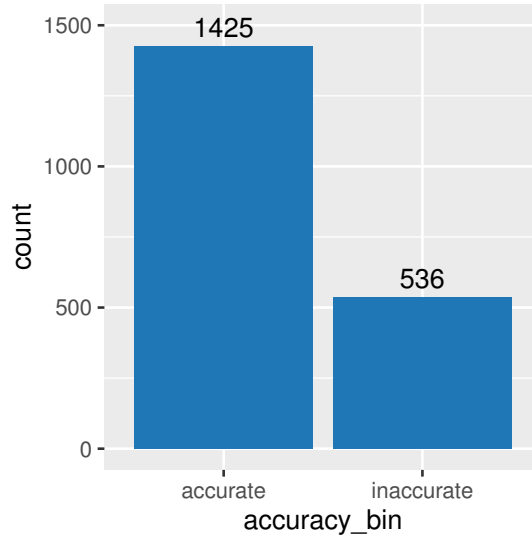


Figure 6.1: Number of news sources per label. The binary accuracy score is used.

In order to compare models trained with different parameters, each model uses the same training and evaluation dataset, respectively. We split the labeled dataset into a training set, which contains 80% of the instances, a validation, and a test set, both containing 10%, respectively. The split is performed in a stratified manner, such that the distribution of the target label in the original dataset, e.g. the accuracy score, is preserved in all three subsets. This means that we yield different data splits depending on the label we train the model on. For each parameter configuration we train the model on the training data and evaluate its performance on the validation data. Since the best performing model regarding the validation set does not guarantee to perform well in general, we validate the performance of the best model with the so far unseen test data [33, Sec. 7.2] [40, Sec. 4.7].

Furthermore, we simplified the training as much as possible. We first transformed the target labels into a binary label. For the accuracy score we identified a clear separation into two groups, since news sources publishing false information, clickbait news, or biased news share some level of inaccurate reporting, while the remaining two do not. Therefore, we transformed the existing accuracy score into a binary label as the following:

$$\text{accuracy}_{\text{bin}} = \begin{cases} \text{inaccurate} & \text{if accuracy} \leq 3 \\ \text{accurate} & \text{otherwise} \end{cases}$$

For the transparency score however such a separation cannot be justified, since news sources with transparency score 2 (“mixed transparency”) show an equal share of transparent and nontransparent behavior. Thus, we will focus only on the binary accuracy label during the parameter tuning process. For this label we observed that its distribution is quite imbalanced, by almost factor 3, as shown in Figure 6.1. This imbalance causes an

underrepresentation of “inaccurate” news sources, which are the “interesting” instances we try to detect. Therefore, we used oversampling for the training set to populate its instances labeled as “inaccurate”, such that we yield an equal number of “accurate” and “inaccurate” news sources [14]. This was done by randomly duplicating training instances from the set of “inaccurate” news sources, where each news source is selected at least once.

We will compare the performances of our model configurations by using the precision on the “inaccurate” class, because we want our model to detect inaccurate sources with high certainty to not misclassify actual accurate sources. As counterpart to the precision on “inaccurate” we will look at the recall of “accurate” sources in order to identify models that misclassify “accurate” sources. Furthermore, we will report the overall accuracy and macro F1 score as additional estimate. All evaluation metrics are detailed in Section B.2.

6.1.2 Results

As mentioned before, a grid search combining all hyperparameter values would blow up the total number of training runs. Therefore, we conducted four grid searches until we ended with a sufficient configuration, each changing a different set of parameters, shown in Table 6.2. Some parameters remained constant during the tuning process, i.e. we did not try different values for them, which are shown in Table 6.3. We first tested different batch sizes with and without an oversampled training set. In either way, all models did not learn anything during training and always put all instances of the test set into the same class. Figure 6.2 shows the learning curves of all models using oversampling, which oscillate with different amplitude around the starting loss of approximately 0.7. In the second grid search shown in Table 6.2 we varied the learning rate and momentum of

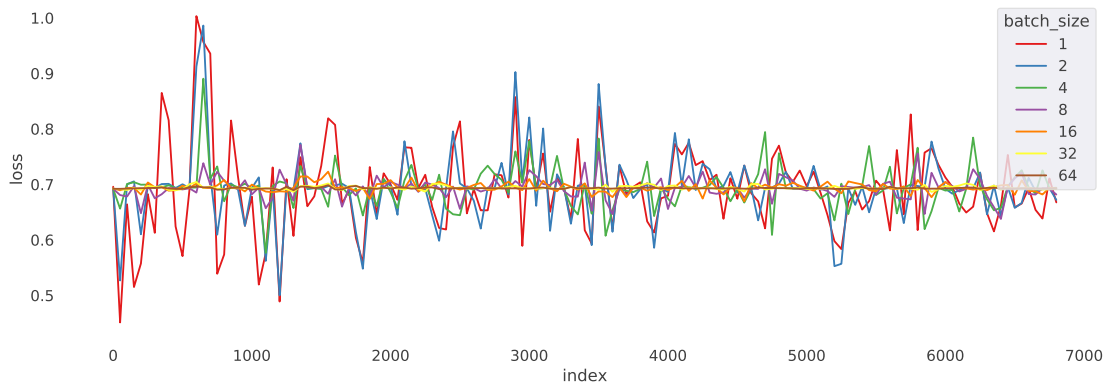


Figure 6.2: Loss curves of several models over training samples. Each trace represents a model using a different batch size. The other hyperparameters are "oversampling": true, "optimizer.name": "SGD", "optimizer.lr": 0.001, "optimizer.momentum": 0.9, "psm.output": "last_state" and the remaining set to the default value.

		optimizer			psm
oversampling	batch_size	name	lr	momentum	output
[false, true]	[1, 2, 4, 8, 16, 32, 64]	"SGD"	0.001	0.9	"last_state"
true	[1, 4, 16, 64]	"SGD"	[0.01, 0.001, 0.0001]	[0.0, 0.5, 0.9]	"last_state"
true	[4, 8, 16, 32, 64]	"Adam"	[0.01, 0.001, 0.0001]	NA	"last_state"
true	[4, 16, 32]	"Adam"	0.0001	NA	["mean", "max", "mean+max"]

Table 6.2: Conducted Grid Searches. Each row displays one grid search with the varied values for each parameter, displayed in the column.

Hyperparameter	Default Value
n_epochs	3
loss_fn	"CrossEntropyLoss"
psm.type	"LSTM"
psm.hidden_size	1000
te.type	"mean"
te.embedding_size	null

Table 6.3: Default Hyperparameters during Grid Search

the stochastic gradient descent algorithm. We combined those values with a few batch size values from the first grid search, which also did not change the models' learning behaviors.

Therefore, we tried a different optimization function, "Adam", with a handful of batch size and learning rate combinations. "Adam" showed first improvements when combining it with a learning rate of 0.0001. Therefore we used this learning rate and tried in the last grid search different values for psm.output, which resulted in an even better

Hyperparameter	Model1	Model2
batch_size	16	32
oversampling	true	
n_epochs	3	
loss_fn	"CrossEntropyLoss"	
optimizer.name	"Adam"	
optimizer.lr	0.0001	
optimizer.momentum	null (not used for Adam optimizer)	
psm.type	"LSTM"	
psm.hidden_size	1000	
psm.output	"mean"	
te.type	"mean"	
te.embedding_size	null (not used for "mean" text encoder, equates the hidden size of the post sequence model)	

Table 6.4: Model configurations for "Model1" and "Model2". Both use similar settings with different batch size

performance on the validation set, as shown in Table 6.5. We can see that with regards to the precision on the "inaccurate" label the two best models are the ones using batch size 16 or 32, "mean" as psm.output function, oversampling, and "Adam" optimizer with learning rate 0.0001. The full configurations of both models are shown in Table 6.4.

6.2 Performance Validations

In this section we will first compare the performance of those two models on the unseen test set and check whether they make significantly different predictions. We also test the predictions of the better model against the ones of two random models. After that we keep the better model and investigate its robustness by computing confidence intervals. Furthermore, we conduct a 10-fold cross-validation to yield a reliable evaluation of the final model configuration. Finally, we test this model configuration in three different settings, namely (i) training without oversampling, (ii) predicting the 5-point accuracy score, and (iii) predicting the 3-point transparency score.

batch size	psm output	total	accurate			inaccurate		
		accuracy	#preds	#true	recall	#preds	#true	precision
4	last state	0.71	97	143	0.64	99	53	0.48
	max	0.79	112	143	0.75	84	53	0.57
	mean	0.69	91	143	0.61	105	53	0.47
	mean + max	0.71	92	143	0.62	104	53	0.48
16	last state	0.76	112	143	0.73	84	53	0.54
	max	0.84	142	143	0.89	54	53	0.70
	mean	0.85	151	143	0.92	45	53	0.76
	mean + max	0.27	0	143	0.00	196	53	0.27
32	last state	0.67	86	143	0.57	110	53	0.45
	max	0.63	79	143	0.52	117	53	0.42
	mean	0.86	143	143	0.90	53	53	0.74
	mean + max	0.74	102	143	0.68	94	53	0.51

Table 6.5: Validation Set results after grid search. Each model is trained on the training set and evaluated on the validation set. Each row displays a different model configuration, where we varied the “batch size” and “psm output” parameter. All models use the Adam optimizer with a learning rate of 0.0001 and oversampling. The columns show the different evaluation metrics and the predicted and total number of instances per label class. We highlighted the best two precision scores regarding the “inaccurate” label.

	total		accurate			inaccurate		
	accuracy	macro F1	precision	recall	F1	precision	recall	F1
Model1	0.838	0.770	0.845	0.951	0.895	0.806	0.537	0.644
Model2	0.838	0.786	0.868	0.916	0.891	0.739	0.630	0.680

Table 6.6: Test Set results of “Model1” and “Model2”. Both models are trained on the training set and evaluated on the test set. In addition to the scores used during grid search, the table also reports F1 scores and the both precision and recall for all label classes.

