Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Anja Rosvold From
Ingvild Unander Netland

Fake News Detection by Weakly Supervised Learning

A Content-Based Approach

Master's thesis in Computer Science
Supervisor: Özlem Özgöbek
June 2021

NTNU
Norwegian University of
Science and Technology

Anja Rosvold From
Ingvild Unander Netland

Fake News Detection by Weakly Supervised Learning

A Content-Based Approach

Master's thesis in Computer Science
Supervisor: Özlem Özgöbek
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

An increased use of social media for reading and sharing news articles coupled with the COVID-19 pandemic has resulted in an *infodemic*, and the challenge of detecting fake news is more relevant than ever. Fake news is here defined as '*the publication of false information, either unintentional or with the intent to deceive or harm*'. Previous research has applied machine learning to automatically detect fake news articles, and promising results have been obtained. However, most research has focused on applying supervised learning that requires manually labeled training data to obtain adequate results, which is expensive to acquire. This thesis aims to efficiently assign noisy, or *weak labels*, to news articles extracted from the NELA-GT-2019 dataset to train a weakly supervised machine learning model to distinguish between fake and real news articles. The performance of two weak labeling systems based on the Snorkel and Snuba frameworks, and five machine learning models, namely Logistic Regression, XGBoost, ALBERT, XLNet and RoBERTa, are evaluated on this task in terms of accuracy and F1 score. The models are trained on the weakly labeled data in two data scenarios: one with limited labeled data and one with considerably more labeled data. A supervised equivalent is trained for each model to measure the effect of expanding the labeled training data with weakly labeled data. Of the three weak labeling systems evaluated, the Snuba system performed best and achieved an accuracy of 0.765 on a source-based test set. This result shows that a content-based approach for labeling fake news should rely on complex heuristics to create high confidence weak labels. The end models were evaluated on a manually labeled test set gathered as part of this work. For the limited labeled data scenario, RoBERTa was the best of the five weakly supervised models, with an F1 score of 0.798, outperforming the supervised approach by 1.9 F1 points. For the scenario with more labeled data, the supervised model outperformed the best weakly supervised model. These results show that a weakly supervised approach is favorable in scenarios where the availability of labeled data is limited, but may degrade the model's performance in scenarios where the labeled dataset is sufficiently large.

Sammendrag

Økt bruk av sosiale medier til lesing og deling av nyheter i kombinasjon med COVID-19-pandemien har resultert i en *infodemi*, som gjør utfordringen ved å oppdage falske nyheter mer relevant enn noen gang. Falske nyheter er her definert som *'publisering av falsk informasjon, enten utilsiktet eller med overlegg, for å bedra eller gjøre skade.*' Tidligere forskning har brukt maskinlæring for å detektere falske nyhetsartikler, noe som har gitt lovende resultater. Imidlertid fokuserer det meste av den tidligere forskningen på å bruke veiledet læring, noe som krever manuelt merket opplæringsdata for å oppnå tilstrekkelige resultater, som er ressurskrevende å samle inn. For å løse dette problemet foreslår vi en metode som tilegner svake merker til et umerket datasett ekstrahert fra NELA-GT-2019, som deretter brukes til å svakt veilede en klassifiseringsmodell. Ytelsen til tre svake merkesystemer basert på rammeverkene Snorkel og Snuba, og de fem klassifiseringsmodellene Logistisk Regresjon, XGBoost, ALBERT, XLNet og RoBERTa, ble evaluaert i forbindelse med nøyaktighet og F1 poengsum. Modellene er trent på svakt merket data i to datascenarier: ett med en begrenset mengde merket data og ett med betydelig mer merket data. En veiledet ekvivalent med kun merket data blir trent for hver modell for å måle effekten av å utvide andelen treningsdata ved å legge til den svakt merkede dataen. Av de tre evaluerte svake merkesystemene, hadde det automatiske Snuba-systemet høyest ytelse, og klassifiserte 76,5% av alle instanser i et kildebasert testsett korrekt. Dette resultatet viser at en innholds-basert tilnærming for merking av falske nyheter bør basere seg på komplekse heuristikker for å skape svake etiketter med høy nøyaktighet. Klassifiseringsmodellene ble evaluert på et manuelt merket testsett som ble samlet i denne masteroppgaven. For scenariet med begrenset mengde merket data, var RoBERTa-modellen den beste av de fem svakt veiledede modellene, med en F1-score på 0,798, noe som overgikk den veiledede tilsvarende modellen med 1,9 F1-poeng. For scenariet med mer merket data, overgikk den veiledede modellen den beste svakt veiledede modellen. Disse resultatene viser at en svakt veiledet tilnærming er gunstig i scenarier der tilgjengeligheten til merket data er begrenset, men at bruken av svakt merket data kan svekke modellens ytelse i scenarier der det merkede datasettet allerede er tilstrekkelig stort.