6.2.1 Comparison of the Two Best Models

Before validating the performance of our model we need to choose one of the two best model configurations from the grid search. Let “Model1” be the model trained using a batch size of 16 and “Model2” trained with batch size 32, where “Model1” performed slightly better than “Model2” regarding the precision on the “inaccurate” label. We now want to find out how good the performance of both models can be generalized by evaluating their performance on the so far unseen test set. Table 6.6 shows that both models show similar or better performance than at the validation set. All scores achieved a very high significance ($p\text{-value} = 5 \times 10^{-5}$) after conducting a permutation test (see Section B.3). In Figure 6.3 one can see the confusion matrices of both models, which gives us more insights into the models’ performances. They show for instance that “Model2” detects more inaccurate sources, but also misclassify more accurate sources, which explains the lower precision than for “Model1”.

In order to test whether “Model1” makes significantly different predictions than “Model2”, we conduct a McNemar’s test [88]. For this purpose we counted the number of test instances, which were predicted correctly by both models, predicted incorrectly by both models, and predicted correctly by “Model1”, but not “Model2”, and vice versa. Those counts can be summarized in a contingency tables, as done in Table 6.7. One can see that respectively 6 instances are correctly classified by one model, but not the other. The McNemar’s test between both models results in a chi-square value of 0, which corresponds to a $p\text{-value}$ of 1. Thus, there is no significant difference in both models’ predictions. However, we keep “Model1” using a batch size of 16 as our final configuration. We further tested this model against two dummy models, “Dummy1” and “Dummy2”. The first dummy model randomly predicts both classes with equal probability, while the second model naively predicts the majority class of the test set, which is “accurate”. Both contingency tables are shown in Table 6.8. In either case, more test instances were correctly classified by “Model1”, but not by the dummy model, than were by the dummy model, but not our model. These differences are also highly significant, i.e. our model predictions are very likely not random guesses.

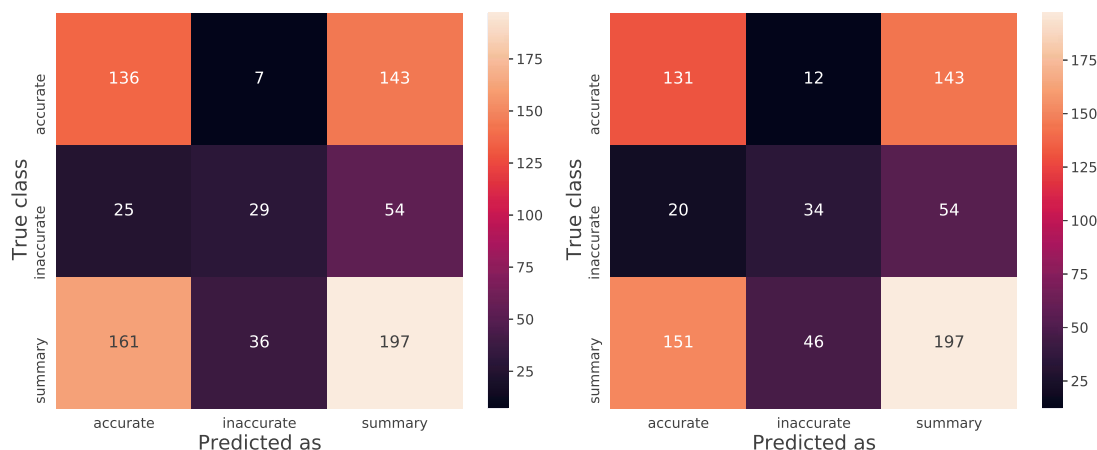


Figure 6.3: Confusion matrices of (a) “Model1” and (b) “Model2”. The cells display the number of news sources having the true label indicated by the row and been predicted as the label indicated by the column. The last row and column are the sum of the upper or left cells, displaying the total number predictions per class and the number of news sources per true label. The rightmost lowest cell is the total number of news sources in total.

	Model2 Correct	Model2 Incorrect
Model1 Correct	159	6
Model1 Incorrect	6	26

Table 6.7: Contingency table between “Model1” and “Model2” (p -value = 1). The cells indicate the number of intersecting news sources that have been predicted correctly or incorrectly by Model1 or Model2.

	Dummy1 Correct	Dummy1 Incorrect
Model1 Correct	136	29
Model1 Incorrect	7	25

	Dummy2 Correct	Dummy2 Incorrect
Model1 Correct	75	90
Model1 Incorrect	20	12

Table 6.8: Contingency tables between “Model1” and (a) “Dummy1” and (b) “Dummy2”. The p -values after a McNemar’s test are 4.74×10^{-11} and 4.653×10^{-4} , respectively. The cells indicate the number of intersecting news sources that have been predicted correctly or incorrectly by Model1 or the dummy models.

6.2.2 Stability of the Model’s Performance

Additionally, we want to analyze whether our scores are biased by the composition of our test set. We therefore compute confidence intervals for each metric using non-parametric bootstrapping (see Section B.3). We compute our metrics on each bootstrap sample and take the 2.5% and 97.5% quantiles of each statistic as our 95% confidence intervals, which are shown in Figure 6.4. The plot additionally shows the distribution over the bootstrap samples and the observed test statistic of the original test set. The confidence intervals show that our model has a stable overall accuracy and macro F1 score, as well as a small interval for precision and recall score on the “accurate” class. However, the model’s precision and recall on the “inaccurate” class is unstable, since both scores show a flat distribution and large confidence intervals. Depending on the test set, the precision on the “inaccurate” class is mostly between 0.67 and 0.93, approximately.

6.2.3 Final Results Using Cross Validation

Besides bootstrapping, we performed a 10-fold cross-validation on the full dataset, which gives us a more accurate estimation of the model’s performance. This technique separates the dataset randomly into 10 equal sized and distinct subsets. Again, each subset shares the same ratio of “accurate” and “inaccurate” news sources. Each fold is used once as a test set to evaluate a model trained on the remaining folds [78]. Since each news source is assigned to exactly one fold and each fold is evaluated once, we yield a prediction for all 1961 news sources, made by 10 different models. The estimated test statistics of each fold and the overall scores are shown in Figure 6.5. Again, each overall score shows a very high significance (see Figure B.2). Compared to the statistics measured in the single split test set in Table 6.6 the cross-validation predictions evaluated worse. The precision on the “inaccurate” label for instance is nearly half as good as on the single test set, i.e. around 0.436. This means that in absolute numbers more actual “accurate” than “inaccurate” news sources were predicted to be “inaccurate”. Also the overall accuracy and recall with regards to the “accurate” class have dropped by almost 0.2 and 0.4 percentage points, respectively. On the other side, the precision on the “accurate” and recall on the “inaccurate” classes have increased, the latter even by ≈ 0.3 percentage points. Both scores also show a smaller interval across the 10 folds than the remaining scores, except once, where the precision on the “accurate” news sources is close to 0. The confusion matrix of the cross-validation predictions in Figure 6.6 provides more information in greater detail. Almost 600 “accurate” news sources were misclassified, which explains the bad precision on the “inaccurate” class. The good precision on the “accurate” news sources is caused by only 76 “inaccurate” sources being misclassified as “accurate”.

6.3 Additional Applications

We evaluated our model configuration used in the previous experiments on three different settings. As training set we use the same data as for the parameter tuning and all models are evaluated on the test set.

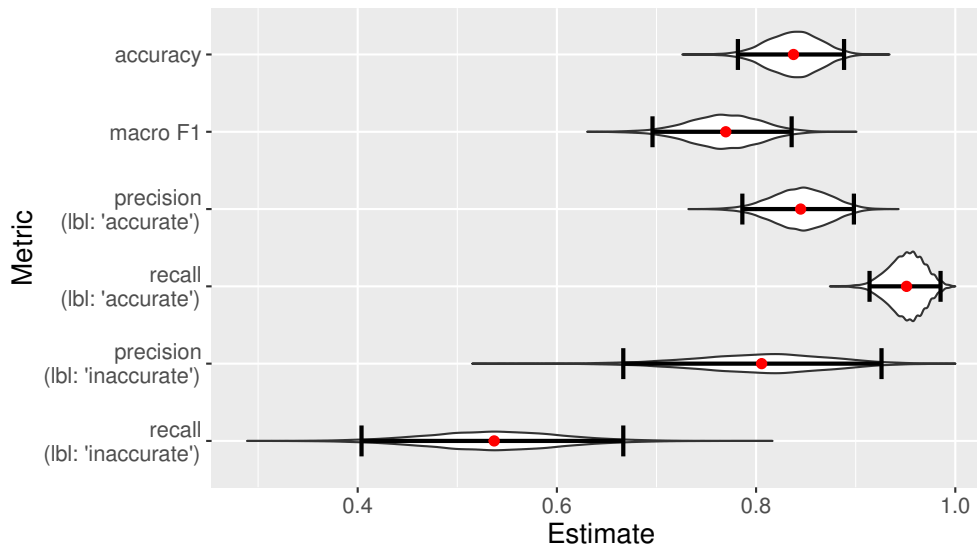


Figure 6.4: Test Statistic distributions of bootstrap samples. Each row displays the violin plot of one metric over the bootstrap samples, where precision and recall are reported per label class. The red dot marks the statistic of the original test set and black error bars indicate the 95% confidence interval.

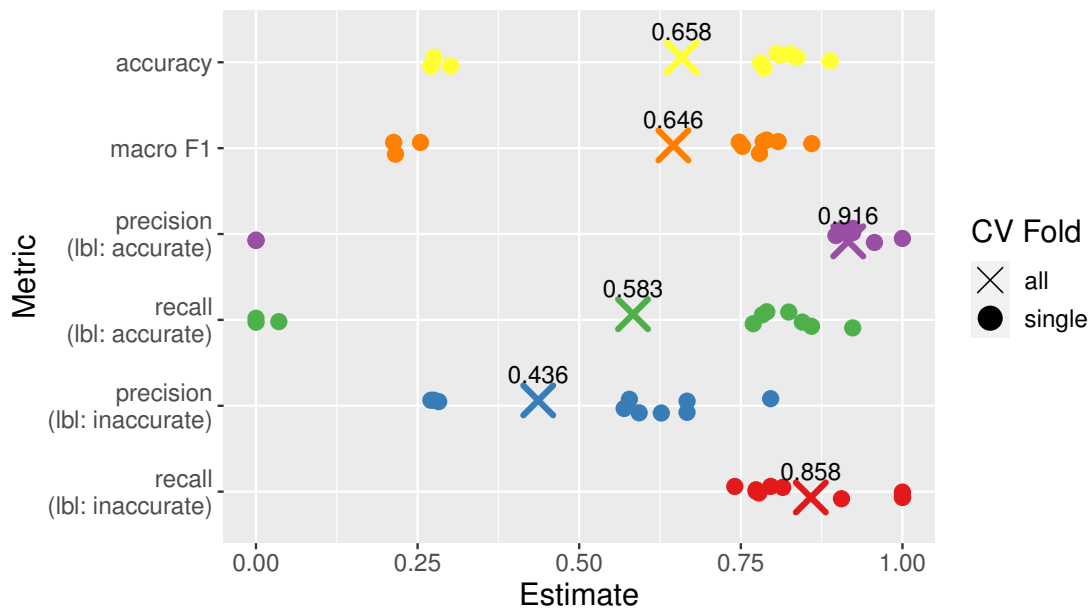


Figure 6.5: Test statistics after 10-fold cross-validation. Each row and color displays another metric, where precision and recall are reported per label class. Additional jitter prevents overlay of marks. The dots mark the metric of each fold, while the crosses indicate the overall estimate.

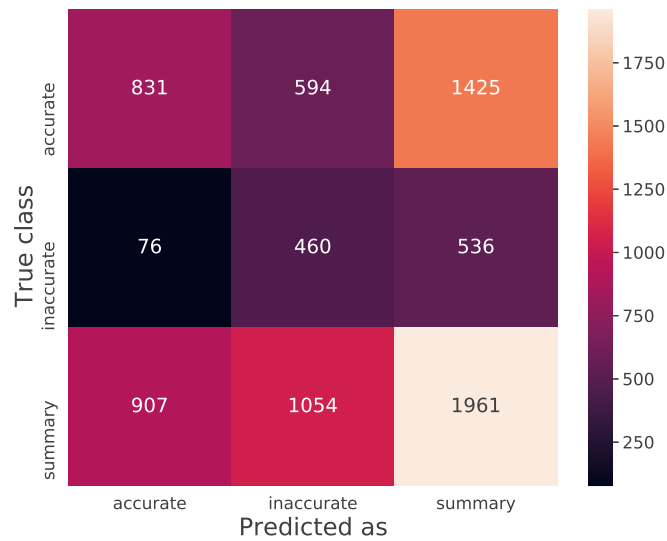


Figure 6.6: Confusion matrices of CV predictions. The cells display the number of news sources having the true label indicated by the row and been predicted as the label indicated by the column. The last row and column are the sum of the upper or left cells, displaying the total number predictions per class and the number of news sources per true label. The rightmost lowest cell is the total number of news sources in total.

In the first setting we predicted the binary accuracy label without oversampling the training data. So far, we oversampled instances from the minority class, i.e. “inaccurate”, in the respective set the model was trained on to prevent the model from missing the class of interest. The results without oversampling are shown in Table 6.9. For a better comparison we also included the results from Table 6.6. We can observe that the overall accuracy and macro F1 score dropped by ≈ 0.08 and 0.04 , respectively, while the precision on the “inaccurate” class shows a larger decrease by ≈ 0.26 . With a precision of ≈ 0.543 almost half of the predicted “inaccurate” sources are actually “accurate”. However, the precision on the “accurate” and recall on the “inaccurate” class have improved compared to training with oversampling.

In the two other settings we used our proposed labels from Section 4.1 as target label, i.e. the 5-point accuracy score and the 3-point transparency score. Both times we used oversampling, since we then expect a better precision on labels considered with inaccurate or nontransparent news sources. The observed evaluation metrics are shown in Table 6.10 and 6.11. Both times, the models achieved a better accuracy than random guessing, which would be 0.2 and 0.33 , respectively. However, trained on the accuracy label, the model has poor precision scores, especially on the classes 1, 2, and 3, which correspond to the binary “inaccurate” class. The model performance predicting the transparency scores also reveals bad precision scores on all transparency classes. In general, for the accuracy and transparency score, we can observe that the precision increases the more

accurate or transparent the news sources are. Another interesting observation can be made in the confusion matrix for the transparency score predictions shown in Figure 6.7. It reveals that almost no instance was predicted having “mixed transparency”, but the model predicts either “transparent” or “no transparency”. Another observation is that in absolute numbers more instances from classes 2 and 3 were predicted having transparency 1 than actually labeled as 1, which explains the low precision of 0.304.

over-sampling	total		accurate			inaccurate		
	accuracy	macro F1	precision	recall	F1	precision	recall	F1
no	0.761	0.735	0.914	0.741	0.819	0.543	0.815	0.652
yes	0.838	0.770	0.845	0.951	0.895	0.806	0.537	0.644

Table 6.9: Test Set results of “Modell1” with and without oversampling. Both models are trained on the training set and evaluated on the test set. The results with oversampling are the same as in Table 6.6.

Accuracy Score	precision	recall
1	0	0
2	0.1	0.25
3	0.289	0.464
4	0.530	0.136
5	0.528	0.857

	accuracy	macro F1
total	0.452	0.362

Table 6.10: Test Set results predicting accuracy scores. All models are trained on the training set and evaluated on the test set.

Transparency Score	precision	recall
1	0.304	0.75
2	0.5	0.027
3	0.544	0.674

	accuracy	macro F1
total	0.447	0.274

Table 6.11: Test Set results predicting transparency scores. All models are trained on the training set and evaluated on the test set.

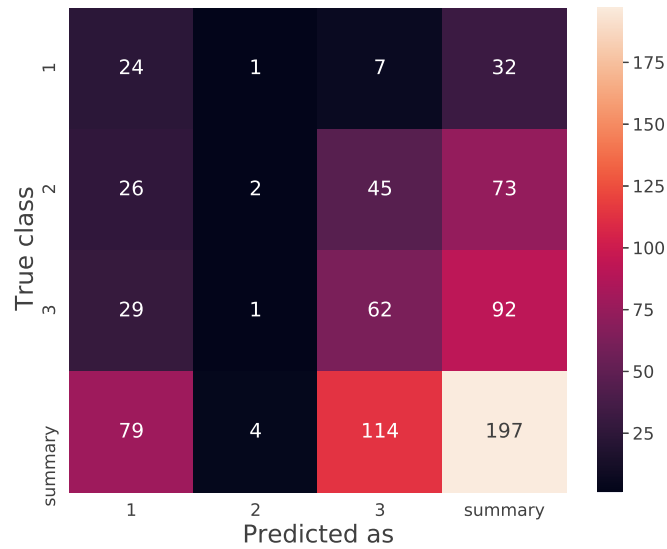


Figure 6.7: Confusion matrix of transparency score predictions. The cells display the number of news sources having the true label indicated by the row and been predicted as the label indicated by the column. The last row and column are the sum of the upper or left cells, displaying the total number predictions per class and the number of news sources per true label. The rightmost lowest cell is the total number of news sources in total.

Discussion and Conclusion

7.1 Limitations and Future Work

Summarizing the results, our model yielded an accuracy of $\approx 65.8\%$ and macro F1 score of ≈ 0.646 after a 10-fold cross-validation. However, it only achieves a precision of $\approx 43.6\%$ on the “inaccurate” label, meaning that more than half the detected “inaccurate” news sources are actually “accurate”. Also when using this model configuration to predict the 5-point accuracy or 3-point transparency score, the results could not keep up with the prior found results for binary classification. Besides the worse accuracy, the precision on the individual labels decreases the less accurate or transparent the news sources are. A similar observation can be made when focusing on the recall, where for instance accuracy score 5 has a much better recall than any other accuracy score. Nevertheless, we yield similar results to the baseline approach by Baly et al. [5], where the best configuration achieved an accuracy of 0.6863, and macro F1 score of 0.615, when predicting their 3-point factuality label using a 5-fold cross validation. When comparing these numbers to our results after a 10-fold cross-validation predicting the binary accuracy label, we yield a worse accuracy of ≈ 0.658 , but better macro F1 score of ≈ 0.646 . However, to validly compare our approach to Baly et al.’s baseline, we would need to train and test our model on the same news sources, including spending more time collecting data and extracting features from those additional sources.

A clear limitation of our approach is that we only focused on the precision regarding the “inaccurate” class. The precision is useful if we want to prevent the model from detecting false positives, but accept false negatives. This is done for instance when implementing an email spam filter, where harmless mails should not be classified as spam, but spam not being detected is acceptable [85]. In terms of false news source detection, a high number of false positives, i.e. misclassifying “accurate” news sources as “inaccurate”, is also not desired and dangerous, when no human experts validate the results, since it would restrict legitimate news. Therefore, a classifier, which is going to be applied in an automated way

at large scale without review of its prediction, needs a very high, nearly 100% precision on the class of interest. The precision of our model is $\approx 43.6\%$, which is far below of a concise detection of false information sources, since more than half of the predicted “inaccurate” news sources are actually accurate. Nevertheless, the results also show that the model makes more precise predictions for the more accurate/transparent classes. In the 10-fold cross-validation experiment the model achieved a $\approx 91.6\%$ precision regarding the “accurate” class. The precision for the accuracy scores 4, 5, and transparency scores 2, and 3 are all above 50%, although the model was not optimized for these tasks.

We derive from these observations that in future work one should focus more on the precision regarding classes representing accurate and transparent news. This change in focus enables a different application for the model. Instead of detecting unreliable news sources with high certainty, the model should rather be applied as filter for reliable, i.e. accurate and transparent, news sources. In practice, when considering the binary accuracy label, this would mean that all news sources classified as “inaccurate” would be candidates, which can be further reviewed by human experts to decide, whether they are truly “inaccurate”. A classifier labeling every website as a candidate would represent the status quo, because fact-checkers like Media Bias Fact Check [15] or NewsGuard [53] have to check every news source and assess their reliability. However, a classifier with high precision on the “accurate” news sources could exclude reliable news sources from the candidate list with high certainty, i.e. human experts do not need to review the sources’ reliability, since its very likely that they publish accurate news. Additionally, it could be beneficial to researchers that need a dataset of accurate news sources with high certainty to not contain unreliable news sources.