Acknowledgements

First, we would like to thank our supervisor Özlem Özgöbek for providing valuable feedback throughout the project, by challenging our reasonings and making us reflect upon important decisions. Second, we would like to thank Ph.D. Candidates Sina Özdemir and Hassan Abedi Firouzjaei from Trondheim Analytica, for taking the time to give their honest opinions and advice on the direction of our project. We are also grateful for the help from Stefan Helmstetter and Heiko Paulheim at the University of Mannheim for giving access to detailed documentation on their project on Weakly Supvervised Learning on Twitter data, which was a great inspiration to our work. In addition we would like to thank NTNU and the team behind the IDUN/EPIC cluster, for enabling us to run code that required extensive computational power. We also want to thank Paroma Varma from Snorkel AI for taking the time to answer our questions in relation to the Snorkel system by e-mail. Lastly, we want to thank NTNU for prioritizing the students during the COVID-19 lockdown so that we could finalize our thesis in educational environments.

Table of Contents

List of Tables

List of Figures

Abbreviations

ANN	=	Artificial Neural Network
BERT	=	Bi-Directional Encoder Representations from Transformers
CART	=	Classification and Regression Trees
CNN	=	Convolutional Neural Network
FW	=	Further Work
GM	=	Generative Model
k-NN	=	k-Nearest Neighbors
LF	=	Labeling Function
LIWC	=	Linguistic Inquiry and Word Count
LR	=	Logistic Regression
ML	=	Machine Learning
MLE	=	Maximum Likelihood Estimation
MLM	=	Masked Language Modeling
MV	=	Majority Vote
NAD	=	Normalized Absolute Difference
NLP	=	Natural Language Processing
NLTK	=	Natural Language Toolkit
NSP	=	Next Sentence Prediction
POS	=	Part-of-Speech
RDL	=	Relative Difference Limit
RNN	=	Recurrent Neural Network
RQ	=	Research Question
SVM	=	Support Vector Machine
TF-IDF	=	Term Frequency-Inverse Document Frequency
XAI	=	Explainable Artificial Intelligence
XGBoost	=	Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Background and Motivation

The spread of *fake news* has become a recognized problem over the last decade, especially following the 2016 US presidential election. However, defining the term is not straightforward. Schudson et al. (2017) identify three types of information disorders covered by the *fake news* term: *misinformation*, *disinformation* and *malinformation*. Here, misinformation is defined as an unintentional publication of false statements, while disinformation is defined as fabricated or deliberately manipulated content intending to conspire or spread rumors. Malinformation is defined as deliberately revealing private information that could potentially have been tampered with to serve a personal or corporate interest. In this work, we include the categories of both misinformation and disinformation, yielding the definition of fake news as the '*publication of false information, either unintentional or with the intent to deceive or harm*'.

Fake news in the media is not a new phenomenon, despite its growing public interest following the 2016 US presidential election (Krause et al., 2019). The term fake news has reportedly been used as early as in 1895 when it appeared in Electricity: A Popular Electrical Journal, stating that the newspaper *'never copies fake news'* (Perry et al., 1895). Nor is it a novel issue within social media; In fact, the spread of falsehoods on Twitter has frequently occurred since the platform became available in 2006 (Wendling, 2018). So why the sudden need for action? There are mainly three reasons for this: First, information has become increasingly more available. Second, fake news is being spread faster on social media now than before, and third, many people use social media as their primary news source. Each of these issues are further elaborated below.

Availability. Since the rise of the internet, the number of web pages continues to grow (Huberman and Adamic, 1999), and as of 2021, there are over 1.8 billion[1] websites online. Coupled with an ever-increasing availability of information, this is a double-edged sword that has also increased the amount of false information available. Information directly

[1] `https://www.internetlivestats.com/total-number-of-websites/`, Last Accessed: 13.06.2021