Another limitation of our approach is the fixed architecture of the model. For instance at the text encoder, we only tried to average the output vectors of the Pooler layers. Since the Pooler output is a representational vector of a sentence from a larger document, one could argue to aggregate the sequence of sentence representations using an RNN, LSTM, or GRU. Similarly, we only used an LSTM to aggregate the embedded posts with its author and article information, but could have tried a GRU.

Also our training shows some possibilities for improvement. We have conducted a grid search varying only 6 out of 12 identified hyperparameters. Some parameters would have caused small architectural changes, as described before. Others would have changed the dimension of hidden layers or embeddings. In addition, we only trained the models for 3 epochs, without trying longer training runs.

As we summarized in Chapter 3, there are many possibilities to extract features in order to detect false information. We have decided to extract features from article text, title, its meta description, the post text, and the user description. Additionally, we combined this information with meta features from the post and the users, as likes or number of followers. Future work could focus on the inclusion of additional features, as already used for fake news article classification. One possibility would be to use network information, like the follower relations of the posting user or the propagation graph of the news article spread. In the original approach by Shu et al. [71] the authors construct a diffusion graph

for each news article, where each user in the graph has shared the article and has at least one following relation to any other user in the graph. In our approach we could not construct such a diffusion graph for two reasons: First, we collected posts from two social media platforms, namely Facebook and Twitter, where no following relationship between the platforms can be identified. Second, we sought a representation vector for a whole news source, instead of a single article. When having two users from the same platform posting different articles from a common news source, one could not argue that the news propagated from one to the other user, even if both have a following relationship. However, studies have shown that the diversity of news that users are exposed to on social media platforms plays a crucial role in their opinion making. Users having a selective and homogeneous news exposure are likely to form echo chambers, where their views are reinforced and contradictory information is denied [6, 69, 84]. Knowing about the type of news sources a given user is exposed to could therefore be a valuable information and promising future work.

Another type of feature having large potential for the reliability classification of news sources are stances or sentiments regarding a certain news source. These could be extracted from the post text itself or the comments a post receives. Research regarding the detection of fake news articles by using sentiments or text features extracted from comments already exist [19, 72], which is why it would be reasonable to use them in automatic news source reliability classification. Finally, one could include more information about the posting user. While we have composed the user representation using its self description, follower count, and whether or not it is a verified user, one could use additional features representing the behavior or credibility of a user [95]. For instance, one could include features extracted by Botometer [68], which is an approach that detects social bot accounts on Twitter. These features tell whether a Twitter account is a known bot or has a suspicious following behavior. In general, bots play an important role in the targeted spread of false information [3].

7.2 Summary

In this thesis, we have proposed an automated framework based on Machine Learning that estimates the reliability of news sources. While prior research focused either on detecting fake news articles or assessing the credibility of sources using content information from its articles, we include information collected from a source's articles and social media. Since the latter plays an important role in spreading news in general, we extracted features from social media posts and users who shared an article from a given news source. We trained the model using various news source reliability ratings compiled by human experts. Although our model is not precise enough to detect false information sources with high certainty, we observed that it can detect accurate news sources with a precision of approximately 91.6%, even without specific optimization regarding this issue. With further optimization focusing on this task, human experts, who manually assess the reliability of news sources, could use our approach to exclude accurate news sources to make their work more efficient. It could also be an attractive tool for researchers,

7. DISCUSSION AND CONCLUSION

who use data linked to news sources and need to be certain about using only accurate ones. Furthermore, promising future work should not only focus on the optimization of detecting accurate news sources, but could also extract additional features by including information like the type of news sources users are exposed to or the reliability of users themselves.

Fact-Checking Websites

A.1 Bufalopedia

Translated with www.DeepL.com/Translator (free version).

Table A.1: Bufalopedia Labels (as of May 2020) [9]

Label	Description	A	T
misleading	-	2	1
satire	“Sites/accounts that are fairly clearly recognizable as satire or parody, but still occasionally claim victims”	1	2
fake news	“It is also to be considered suspicious, and at high risk of hoax, any news coming from names like these”	1	1

A.2 Bufale

Translated with www.DeepL.com/Translator (free version).

Table A.2: The Black List by Bufale (as of 2018) [8]

Label	Description	A	T
false satire	“Some of them have been around for quite some time, but in 2015 there has been a considerable increase in ‘fake newspapers’, or rather those domains that have misled many users because of their names that are easily confused with the most famous newspapers. Many of these claim to be ‘satirical’ even if, in some cases, they have nothing to do with ‘satire’. Many of these sites, however, have not been updated for some time, due to the fact that today, most users have learned to recognize ‘fake newspapers’”.	1	1
hoaxes and medical and scientific disinformation	-	1	1
political disinformation	-	1	1
religious and racially motivated disinformation	-	1	1
scandalous hoaxes	-	1	1
political and racial hoax	-	1	1
clickbait	“News aggregators, copy pasters, with no fact checking and catchy headlines to attract the reader.”	2	2
websites of conspiracy theorists and hoax spreaders	“New World Order, chemtrails, aliens, 9/11... you name it.”	1	1
social media sites of conspiracy theorists and hoax spreaders	-	1	1

A.3 Butac

Translated with www.DeepL.com/Translator (free version).

Table A.3: The Black List by Butac (as of May 2021) [10]

Label	Description	A	T
close to Qanon	“The sites (Italian and not) close to the disinformation of QAnon”	1	1
pseudoscience	“Sites that propose experiments, inventions, treatments that have no scientific merit or insufficient evidence”	1	1
pseudojournalism and politics	“In this black list there are sites that use the language of journalism to write articles often based on nothing that bring water to their mill, whatever it is. It can be about politics, religion, ideals. They are sites that hurt, because many of those who read them are there because they are like-minded, and they can’t distinguish between real news and news fueled only by cheering for this or that faction, or they copy news from unreliable sources without doing any verification.”	1	1
pseudojournalism blogs	-	1	1
pseudoscience blogs	-	1	1
pseudomedicine	“Actually this black list should be much longer, but almost always the other sites that publish news of pseudo medicine have as source one of the following pages. These are dangerous pages, because they peddle cures that have no foundation, often recommending books and treatments - for a fee - sold and written by the same people who invented these miraculous cures.”	1	1
conspiracy	“The pages in this black list believe in one or another conspiracy theory, assuming that what mainstream information says is false. I have not yet understood if the minds behind these pages are ‘accomplices’ of those who invented these theories for profit or if they are just other ‘cheated’”	1	1
conspiracy blogs	-	1	1

A.4 BuzzFeed News

Table A.4: Partisan News Analysis by BuzzFeed News (as of August 2017) [50, 51]

Label	Description	A	T
left bias	-	3	2
right bias	-	3	2

A.5 Politifact

Table A.5: PolitiFact's Fake news almanac (as of November 2017) [61]

Label	Description	A	T
Imposter site	“Adding to the fog of fake news online, several websites appear to try to confuse readers into thinking they are the online outlets of traditional or mainstream media sources. These sites attempt to trick readers into thinking they are newspapers or radio or television stations. Like many other fake news sites, it’s very difficult to see who owns them, thanks to private registrations.”	1	1
Fake news	“There’s little consistency of content or style among fake news sites – the common thread appears to be that they distribute fabricated content, but the reasons aren’t always apparent.”	1	1
Parody site	“It can be difficult to parse what the aims of these parody sites may be. They could be writing stories purely for entertainment, or they may be trolling a particular set of voters. These websites may or may not make it clear on individual links that their stories are fake, but will almost always say in a disclaimer somewhere on their site that their content is exaggerated or fictional.”	1	2
Some fake stories	“We have checked statements from sites that appear to carry real content, but contained one or more fake news stories. We don’t know whether this is intentional or accidental, but it happens.”	2	1

A.6 Columbia Journalism Review

We accessed the list in July 2021. Unfortunately the website is not available anymore and the Google Spreadsheet was not indexed¹.

Table A.6: Columbia Journalism Review Labels (as of April 2021) [64]

Label	Description	A	T
bias	-	3	2
unreliable	-	2	1
fake	-	1	1
conspiracy	-	1	1
satire	-	1	2
clickbait	-	2	2

A.7 Fake News Watch

Table A.7: Fake News Watch Labels (as of January 2016) [91]

Label	Description	A	T
fake/hoax	“Fake/Hoax News sites are satire sites that are not funny. They are an attempt to play on gullible people who do not check sources and will just pass the news on as if it were really true.”	1	1
satire	“Satire websites are sites that make fun of the news. The stories are typically over the top and meant to be funny.”	1	2
clickbait	“Clickbait websites are sites that take bits of true stories but insinuate and make up other details to sew fear. Most of these are conspiratorial in nature are very unreliable.”	2	2

¹The current version of the Spreadsheet is still available at https://docs.google.com/spreadsheets/d/1ck1_FZC-97uDLI1vRJDTrGqBk0FuDe9yHkluROgpGS8/edit#gid=1728720316

A.8 Media Bias Fact Check

Table A.8: Media Bias Fact Check Labels (as of May 2021) [15]