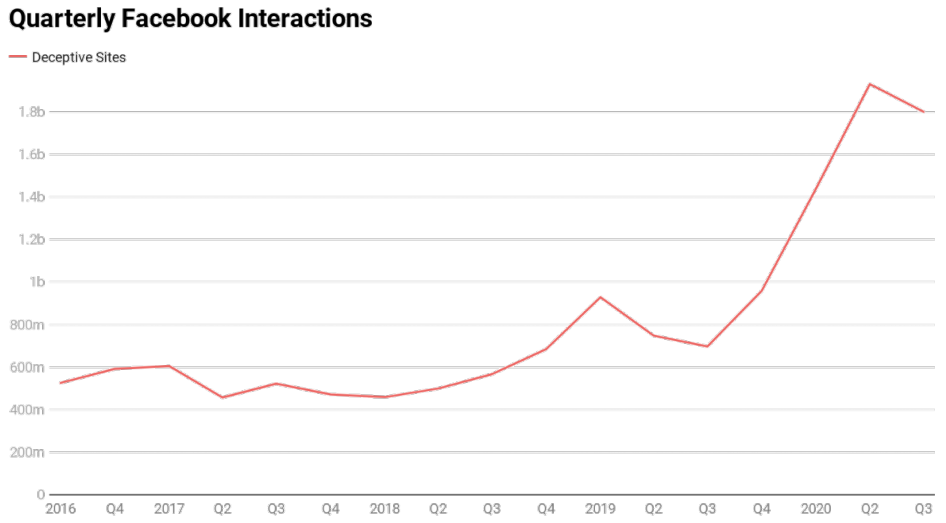

Figure 1.1: The number of interactions (likes, comments and shares) with deceptive sites through Facebook on a quarterly basis from 2016 until 2020, in steps of 200 million interactions (Kornbluh et al., 2020).

impacts our decision-making process, and can over time create a cognitive bias, which is a systematic error in our thinking (Tversky and Kahneman, 1973; Spohr, 2017). The most prevalent bias within the fake news domain is the availability bias which, according to Tversky and Kahneman (1973), *'occurs when a person evaluating the probability of a chance event makes the judgment in terms of the ease with which relevant instances come to mind'*. Humans, therefore, consider information that comes quickly to mind as more likely to be true (Morin, 2020).

Spread on social media. Falsehoods were frequently shared in the early days of Twitter, albeit at a rate that allowed the community of users to disprove them quickly. Today, an increasing number of autonomous programs are posting fabricated stories on social media at a scale that makes it hard for fact-checkers to keep up (Wendling, 2018). Leading up to the 2016 US presidential election, fake news related to the election spread rapidly on social media. As an example, pro-Trump fake news stories were shared over 30 million times on Facebook (Allcott and Gentzkow, 2017). A growing number of deceptive sites disguised as news story outlets designed to promote conspiracies are appearing. In 2020, The Digital New Deal project embarked on a mission to map out these sites and their impact (Kornbluh et al., 2020). They found that the number of interactions (likes, comments, and shares) with these deceptive media sites through Facebook had increased by 102% since the US presidential election in 2016, as shown in Figure 1.1. The spread of fake news is therefore an even bigger problem now than in 2016.

Social media as a primary news source. Research shows a rise in the number of people who use social media platforms as their news source. Reuters Institute for the Study of Journalism conducts an annual report named *Reuters Institute Digital News Report*, which analyses news consumption patterns based on data collected from 40 countries across all

continents (Newman et al., 2020). According to the 2020 report, 42% of people above 35 years old used social media as a source of news in April 2020, and for people at the age of 35 years or younger, the number was 61%. The report also showed a global concern about misinformation and fake news, where social media is perceived as the culprit behind the massive spread of fake news. Facebook is regarded as the most problematic platform in almost every participating country.

The three mentioned issues amplify each other, resulting in a toxic concoction of fake news and media mistrust which can cause irrational fear. To illustrate the magnitude of the problem, we can look at the complications that the spread of false information has introduced during the COVID-19 pandemic. Facebook's quarterly Community Standards Enforcement Report[2], established to track their efforts in policing the content shared on their platform, reported the removal of 7 million false stories regarding the COVID-19 virus and fabricated preventive measures for handling the virus in the second quarter of 2020 (Paul and Vengattil, 2020). The deteriorating quality and excessive quantity of information spreading about the COVID-19 virus has reached the point of being referred to as an *infodemic* by the Director-General of the World Health Organization, Tedros Adhanom (Diseases, 2020). An infodemic is a term used to describe *'a rapid and far-reaching spread of both accurate and inaccurate information about something, such as a disease'*[3]. An infodemic may result in widespread confusion and growing mistrust in health authorities, ultimately causing the pandemic accelerate the number of COVID-19-related deaths.

The sudden need for action is thus not caused by the novelty of fake news, but is rather a consequence of the explosive growth in the spread of false information and the complex and widespread repercussions this introduces.

1.2 Problem Outline

A step towards countering the issue of fake news is to create fake news detection systems. Within artificial intelligence, much research has focused on using machine learning to detect false news stories automatically (Pérez-Rosas et al., 2017; Reis et al., 2019; Kaliyar et al., 2020). Detecting whether an article is fake or real is considered a classification task and is commonly solved by supervised learning approaches. Multiple supervised learning algorithms are designed to handle classification, but a requirement is to have labeled training data as supervision signals. However, for many applications, there is an issue of acquiring enough labeled training data. According to Roh et al. (2021) there are mainly two reasons for this: 1) there is little to no data available due to little data being gathered, and 2) that the cost of labeling the data is expensive.