Label	Description	A	T
questionable sources	“A questionable source exhibits one or more of the following: extreme bias, consistent promotion of propaganda/conspiracies, poor or no sourcing to credible information, a complete lack of transparency and/or is fake news. Fake News is the deliberate attempt to publish hoaxes and/or disinformation for the purpose of profit or influence. Sources listed in the Questionable Category may be very untrustworthy and should be fact checked on a per article basis. Please note sources on this list are not considered fake news unless specifically written in the reasoning section for that source.”	1	1
conspiracy / pseudoscience	“Sources in the Conspiracy-Pseudoscience category may publish unverifiable information that is not always supported by evidence. These sources may be untrustworthy for credible/verifiable information, therefore fact checking and further investigation is recommended on a per article basis when obtaining information from these sources.”	1	1
right bias	“These media sources are moderately to strongly biased toward conservative causes through story selection and/or political affiliation. They may utilize strong loaded words (wording that attempts to influence an audience by using appeal to emotion or stereotypes), publish misleading reports and omit reporting of information that may damage conservative causes. Some sources in this category may be untrustworthy.”	3	2
center right bias	“These media sources are slightly to moderately conservative in bias. They often publish factual information that utilizes loaded words (wording that attempts to influence an audience by using appeal to emotion or stereotypes) to favor conservative causes. These sources are generally trustworthy for information, but may require further investigation.”	4	3

left center bias	“These media sources have a slight to moderate liberal bias. They often publish factual information that utilizes loaded words (wording that attempts to influence an audience by using appeal to emotion or stereotypes) to favor liberal causes. These sources are generally trustworthy for information, but may require further investigation.”	4	3
left bias	“These media sources are moderately to strongly biased toward liberal causes through story selection and/or political affiliation. They may utilize strong loaded words (wording that attempts to influence an audience by using appeal to emotion or stereotypes), publish misleading reports and omit reporting of information that may damage liberal causes. Some sources in this category may be untrustworthy.”	3	2
least biased	“These sources have minimal bias and use very few loaded words (wording that attempts to influence an audience by using appeal to emotion or stereotypes). The reporting is factual and usually sourced. These are the most credible media sources.”	5	3
satire	“These sources exclusively use humor, irony, exaggeration, or ridicule to expose and criticize people’s stupidity or vices, particularly in the context of contemporary politics and other topical issues. Primarily these sources are clear that they are satire and do not attempt to deceive.”	1	2
pro science	“These sources consist of legitimate science or are evidence based through the use of credible scientific sourcing. Legitimate science follows the scientific method, is unbiased and does not use emotional words. These sources also respect the consensus of experts in the given scientific field and strive to publish peer reviewed science. Some sources in this category may have a slight political bias, but adhere to scientific principles.”	5	3

A.9 Melissa Zimdars – Washington Post

Table A.9: Washington Post’s Fake news list (as of 2016) [62, 98]

Label	Description	A	T
bias	“Sources that come from a particular point of view and may rely on propaganda, decontextualized information, and opinions distorted as facts.”	3	2
fake	“Sources that entirely fabricate information, disseminate deceptive content, or grossly distort actual news reports.”	1	1
conspiracy	“Sources that are well-known promoters of kooky conspiracy theories. Ex: 9/11 conspiracies, chem-trails, lizard people in the sewer systems, birther rumors, flat earth ‘theory’, fluoride as mind control, vaccines as mind control etc.”	1	1
unreliable	“Sources that may be reliable but whose contents require further verification or to be read in conjunction with other sources.”	2	1
satire	“Sources that use humor, irony, exaggeration, ridicule, and false information to comment on current events.”	1	2
hate	“Sources that actively promote racism, misogyny, homophobia, and other forms of discrimination.”	-	-
clickbait	“Sources that provide generally credible content, but use exaggerated, misleading, OR questionable headlines, social media descriptions, and/or images. These sources may also use sensational language to generate interest, clickthroughs, and shares, but their content is typically verifiable.”	2	2
political	“Sources that provide generally verifiable information in support of certain points of view or political orientations.”	3	2
junksci	“Sources that promote pseudoscience, metaphysics, naturalistic fallacies, and other scientifically dubious claims.”	1	1
rumor	“Sources that traffic in rumors, gossip, innuendo, and unverified claims.”	2	2

unknown	“Sources that have not yet been analyzed (many of these were suggested by readers/users or are found on other lists and resources).”	-	-
reliable	“Sources that circulate news and information in a manner consistent with traditional and ethical practices in journalism (Remember: even credible sources sometimes rely on clickbait-style headlines or occasionally make mistakes. No news organization is perfect, which is why a healthy news diet consists of multiple sources of information).”	5	3
state	“Sources in repressive states operating under government sanction.”	3	2

A.10 Newsguard

Table A.10: NewsGuard Credibility Criteria (as of September 2021) [53]. A numeric score of each criteria is noted in the parentheses.

ID	Label	Description
c_1	Does not repeatedly publish false content (22)	“The site does not repeatedly produce stories that have been found—either by journalists at NewsGuard or elsewhere—to be clearly and significantly false, and which have not been quickly and prominently corrected.”
c_2	Gathers and presents information responsibly (18)	“Content providers are generally fair and accurate in reporting and presenting information. They reference multiple sources, preferably those that present direct, firsthand information on a subject or event or from credible second hand news sources, and they do not egregiously distort or misrepresent information to make an argument or report on a subject.”
c_3	Regularly corrects or clarifies errors (12.5)	“The site makes clear how to report an error or complaint, has effective practices for publishing clarifications and corrections, and notes corrections in a transparent way.”
c_4	Handles the difference between news and opinion responsibly (12.5)	“Content providers who convey the impression that they report news or a mix of news and opinion distinguish opinion from news reporting, and when reporting news, do not egregiously cherry pick facts or stories to advance opinions. Content providers who advance a particular point of view disclose that point of view.”
c_5	Avoids deceptive headlines (10)	“The site generally does not publish headlines that include false information, significantly sensationalize, or otherwise do not reflect what is actually in the story.”

Table A.11: NewsGuard Transparency Criteria (as of September 2021) [53]. A numeric score of each criteria is noted in the parentheses.

ID	Label	Description
t_1	Website discloses ownership and financing (7.5)	“The site discloses its ownership and/or financing, as well as any notable ideological or political positions held by those with a significant financial interest in the site, in a user-friendly manner.”
t_2	Clearly labels advertising (7.5)	“The site makes clear which content is paid for and which is not.”
t_3	Reveals who’s in charge, including possible conflicts of interest (5)	“Information about those in charge of the content is made accessible on the site.”
t_4	The site provides the names of content creators, along with either contact or biographical information (5)	“Information about those producing the content is made accessible on the site.”

Table A.12: Rules to derive Accuracy labels from NewsGuard Credibility Criteria. The IDs of the criteria are used as propositional variables, i.e. $c_i = 1$, if criterion c_i is satisfied, and $c_i = 0$ otherwise. Have to be applied top-down.

	Rule	Accuracy
if	$\bigwedge_{i=1}^5 c_i$	5
else if	$c_1 \wedge c_2 \wedge \left(\sum_{i=3}^5 c_i = 2 \right)$	4
else if	$c_1 \wedge \left(\sum_{i=2}^5 c_i \geq 2 \right)$	3
else if	$\sum_{i=1}^5 c_i = 2$	2
else if	$\sum_{i=1}^5 c_i \leq 1$	1

Table A.13: Rules to derive Transparency labels from NewsGuard Transparency Criteria. The IDs of the criteria are used as propositional variables, i.e. $t_i = 1$, if criterion t_i is satisfied, and $t_i = 0$ otherwise. Have to be applied top-down.

	Rule	Transparency
if	$\bigwedge_{i=1}^4 t_i$	3
else if	$2 \leq \sum_{i=1}^4 t_i \leq 3$	2
else if	$\sum_{i=1}^4 t_i \leq 1$	1

Supplementary Material

B.1 Additional Figures for Model Implementation

```
{  
  "_name_or_path": "roberta-base",  
  "architectures": ["RobertaForMaskedLM"],  
  "attention_probs_dropout_prob": 0.1,  
  "bos_token_id": 0,  
  "classifier_dropout": null,  
  "eos_token_id": 2,  
  "hidden_act": "gelu",  
  "hidden_dropout_prob": 0.1,  
  "hidden_size": 768,  
  "initializer_range": 0.02,  
  "intermediate_size": 3072,  
  "layer_norm_eps": 1e-05,  
  "max_position_embeddings": 514,  
  "model_type": "roberta",  
  "num_attention_heads": 12,  
  "num_hidden_layers": 12,  
  "pad_token_id": 1,  
  "position_embedding_type": "absolute",  
  "transformers_version": "4.13.0",  
  "type_vocab_size": 1,  
  "use_cache": true,  
  "vocab_size": 50265  
}
```

Listing B.1: Configuration for RoBERTa used in the Text Encoder

B.2 Evaluation Metrics

In order to quantify and compare the model performances among each other we use the following metrics [40, Sec. 4.7]: The *accuracy* of a model tells about the percentage of test instances that were labeled correctly. As an overall metric it can be well used as a first estimate of the model’s performance. However, optimizing the accuracy alone does not always yield a desired model, as described in [81]. A model predicting more false positives than true positives can increase its accuracy by simply not predicting the positive class anymore. Since such a naive classifier is not desired, we need other metrics accounting this issue. One of these alternatives is the *precision*. It is calculated per class c and reports the percentage of correct predictions among all instances predicted as c . It therefore penalizes for classifying instances wrongly as c and gives a high reliability on those instances predicted as class c . The other metric is the *recall* of a class c , which can be seen as counterpart to the precision. It measures the percentage of instances belonging to c , which were also detected as such by the model. It therefore penalizes for missing (not detecting) instances that actually belong to class c . It depends on the use case, whether it is beneficial to optimize the model on precision or recall. Therefore, the F1 score is a combined metric consisting of precision and recall regarding a given label as follows:

$$\frac{2 \cdot \textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}}$$

The macro F1 score reports the average over the F1 scores of each label.

B.3 Permutation Test and Bootstrapping

We tested the significance of all measured performance criteria using a permutation test. The null hypothesis H_0 is that our model predictions were randomly assigned to the true labels, which resulted in the respective test statistics. The alternative hypothesis H_1 is that the observed accuracy, precision, and recall scores are better than if observed on random predictions. Let $\{(x_1, y_1), \dots, (x_n, y_n)\}$ be our test set containing n news sources x_i with ground truth label y_i and \hat{y}_i be the predicted label of our model. We draw each sample under H_0 , i.e. we take a prediction \hat{y}_i by random without replacement and assign it to the test instance (x_1, y_1) , continuing with (x_2, y_2) and so on. Let $\theta_1, \dots, \theta_m$ be a particular test statistic on m permutation samples and θ_0 the observed value of the original test set. We then compute an one-sided p-value as the following:

$$p = \frac{\sum_{i=1}^m I_{\geq \theta_0}(\theta_i) + 1}{m}$$

where $I_{\geq \theta_0}(x) = 1$ if $x \geq \theta_0$ and 0 otherwise. The distribution of each test statistic $\theta_1, \dots, \theta_m$ using $m = 20,000$ samples is shown in Figure B.1, where the red vertical line indicates θ_0 . We can see that none of the permuted samples achieves a better metric than we actually observed, therefore all p-values are $1/20,000 = 0.00005$ and we can reject H_0 with high significance [28].