There are large amounts of news data available for fake news detection, as online news sites generate a continuous flow of articles. However, annotating labels to the data is complex and time-consuming. In order to use the data for supervised learning, the articles must first be fact-checked and annotated by domain-experts, which is a scarce resource. As

[2] `https://transparency.fb.com/data/community-standards-enforcement`, Last accessed: 14.6.2021

[3] `https://www.merriam-webster.com/words-at-play/words-were-watching-infodemic-meaning`, Last accessed: 20.02.21

a result very few labeled datasets of sufficient size and quality for supervised learning exist. This data deficiency is considered a bottleneck within the task of fake news detection.

Additionally, the content and topic of articles vary drastically and are time-dependent, as new events cause new topics to be introduced (Castelo et al., 2019). The performance of machine learning models trained on manually annotated data can therefore deteriorate over time as the content of unseen articles diverges from the content of the training data. To conquer the time-dependency issues related to news data, it is necessary to regularly re-train the end-model on new data, requiring new instances to be efficiently labeled.

A way to bypass the issues related to the data bottleneck is to apply a weak supervision approach. Weak supervision allows for efficient labeling by using noisy labels as weak supervision signals. Weak supervision systems have previously been developed that utilize both content-based and contextual features such as likes, comments, and shares of an article to generate labels for fake news data. This approach has given promising results, but contextual features are time-dependent as the number of shares and likes changes over time. As a result, contextual features take time to accumulate and are not necessarily available.

In this work, we focus on using only content-based features such as the title and content of the articles. By implementing a weak supervision approach based solely on the content, it is possible to apply weak labels to the articles in real-time and not wait for contextual features to be gathered.

1.3 Research Goal

This thesis aims to address the previously outlined challenges of fake news detection by developing weak labeling systems that efficiently label news articles based on features extracted from their content. Two frameworks, Snorkel[4] and Snuba[5], are utilized for creating the weak labeling systems. After annotation, the weak labels are used to train five weakly supervised machine learning models to distinguish between fake and real news content, namely Logistic Regression, XGBoost, ALBERT, XLNet and RoBERTa. To assess the quality of the proposed weak supervision system, a comparison of the weakly supervised models is made to their supervised equivalents. In relation to the research goal, this thesis will study the following research questions (RQs):

RQ1 *What is the best weak labeling system that uses content-based features for fake news detection?*

RQ2 *Which weakly supervised machine learning model performs best at detecting fake news?*

RQ3 *How is the performance of a machine learning model affected by expanding the training data with weakly labeled data?*

[4] `www.snorkel.org`, Last accessed: 16.6.2021

[5] `https://github.com/HazyResearch/reef`, Last accessed: 16.6.2021

1.4 Preliminary Work

The basis for this thesis is the content-based weak labeling system for fake news articles proposed in From and Netland (2020). The weak labeling system is based on previous research showing that the content and style of real and fake news articles are inherently different (Horne and Adali, 2017; Rashkin et al., 2017). The system's primary purpose is to augment the news data by creating numerical features extracted from the content and title of an article and analyze their distributions to manually find good heuristics that distinguish real and fake news articles. The heuristics' purpose is to find these differences and assign labels to unseen instances accordingly. The proposed weak labeling system was implemented using the Snorkel framework and will be referred to as the *manual weak labeling system in Snorkel*. The best result achieved by the manual weak labeling system was an accuracy of 70%, an F1-score of 0.71, and a coverage of 86%. This weak labeling system will serve as a baseline for evaluating the weak labeling systems developed in this work.

1.5 Contributions

The contribution of this work is three-fold. The first contribution is the creation of a weak labeling system that inputs unlabeled data, extracts features from the content of an article, and outputs a probabilistic weak label for each instance, indicating the probability that an article is fake. The probabilistic labels are generated by multiple heuristics created from and evaluated by a smaller dataset with ground truth labels. The generated weak labels can then be used for training a machine learning model.

The second contribution is a thorough understanding of five machine learning models' performance at detecting fake news articles. For the weak supervision approach, the models are trained on a combination of ground truth labels and weak labels, which is then compared to a supervised approach trained on only ground truth labels.

The third contribution is the collection of a balanced test set consisting of 434 news articles. The articles are manually labeled by experts from the fact-checking sites Snopes[6] and PolitiFact[7]. The dataset can be downloaded from a GitHub repository[8].