Besides permutation tests we also used bootstrapping to validate our results and make statements about the stability of a measured statistic. Non-parametric bootstrapping enables us to simulate different test sets consisting of news sources from the original test set and the respective model predictions. Each test set in a simulation run is a sample of the original test set containing n triples (x_i, y_i, \hat{y}_i) , which were taken randomly with replacement [21]. Compared to the samples used in the permutation test, each labeled news sources is assigned with the actual model prediction, as if the model were making new predictions for each sample.

B.4 Additional Figures for Performance Validation

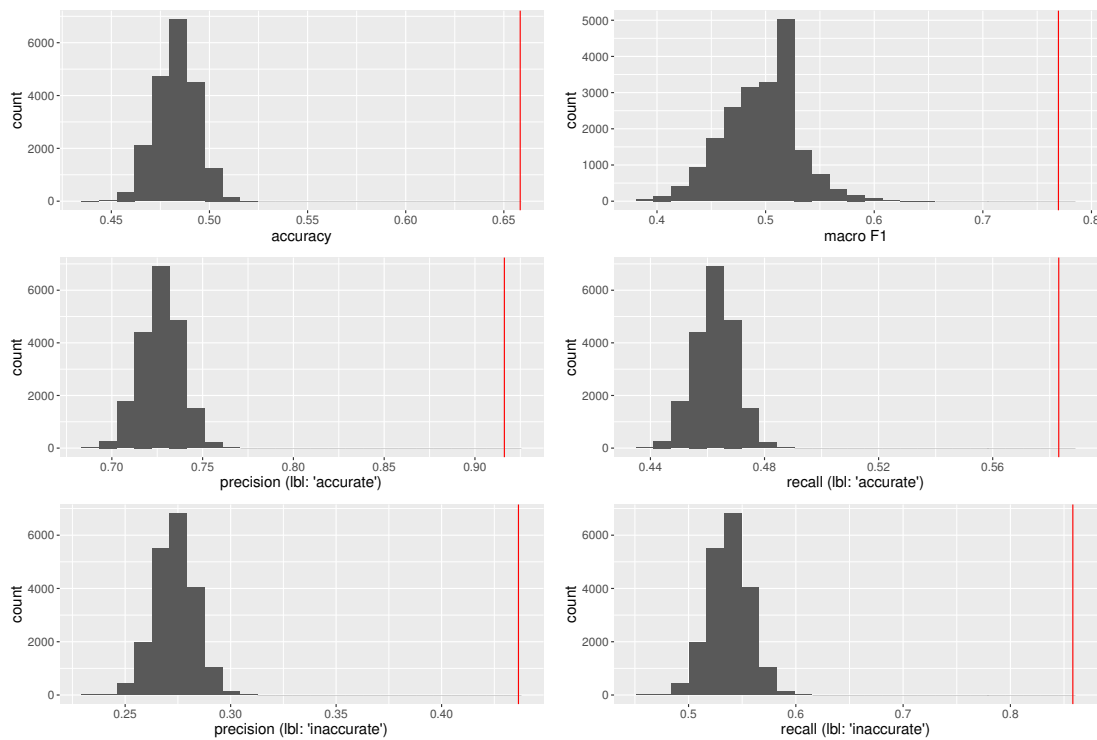


Figure B.1: Evaluation metric distribution over permutation samples of the test split. From left to right, top to bottom, the figures show the distribution of the (i) accuracy, (ii) macro F1, (iii) precision and (iv) recall of the “accurate” label, and (v) precision and (vi) recall of the “inaccurate” label. The red line indicates the respective metric on the original test set.

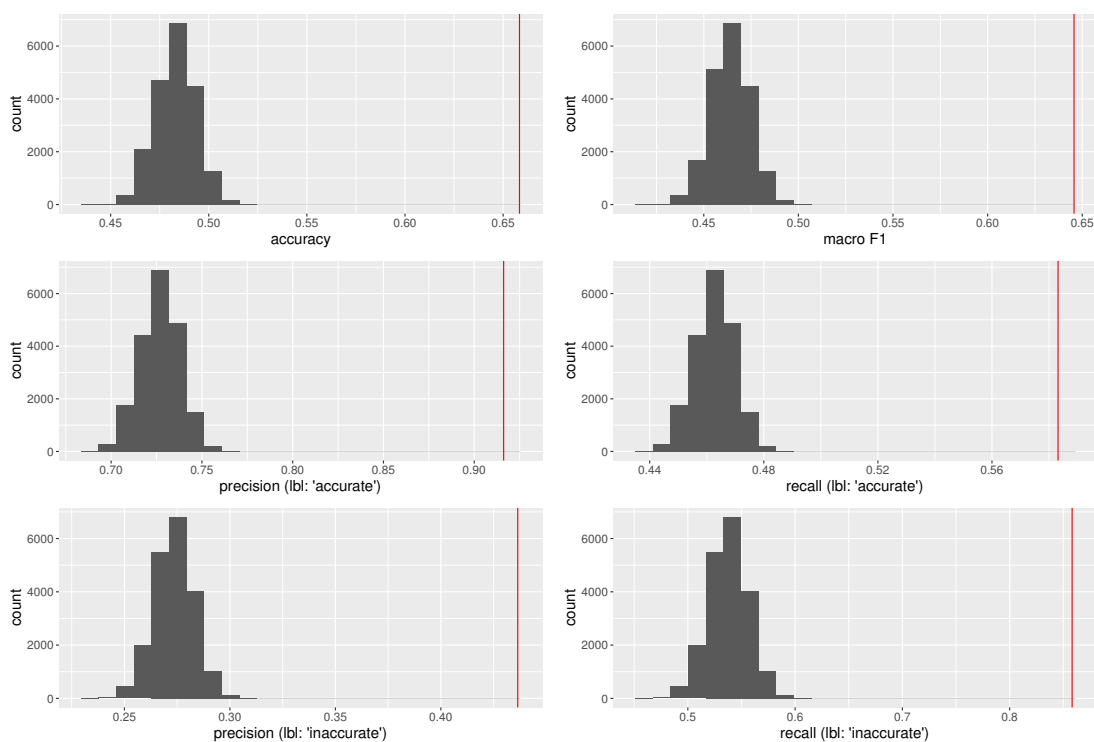


Figure B.2: Evaluation metric distribution over permutation samples of the 10-fold cross validation prediction. From left to right, top to bottom, the figures show the distribution of the (i) accuracy, (ii) macro F1, (iii) precision and (iv) recall of the “accurate” label, and (v) precision and (vi) recall of the “inaccurate” label. The red line indicates the respective metric on the original test set.

List of Figures

2.1	Transformer Block mapping embedding x_n to y_n using its preceding words [40, Sec. 9.7]	7
2.2	Multihead attention layer using 4 heads [40, Sec. 9.7]	8
2.3	BERT Architecture during pre-training and fine-tuning	9
2.4	Family of pre-trained language models and their relation to BERT in 2019 [90]	10
3.1	Available hand-crafted features for detecting false information [7]	13
3.2	Framework by Baly et al. [5] to predict political bias and factuality of news sources. For each news source (medium) 5 entities are scraped, from which features are extracted. Features inside curly brackets {} are averaged per news source.	21
4.1	Number of news sources per accuracy and transparency label. Each bar additionally shows the share of inactive news sources, i.e. news sources not responding with HTTP status code 200.	27
4.2	Number of news sources per accuracy and transparency label after data cleaning. Each news source is required to have at least 5 articles, each being shared in at least 5 posts on social media.	32
4.3	Number of distinct articles, posts, and users per accuracy and transparency label.	32
5.1	RNN framework for learning news temporal representations. Each \mathbf{x}_i represents the features associated with a post sharing an article on social media. By further embedding it into $\tilde{\mathbf{x}}_i$ and passing the sequence to an RNN, we yield an representational vector \mathbf{h}_m usable for further classification tasks. [71]	34
5.2	Text Encoder module transforming text into numerical features. After tokenizing the input text into its sentences, those are embedded using RoBERTa, but only the first token embedding will be further processed using the Pooler Layer, respectively. The trainable elements are marked green.	37
5.3	Post Embedder concatenating all available features. Features from articles are marked red, those from posts blue, and from users yellow. Trainable components are marked green.	38
		75

5.4	Post sequence model classifying news sources. The upper part shows a sample data structure consisting of 3 engagements. In the lower part features for every entity are extracted and embedded using the Post Embedder. . . .	40
6.1	Number of news sources per label. The binary accuracy score is used. . .	43
6.2	Loss curves of several models over training samples. Each trace represents a model using a different batch size. The other hyperparameters are "oversampling": true, "optimizer.name": "SGD", "optimizer.lr": 0.001, "optimizer.momentum": 0.9, "psm.output": "last_state" and the remaining set to the default value.	44
6.3	Confusion matrices of (a) "Model1" and (b) "Model2". The cells display the number of news sources having the true label indicated by the row and been predicted as the label indicated by the column. The last row and column are the sum of the upper or left cells, displaying the total number predictions per class and the number of news sources per true label. The rightmost lowest cell is the total number of news sources in total.	49
6.4	Test Statistic distributions of bootstrap samples. Each row displays the violin plot of one metric over the bootstrap samples, where precision and recall are reported per label class. The red dot marks the statistic of the original test set and black error bars indicate the 95% confidence interval.	51
6.5	Test statistics after 10-fold cross-validation. Each row and color displays another metric, where precision and recall are reported per label class. Additional jitter prevents overlay of marks. The dots mark the metric of each fold, while the crosses indicate the overall estimate.	51
6.6	Confusion matrices of CV predictions. The cells display the number of news sources having the true label indicated by the row and been predicted as the label indicated by the column. The last row and column are the sum of the upper or left cells, displaying the total number predictions per class and the number of news sources per true label. The rightmost lowest cell is the total number of news sources in total.	52
6.7	Confusion matrix of transparency score predictions. The cells display the number of news sources having the true label indicated by the row and been predicted as the label indicated by the column. The last row and column are the sum of the upper or left cells, displaying the total number predictions per class and the number of news sources per true label. The rightmost lowest cell is the total number of news sources in total.	54
B.1	Evaluation metric distribution over permutation samples of the test split. From left to right, top to bottom, the figures show the distribution of the (i) accuracy, (ii) macro F1, (iii) precision and (iv) recall of the "accurate" label, and (v) precision and (vi) recall of the "inaccurate" label. The red line indicates the respective metric on the original test set.	73

B.2 Evaluation metric distribution over permutation samples of the 10-fold cross validation prediction. From left to right, top to bottom, the figures show the distribution of the (i) accuracy, (ii) macro F1, (iii) precision and (iv) recall of the “accurate” label, and (v) precision and (vi) recall of the “inaccurate” label. The red line indicates the respective metric on the original test set.

74



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

4.1	Accuracy scores	24
4.2	Transparency scores	24
5.1	Combined features for Tweets and Facebook Posts. The right column show how those features are compiled using original platform features.	36
5.2	Combined features for Twitter users and Facebook accounts.	36
5.3	Article features we keep. Formerly extracted by Newspaper3k [58].	36
6.1	Available hyperparameter and their description. Depending on their value they can change the model’s architecture or training behavior.	42
6.2	Conducted Grid Searches. Each row displays one grid search with the varied values for each parameter, displayed in the column.	45
6.3	Default Hyperparameters during Grid Search	45
6.4	Model configurations for “Model1” and “Model2”. Both use similar settings with different batch size	46
6.5	Validation Set results after grid search. Each model is trained on the training set and evaluated on the validation set. Each row displays a different model configuration, where we varied the “batch size” and “psm output” parameter. All models use the Adam optimizer with a learning rate of 0.0001 and oversampling. The columns show the different evaluation metrics and the predicted and total number of instances per label class. We highlighted the best two precision scores regarding the “inaccurate” label.	47
6.6	Test Set results of “Model1” and “Model2”. Both models are trained on the training set and evaluated on the test set. In addition to the scores used during grid search, the table also reports F1 scores and the both precision and recall for all label classes.	48
6.7	Contingency table between “Model1” and “Model2” (p-value = 1). The cells indicate the number of intersecting news sources that have been predicted correctly or incorrectly by Model1 or Model2.	49
6.8	Contingency tables between “Model1” and (a) “Dummy1” and (b) “Dummy2”. The p-values after a McNemar’s test are 4.74×10^{-11} and 4.653×10^{-4} , respectively. The cells indicate the number of intersecting news sources that have been predicted correctly or incorrectly by Model1 or the dummy models.	49
		79

6.9	Test Set results of “Modell” with and without oversampling. Both models are trained on the training set and evaluated on the test set. The results with oversampling are the same as in Table 6.6.	53
6.10	Test Set results predicting accuracy scores. All models are trained on the training set and evaluated on the test set.	53
6.11	Test Set results predicting transparency scores. All models are trained on the training set and evaluated on the test set.	54
A.1	Bufalopedia Labels (as of May 2020) [9]	59
A.2	The Black List by Bufale (as of 2018) [8]	60
A.3	The Black List by Butac (as of May 2021) [10]	61
A.4	Partisan News Analysis by BuzzFeed News (as of August 2017) [50, 51]	62
A.5	PolitiFact’s Fake news almanac (as of November 2017) [61]	62
A.6	Columbia Journalism Review Labels (as of April 2021) [64]	63
A.7	Fake News Watch Labels (as of January 2016) [91]	63
A.8	Media Bias Fact Check Labels (as of May 2021) [15]	64
A.9	Washington Post’s Fake news list (as of 2016) [62, 98]	66
A.10	NewsGuard Credibility Criteria (as of September 2021) [53]. A numeric score of each criteria is noted in the parentheses.	68
A.11	NewsGuard Transparency Criteria (as of September 2021) [53]. A numeric score of each criteria is noted in the parentheses.	69
A.12	Rules to derive Accuracy labels from NewsGuard Credibility Criteria. The IDs of the criteria are used as propositional variables, i.e. $c_i = 1$, if criterion c_i is satisfied, and $c_i = 0$ otherwise. Have to be applied top-down.	69
A.13	Rules to derive Transparency labels from NewsGuard Transparency Criteria. The IDs of the criteria are used as propositional variables, i.e. $t_i = 1$, if criterion t_i is satisfied, and $t_i = 0$ otherwise. Have to be applied top-down.	70

Bibliography

- [1] Charu C. Aggarwal. *Neural networks and deep learning: A textbook*. Springer International Publishing, Cham, 2018.
- [2] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. In Issa Traore, Isaac Woungang, and Ahmed Awad, editors, *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pages 127–138, Cham, 2017. Springer International Publishing.
- [3] ABC News AU. Twitter bot network amplifying russian disinformation about ukraine war, researcher says. <https://www.abc.net.au/news/science/2022-03-30/ukraine-war-twitter-bot-network-amplifies-russian-disinformation/100944970>. Accessed: 2022-03-31.
- [4] Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. Predicting factuality of reporting and bias of news media sources, 2018.
- [5] Ramy Baly, Georgi Karadzhov, Jisun An, Haewoon Kwak, Yoan Dinkov, Ahmed Ali, James Glass, and Preslav Nakov. What was written vs. who read it: News media profiling using text analysis and social media context, 2020.
- [6] Alessandro Bessi, Fabiana Zollo, Michela Del Vicario, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. Trend of narratives in the age of misinformation. *PLOS ONE*, 10(8):1–16, 08 2015.
- [7] Alessandro Bondielli and Francesco Marcelloni. A survey on fake news and rumour detection techniques. *Information Sciences*, 497:38–55, 2019.
- [8] Bufale. The black list. <https://www.bufale.net/the-black-list-la-lista-nera-del-web/>. Accessed: 2021-07-23.
- [9] Bufalopedia. Un catalogo di indagini e risorse antibufala. <https://bufalopedia.blogspot.com/p/siti-creatori-di-bufale.html>. Accessed: 2021-07-23.
- [10] Butac. The black list. <https://www.butac.it/the-black-list/>. Accessed: 2021-07-23.

- [11] C. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media*, 8(1):216–225, 2014.
- [12] Pew Research Center. How americans tweet about the news. <https://www.pewresearch.org/fact-tank/2021/12/14/how-americans-tweet-about-the-news/>. Accessed: 2022-04-02.
- [13] Pew Research Center. News consumption across social media in 2021. <https://www.pewresearch.org/journalism/2021/09/20/news-consumption-across-social-media-in-2021/>. Accessed: 2022-04-02.
- [14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [15] Media Bias Fact Check. <https://mediabiasfactcheck.com/>. Accessed: 2021-07-23.
- [16] Kyunghyun Cho. Natural language understanding with distributed representation, 2015.
- [17] CrowdTangle. Crowdtangle api. <https://github.com/CrowdTangle/API>. Accessed: 2022-02-14.
- [18] CrowdTangle. Post. <https://github.com/CrowdTangle/API/wiki/Post>. Accessed: 2022-02-17.
- [19] Limeng Cui, Suhang Wang, and Dongwon Lee. Same: Sentiment-aware multi-modal embedding for detecting fake news. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '19*, pages 41–48, New York, NY, USA, 2019. Association for Computing Machinery.
- [20] Kareem Darwish, Peter Stefanov, Michaël Aupetit, and Preslav Nakov. Unsupervised user stance detection on twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, 14:141–152, 2020.
- [21] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.
- [22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

- [23] Sarah Evanega, Mark Lynas, Jordan Adams, and Karinne Smolenyak. Coronavirus misinformation: quantifying sources and themes in the covid-19 'infodemic'. *JMIR Preprints*, 19(10):2020, 2020.
- [24] Hugging Face. Hugging face transformers. <https://huggingface.co/docs/transformers/index>. Accessed: 2022-02-18.
- [25] FactCheck.org. A project of the annenberg public policy center. <https://www.factcheck.org/>. Accessed: 2021-09-13.
- [26] Fiskkit. Discuss news that matters and find out what's true. <https://fiskkit.com/>. Accessed: 2021-09-13.
- [27] Riccardo Gallotti, Francesco Valle, Nicola Castaldo, Pierluigi Sacco, and Manlio de Domenico. Assessing the risks of 'infodemics' in response to covid-19 epidemics. *Nature Human Behaviour*, 4(12):1285–1293, 2020.
- [28] David Garcia. Social data science. <https://dgarcia-eu.github.io/SocialDataScience/>. Accessed: 2022-03-21.
- [29] Jon Green, Stefan McCabe, Sarah Shugars, John Harrington, Hanyu Chwe, Luke Horgan, Shuyang Cao, and David Lazer. Curation bubbles: Domain versus url level analysis of partisan news sharing on social media. 2021.
- [30] Nir Grinberg, Kenneth Joseph, Lisa Friedland, Briony Swire-Thompson, and David Lazer. Fake news on twitter during the 2016 u.s. presidential election. *Science (New York, N. Y.)*, 363(6425):374–378, 2019.
- [31] The Guardian. Tiktok algorithm directs users to fake news about ukraine war, study says. <https://www.theguardian.com/technology/2022/mar/21/tiktok-algorithm-directs-users-to-fake-news-about-ukraine-war-study-says>. Accessed: 2022-04-02.
- [32] Nuno Guimarães, Álvaro Figueira, and Luís Torgo. An organized review of key factors for fake news detection. *arXiv preprint arXiv:2102.13433*, 2021.
- [33] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
- [34] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.
- [35] Benjamin Horne and Sibel Adali. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):759–766, May 2017.