1.6 Report Outline

This thesis is organized in six parts. Chapter 2 presents the theoretical background that serves as a basis for the experiments conducted in this work. Chapter 3 summarizes related research conducted within fake news detection, weak labeling and weak supervision, and their findings. Chapter 4 explains the method used, including characteristics of the dataset used in the experiments as well as the implementation of all systems used in this work. Chapter 5 describes the experiments conducted to evaluate the weak labeling systems and the end models by clearly stating the steps taken for each experiment. Chapter 6

[6] `www.snopes.com`, Last accessed: 14.6.2021

[7] `www.politifact.com`, Last accessed: 14.6.2021

[8] `https://github.com/piiingz/fake-news-detection-test-set`, Last accessed: 14.6.2021

presents results and discussion for each experiment, and Chapter 7 concludes the results by answering the research questions formulated in this chapter and proposes improvements and experiments for further work.

Chapter 2

Background

This chapter provides an overview of the theoretical background needed as a prerequisite for the experiments in this thesis. First, the Natural Language Processing techniques applied are explained in Section 2.1, and document representation is presented in Section 2.2. Machine learning in general is presented in Section 2.3, including both supervised and weakly supervised learning. Section 2.4 presents the weak labeling systems, and the classification models used in this work are presented in Section 2.5. Lastly, the evaluation metrics are presented in Section 2.6.

2.1 Natural Language Processing

Natural Language Processing (NLP) is an essential step of allowing computer systems to interpret and derive meaning behind the human language. The purpose of NLP is to extract a meaningful representation from raw text data, based on linguistic principles like Part-of-Speech (POS) and grammatical structure (Kao and Poteet, 2006). Numerous different techniques can be applied, but not all are suitable for each use case and dataset. The following section will explain the specific NLP techniques applied in this thesis.

2.1.1 Preprocessing Techniques

Tokenization

Tokenization in NLP is the process of splitting raw text into smaller parts, called *tokens* (Kao and Poteet, 2006). A token can typically be a single word, character, or sentence. This step is helpful for further preprocessing where, for instance, single words or sentences are addressed individually. An example of word tokenization is splitting the following sentence into word tokens like so:

‘This is a sample sentence’ $\rightarrow$ [‘This’, ‘is’, ‘a’, ‘sample’, ‘sentence’]

Case Normalization

Case Normalization is an NLP technique that involves obtaining all input words in the same case variation. This technique is necessary because a computer will interpret the words 'Book' and 'book' as different words, even though the semantic meaning of the words are identical (Bird et al., 2009). This step is a basic form of NLP preprocessing and can be implemented in several ways. An option is to retain capitalized or uppercased words, like proper nouns and abbreviations. Doing so keeps distinctions like 'Apple' the company vs. 'apple' the fruit, in the text. This approach requires extensive pre-analysis, so a common approach is to skip such considerations and simply lowercase all words. Below is an example of lowercasing an input sentence:

> 'This is a sample sentence' → 'this is a sample sentence'

Remove Punctuation

A simple technique to remove noise in the data is removing punctuation. Raw text contains punctuation characters, such as commas, apostrophes, and quotes. Similar to case normalization, the computer may interpret 'book.' and 'book' as different words. On the other hand, we may want to keep contractions and hyphenated words like 'it's' and 'five-year-old.' See the following example:

> 'This is a sample sentence.' → 'This is a sample sentence'

Stop Word Removal

Stop word removal involves removing stop words from the document, which are words that frequently occur in all documents and do not contribute to an additional meaning of a text. Examples of stop words in the English language are 'the', 'a' and 'and'. Removing them could be beneficial to reduce the dimensionality of the input without losing meaning. No universal list of stop words is defined, but they are often considered the most common words of a language that will likely be present in all texts. See the following example:

> 'This is a sample sentence' → 'sample sentence'

Part-of-Speech Tagging

Part-of-Speech tagging is the process of classifying a word to its part of speech, also called word class or lexical category, based on both its definition and context (Bird et al., 2009). Each word in a context is assigned a tag representing, for instance, a noun, verb, or adjective. The tags may also include more complex textual information like the words' tense and number form (plural or singular). A simple example of POS tagging is given below.

> 'This is a sample sentence' → [('This', 'determiner'), ('is', 'verb'), ('a', 'determiner'), ('sample', 'adjective'), ('sentence', 'noun')]

Lemmatization

Lemmatization is a text normalization technique that returns the lemma of a word. A lemma is a base or canonical form of a word without inflectional endings (Bird et al., 2009), and is similar to the stemming technique that simply cuts the suffix of a word. However lemmatization also considers the Part-of-Speech tag to return the word to its correct base form depending on the word class. For example, plural nouns will be changed to singular, and verbs converted to present tense. See the simple example below:

'There are many sample sentences' → 'There be many sample sentence'

2.1.2 Sentiment Analysis

Sentiment analysis is a sub-field within NLP that computationally quantifies the subjective sentiments and emotions in natural language. Natural languages are expressive, meaning words and phrases can embody a tone of opinion that conveys an implicit, underlying goal. The formal goal of sentiment analysis of a text is to find a measure of the sentiment of a document (Dey et al., 2018).