- [36] Twitter Inc. Building queries for search tweets. <https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query>. Accessed: 2022-02-14.
- [37] Twitter Inc. Metrics. <https://developer.twitter.com/en/docs/twitter-api/metrics>. Accessed: 2022-02-16.
- [38] Twitter Inc. Twitter api. <https://developer.twitter.com/en/docs/twitter-api>. Accessed: 2022-02-14.
- [39] Twitter Inc. Twitter api v2 data dictionary – tweet. <https://developer.twitter.com/en/docs/twitter-api/data-dictionary/object-model/tweet>. Accessed: 2022-02-17.
- [40] Daniel Jurafsky and James Martin. *Speech and language processing*. 3rd edition, 2018.
- [41] Rohit Kumar Kaliyar, Anurag Goswami, and Pratik Narang. Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia Tools and Applications*, 80(8):1–24, 2021.
- [42] Srijan Kumar and Neil Shah. False information on web and social media: A survey, 2018.
- [43] Yinhan Liu, Myle Ott, Naman Goyal, Du Jingfei, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach.
- [44] Zhiyuan Liu, Yankai Lin, and Maosong Sun. *Representation Learning for Natural Language Processing*. Springer Nature, 1st ed. 2020 edition, 2020.
- [45] Sahil Loomba, Alexandre de Figueiredo, Simon J. Piatek, Kristen de Graaf, and Heidi J. Larson. Measuring the impact of covid-19 vaccine misinformation on vaccination intent in the uk and usa. *Nature Human Behaviour*, 5(3):337–348, 2021.
- [46] Meta. Reactions now available globally. <https://about.fb.com/news/2016/02/reactions-now-available-globally/>. Accessed: 2022-02-16.
- [47] Tanushree Mitra and Eric Gilbert. Credbank: A large-scale social media corpus with associated credibility annotations. *Proceedings of the International AAAI Conference on Web and Social Media*, 9(1):258–267, Aug. 2021.
- [48] Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. Stance and sentiment in tweets. *ACM Transactions on Internet Technology*, 17(3):1–23, 2017.
- [49] Pandu Nayak. Understanding searches better than ever before. <https://www.blog.google/products/search/search-language-understanding-bert/>. Accessed: 2022-02-17.

- [50] BuzzFeed News. Inside the partisan fight for your news feed. <https://www.buzzfeednews.com/article/craigsilverman/inside-the-partisan-fight-for-your-news-feed>. Accessed: 2021-07-23.
- [51] BuzzFeed News. Partisan news analysis — january 1, 2015 to march 31, 2017. <https://github.com/BuzzFeedNews/2017-08-partisan-sites-and-facebook-pages>. Accessed: 2021-07-23.
- [52] BuzzFeed News. This analysis shows how viral fake election news stories outperformed real news on facebook. <https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook>. Accessed: 2022-04-02.
- [53] NewsGuard. Newsguard ratings - newsguard. <https://www.newsguardtech.com/solutions/newsguard/>. Accessed: 2021-10-29.
- [54] Harvard NLP. The annotated transformer. <https://nlp.seas.harvard.edu/2018/04/03/attention.html>. Accessed: 2022-03-28.
- [55] Jeppe Nørregaard, Benjamin D. Horne, and Sibel Adah. Nela-gt-2018: A large multi-labelled news dataset for the study of misinformation in news articles. *Proceedings of the International AAAI Conference on Web and Social Media*, 13(01):630–638, Jul. 2019.
- [56] Nicole O’Brien, Sophia Latessa, Georgios Evangelopoulos, and Xavier Boix. *The Language of Fake News: Opening the Black-Box of Deep Learning Based Detectors*. Center for Brains, Minds and Machines (CBMM), 2018.
- [57] Ray Oshikawa, Jing Qian, and William Yang Wang. A survey on natural language processing for fake news detection, 2020.
- [58] Lucas Ou-Yang. Newspaper3k: Article scraping & curation. <https://github.com/codelucas/newspaper/>. Accessed: 2022-02-14.
- [59] Gordon Pennycook and David G. Rand. Fighting misinformation on social media using crowdsourced judgments of news source quality. *Proceedings of the National Academy of Sciences of the United States of America*, 116(7):2521–2526, 2019.
- [60] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news.
- [61] Politifact. Politifact’s guide to fake news websites and what they peddle. <https://www.politifact.com/article/2017/apr/20/politifacts-guide-fake-news-websites-and-what-they/>. Accessed: 2021-07-23.

- [62] Washington Post. My "fake news list" went viral, but made-up stories are only part of the problem. <https://www.washingtonpost.com/posteverything/wp/2016/11/18/my-fake-news-list-went-viral-but-made-up-stories-are-only-part-of-the-variant=116ae929826d1fd3>. Accessed: 2021-07-23.
- [63] NLTK Project. Natural language toolkit. <https://www.nltk.org/index.html>. Accessed: 2022-02-18.
- [64] Columbia Journalism Review. Cjr index of fake-news, clickbait, and hate sites. <http://web.archive.org/web/20210720140548/https://www.cjr.org/fake-beta>. Originally accessed 2021-07-23, but not available anymore.
- [65] Arjun Roy, Kingshuk Basak, Asif Ekbal, and Pushpak Bhattacharyya. A deep ensemble framework for fake news detection and classification.
- [66] Natali Ruchansky, Sungyong Seo, and Yan Liu. *CSI: A Hybrid Deep Model for Fake News Detection*, pages 797–806. Association for Computing Machinery, New York, NY, USA, 2017.
- [67] Bayerische Rundfunk. Verwüsteter zug: Wie gegen ukrainische flüchtlinge gehetzt wird. <https://www.br.de/nachrichten/bayern/verwuesteter-zug-wie-gegen-ukrainische-fluechtlinge-gehetzt-wird,T0zPTuU>. Accessed: 2022-04-02.
- [68] Mohsen Sayyadiharikandeh, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, and Filippo Menczer. Detection of novel social bots by ensembles of specialized classifiers. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, Oct 2020.
- [69] Ana Lucía Schmidt, Fabiana Zollo, Michela Del Vicario, Alessandro Bessi, Antonio Scala, Guido Caldarelli, H. Eugene Stanley, and Walter Quattrociocchi. Anatomy of news consumption on facebook. *Proceedings of the National Academy of Sciences*, 114(12):3035–3039, 2017.
- [70] Indira Sen, Fabian Flöck, and Claudia Wagner. On the reliability and validity of detecting approval of political actors in tweets. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1413–1426, 2020.
- [71] Kai Shu, H. Russell Bernard, and Huan Liu. *Studying Fake News via Network Analysis: Detection and Mitigation*, pages 43–65. Springer International Publishing, Cham, 2019.
- [72] Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu. Defend: Explainable fake news detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19*, pages 395–405, New York, NY, USA, 2019. Association for Computing Machinery.

- [73] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fake-newsnet: A data repository with news content, social context and spatialtemporal information for studying fake news on social media.
- [74] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36, sep 2017.
- [75] Kai Shu, Suhang Wang, Dongwon Lee, and Huan Liu. Mining disinformation and fake news: Concepts, methods, and recent advancements, 2020.
- [76] Kai Shu, Suhang Wang, and Huan Liu. Beyond news contents: The role of social context for fake news detection. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM '19*, pages 312–320, New York, NY, USA, 2019. Association for Computing Machinery.
- [77] Snopes. The definitive fact-checking site and reference source for urban legends, folklore, myths, rumors, and misinformation. <https://www.snopes.com/>. Accessed: 2021-09-13.
- [78] Russel Stuart and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3 edition, 2010.
- [79] The New York Times. Fact and mythmaking blend in ukraine's information war. <https://www.nytimes.com/2022/03/03/technology/ukraine-war-misinfo.html>. Accessed: 2022-04-02.
- [80] The New York Times. Truth is another front in putin's war. <https://www.nytimes.com/2022/03/20/world/asia/russia-putin-propaganda-media.html>. Accessed: 2022-04-02.
- [81] Alan Descoins tryo labs. Why accuracy alone is a bad measure for classification tasks, and what we can do about it. <https://tryolabs.com/blog/2013/03/25/why-accuracy-alone-bad-measure-classification-tasks-and-what-we-can-do-about>. Accessed: 2022-11-03.
- [82] Muhammad Umer, Zainab Imtiaz, Saleem Ullah, Arif Mehmood, Gyu Sang Choi, and Byung-Won On. Fake news stance detection using deep learning architecture (cnn-1stm). *IEEE Access*, 8:156695–156706, 2020.
- [83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
- [84] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H. Eugene Stanley, and Walter Quattrociocchi. The spreading of misinformation online. *Proceedings of the National Academy of Sciences*, 113(3):554–559, 2016.

- [85] Juan Orozco Villalobos. Precision vs recall. <https://www.brainstobytes.com/precision-vs-recall/>. Accessed: 2022-03-31.
- [86] Nguyen Vo and Kyumin Lee. Hierarchical multi-head attentive network for evidence-aware fake news detection, 2021.
- [87] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.
- [88] W. Patrick Walters. Comparing classification models—a practical tutorial. *Journal of Computer-Aided Molecular Design*, 2021.
- [89] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426. Association for Computational Linguistics, 2017.
- [90] Xiaozhi Wang and Zhengyan Zhang. Must-read papers on pre-trained language models (plms). <https://github.com/thunlp/PLMpapers>. Accessed: 2022-03-24.
- [91] Fake News Watch. <https://web.archive.org/web/20180213181029/http://www.fakenewswatch.com/>. Accessed: 2021-07-23.
- [92] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.
- [93] Jianhui Xie, Song Liu, Ruixin Liu, Yinghong Zhang, and Yuesheng Zhu. Sern: Stance extraction and reasoning network for fake news detection. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2520–2524, 2021.
- [94] Xinyi Zhou, Atishay Jain, Vir V. Phoha, and Reza Zafarani. Fake news early detection. *Digital Threats: Research and Practice*, 1(2):1–25, 2020.
- [95] Xinyi Zhou and Reza Zafarani. A survey of fake news. *ACM Computing Surveys*, 53(5):1–40, Oct 2020.
- [96] Jeffrey Zhu. Bing delivers its largest improvement in search experience using azure gpus. <https://azure.microsoft.com/en-us/blog/bing-delivers-its-largest-improvement-in-search-experience-using-azure/>. Accessed: 2022-02-17.

- [97] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.
- [98] Melissa Zimdars. False, misleading, clickbait-y, and/or satirical “news” sources. https://docs.google.com/document/d/10eA5-mCZLSS4MQY5QGb5ewC3VAL6pLkT53V_81ZyitM/preview. Accessed: 2021-07-23.