A measure of sentiment consists of mainly two components, which is the *subjectivity* and the *polarity* of a document (Baccianella et al., 2008). Subjectivity measures whether the text is neutral or opinionated. Given subjectiveness in the text, polarity measures whether the opinions are positive or negative and the strength of the negativity and positivity.

It is possible to extract these features by applying a lexical approach. Lexical approaches utilize a lexicon of words to map sentiments to their respective polarity and subjectivity scores. Each word in the lexica has been (often-most manually) assigned a subjectivity and polarity score. The scores are retrieved and subsequently combined to find a score for the document as a whole.

2.2 Document Representation

An important issue is representing natural language text in a way that a machine learning model can interpret. The text needs to be converted into features as input to the model, and converting the text into a vector of integers or floats is a common approach. One possibility is to use *one-hot-encoding*, in which each document is represented as an array consisting of 0's and 1's. Each non-zero value in the array corresponds to a particular word in the document. For a document of 20 words in a vocabulary of 40,000 words, the resulting encoding will be a sparse 40,000-dimensional vector with at most 20 rows holding a non-zero value (Goldberg, 2017, p.89). These vectors are very high-dimensional and sparse, which can be challenging for many machine learning methods to handle.

2.2.1 Term Frequency-Inverse Document Frequency

A more clever approach than one-hot-encoding is the Term Frequency-Inverse Document Frequency (TF-IDF) method, which uses a denser representation and also considers the

words' relative frequency. TF-IDF is a statistical measure commonly used in information retrieval. As defined in Manning et al. (2008), the TF-IDF measure consists of two terms; *term frequency* and *inverse document frequency*. Term frequency is denoted $tf_{t,d}$, and measures the number of occurrences by a term t in a document d. This view of a document is called a *bag-of-words model*, a model which only takes into account the number of occurrences but ignores the order of the words. The other term, inverse document frequency, idf_t, includes the document frequency df_t which measures the proportion of documents in a collection of total size N that contains the term t. The idea is to disregard terms that often appear in all documents and thus have little discriminating power and pay more consideration to rare terms. Combining term frequency and inverse document frequency yields the composite TF-IDF score for each term by

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t, \tag{2.1}$$

where

$$idf_t = log\frac{N}{df_t}. \tag{2.2}$$

A collection of documents is represented by a TF-IDF matrix M where $M_{i,j}$ equals the TF-IDF score of term j in document i. This document representation has a limitation of not capturing similarities between words, but it has the advantage of being simple and inexpensive to implement.

2.3 Machine Learning

Machine learning (ML) is a sub-field within Artificial Intelligence with the goal of enabling computer programs to learn complex tasks. More formally, the definition of machine learning is for a program to learn from experience by improving a defined performance measure for a specific task (Zhang, 2020). Due to its versatility and improvement of performance over the years, machine learning has become a standard approach for solving a wide range of tasks, and especially classification tasks. *Classification* is considered the task of predicting the related class of a given data point, where the predictions often are referred to as targets or labels (Asiri, 2018).

Machine learning methods are commonly divided into traditional and deep learning approaches. The term 'traditional approaches' is vast and covers various algorithms but is commonly used to describe simple statistical techniques for prediction that have been around for years. Examples of such algorithms are Linear Regression, k-Nearest Neighbors, Decision Trees, and Näive Bayes. Common for them all is that they input a set of instances with several pre-defined features and find patterns and correlations in the data (Edgar and Manz, 2017). On the other hand, deep learning approaches are algorithms that mimic the workings of the human brain. Therefore, a mathematical model within the deep learning domain is called an artificial neural network (ANN). Their architecture consists of networks that input pre-defined features that can automatically extract additional features from the data. However, this results in more of a 'black-box' approach, as it is difficult to pinpoint which features contributed to the final output.

Machine learning algorithms can also be grouped by their type of learning scenario. We will cover the two learning scenarios relevant for this work: supervised learning and weakly supervised learning.

2.3.1 Supervised Learning

Supervised learning is a learning scenario in which the learner receives labeled instances that are often hand-labeled by domain experts. The name 'supervised' stems from the concept of supervising the model during training. The labels are used to fit a parameterized mathematical model that can make predictions concerning new instances (Mohri et al., 2018). The pipeline of this process is shown in Figure 2.1. During model training, data points with corresponding labels are used to tune the parameters of the model. The resulting model can then be used to predict the label of an unseen data point.

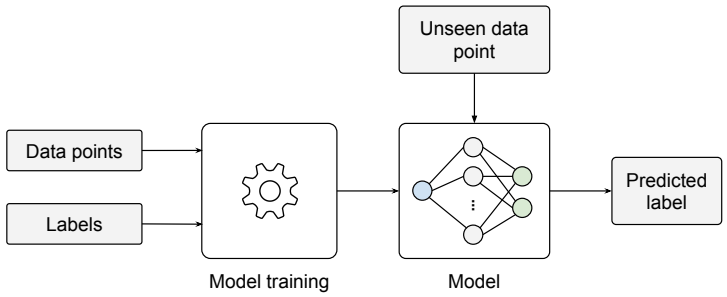

Figure 2.1: Pipeline for supervised learning. During model training, data points with corresponding labels are used to fit a parameterized mathematical model. The resulting model then predicts the label of an unseen data point. The figure is retrieved from From and Netland (2020).

2.3.2 Weakly Supervised Learning

Weak supervision is a new programming paradigm within machine learning that has risen to counteract the need for labeled data. It has the same objective as supervised learning but is trained using low-quality labels to fit the model instead of ground truth labels acquired by domain-experts (Ratner et al., 2017b). The low-quality labels, also called *weak labels*, are attained from a single weak supervision source or aggregated by multiple weak supervision sources. The weak supervision sources can be of various types, ranging from rules provided by domain experts, to cheap annotations from non-experts (known as crowdsourcing) or noisy predictions from other pre-trained models. The cost of annotating an instance by the weak supervision sources is equal regardless of the number of data points, allowing for a cheap augmentation of training data. The overall pipeline of the process for weakly supervised learning can be seen in Figure 2.2. First, unlabeled data points are combined with an ensemble of weak supervision sources to create a weak label for each data point. The data points and the weak labels are then used to fit a weakly supervise a machine learning model that subsequently can predict the label of an unseen data point.

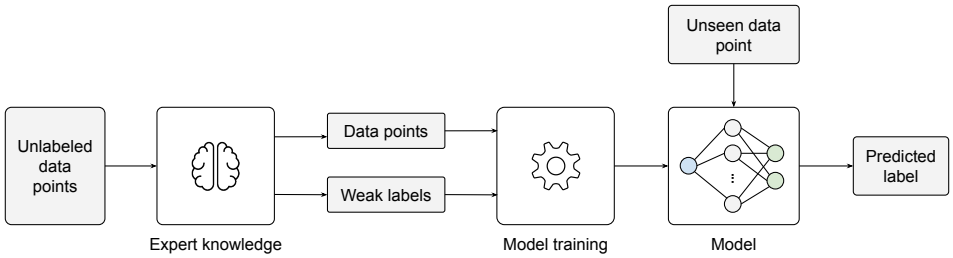

Figure 2.2: Pipeline for weakly supervised learning. Unlabeled data points are combined with an ensemble of weak supervision sources defined through expert knowledge to create a weak label for each data point. The data points and the weak labels are then used to fit a supervised learning model that predicts the label of an unseen data point. The figure is retrieved from From and Netland (2020).

2.4 Weak Labeling Systems

In order to train a weakly supervised classification model, we must first acquire the weak labels. A common approach for weakly labeling instances is to design weak supervision sources that assign labels to instances and accumulate them into a *weak labeling system*.

The weak supervision sources can be of different types, e.g. a heuristic, a constraint or an expected distribution, to name a few. More formally, according to Ratner et al. (2017b), given a set of unlabeled data of size N, $X = \{x_1, x_2, ..., x_N\}$, with corresponding ground truth labels $Y = \{y_1, y_2, ..., y_N\}$, we can define M weak supervision sources as $p_j(x_i) = y^*_{ij}$ where $i = 1, 2, ..., N$, $j = 1, 2, ..., M$ and each $p_j(X)$ will have

- a coverage set, C_j, which is the subset of X that $p_j(X)$ is able to weakly label.
- a coverage, c_j, which is the number of samples in C_j divided by the M number of samples in X.
- an accuracy, acc_j, which is the combined expected probability that $y^* = y$ for all x_i in C_j, and is assumed to be less than 1.

This section will present two types of frameworks used for simplifying the creation of weak labeling systems: the Snorkel and Snuba systems.

2.4.1 Snorkel

Snorkel[1] is a system developed by Ratner et al. (2017a) at Stanford University that provides an interface for users to simplify the creation of weak supervision sources known as *labeling functions* (LFs). LFs are heuristics such as rules-of-thumb and regular expressions, and are applied to all dataset instances. Each LF will have an unknown accuracy and correlation to other LFs. Snorkel's task is to denoise the output from the individual LFs and aggregate them into a single output without knowing their ground truth labels. The process of weakly labeling instances in Snorkel is shown in Figure 2.3.

[1] `https://www.snorkel.org/`, Last accessed: 14.05.21

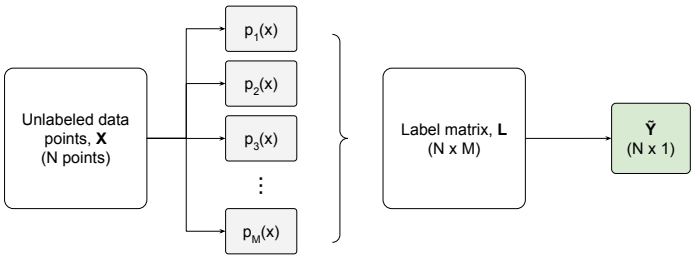

Figure 2.3: The process of assigning weak labels to unlabeled instances of data using the Snorkel framework. N unlabeled data points, X, are processed by M LFs, denoted $p_j(x_i)$, and outputs a labeling matrix of weak labels, L of size NxM. For each instance, x_i, its assigned weak labels in L are aggregated into a single weak label, and the result is an array of aggregated weak labels of size N denoted as $\tilde{Y}$. The figure is retrieved from From and Netland (2020).

Each LF inputs an instance and outputs a label within a defined set of labels. The labels can be binary or multi-class (more than two labels), but the LF can also abstain from labeling an instance. Thus an abstain label of value -1 is also included in the labeling set. If the LF abstains from labeling, it is said that the LF does not cover the instance. More formally, each training instance x_i will be assigned a weak label y_{ij}^* by each LF, $p_j(x_i)$, that has x_i in its coverage set C_j, where C_j includes all the instances that $p_j(x_i)$ did not abstain from labeling, e.g. $p_j(x_i) = -1 \rightarrow x_i \notin C_j$ (Ratner et al., 2017a). The result is a matrix of weak labels,

$$L = \begin{bmatrix} y_{11}^* & y_{12}^* & \cdots & y_{1M}^* \\ y_{21}^* & y_{22}^* & \cdots & y_{2M}^* \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1}^* & y_{N2}^* & \cdots & y_{NM}^* \end{bmatrix}, \tag{2.3}$$

which we will refer to as the *label matrix*.

The label matrix has a corresponding label density, d_L, which is the mean of the number of non-abstained labels per data point and is defined as

$$d_L = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} h(y_{ij}^*)}{N}, \tag{2.4}$$

where $h(y_{ij}^*)$ is given by

$$h(y_{ij}^*) = \begin{cases} 1, & y_{ij}^* \neq -1 \\ 0, & y_{ij}^* = -1 \end{cases}. \tag{2.5}$$

To be able to train a weakly supervised model with weak labels, the labels $y_i^* = \{y_{i1}^*, y_{i2}^*, ..., y_{iM}^*\}$ assigned to each instance x_i in L has to be aggregated into a single label, $\tilde{y}_i$. The Snorkel framework provides models for aggregating the labels through either a majority vote (MV) or by fitting a generative model (GM). An MV model outputs the most frequent label y_{ij}^* assigned to the instance x_i as the aggregated label $\tilde{y}_i$, with ties

broken according to policy. For MV, the policy is to abstain from labeling tied instances. A generative model, on the other hand, offers a more complex way to aggregate the labels. It learns the conditional probabilities each LF has of outputting the ground truth label y, namely $P(p_j(x_i)|Y)$, and utilizes the probabilities to weight and to combine all weak labels y^*_{ij} into an aggregated label, $\tilde{y}_i$. This process can be executed without the use of ground truth labels to validate the probabilities (Ratner et al., 2017a).

To select the best label aggregation method for a task, one must consider the label density, d_L of L. In low-label density settings where most data points have at most one assigned label, it has been shown that the more complex GM will not necessarily outperform an MV as there are few conflicts between labels it can learn from. In high label density settings, meaning many data points are assigned multiple labels, it is known that the MV converges to the optimal solution (Ratner et al., 2017b). Thus, a GM often excels in medium-label density settings. For a more elaborate explanation on the trade-offs between label density and model selection for aggregating labels, the reader is encouraged to read Ratner et al. (2017b).

2.4.2 Snuba

Just as for manually labeling a dataset of news articles, manually designing heuristics that can be used as weak supervision sources requires time and effort dy domain experts. To solve this issue, Varma and Ré (2018) proposed *Snuba*, a framework for automatically creating heuristics that assign probabilistic labels to instances. Snuba creates heuristics from a small labeled set of instances, U_L, given as input. The system inputs an additional, more extensive unlabeled set of instances, U_U, to which it applies the generated heuristics and outputs a probabilistic label for each unlabeled instance in U_U. The probability of an instance belonging to a class is called *label confidence* in Snuba.

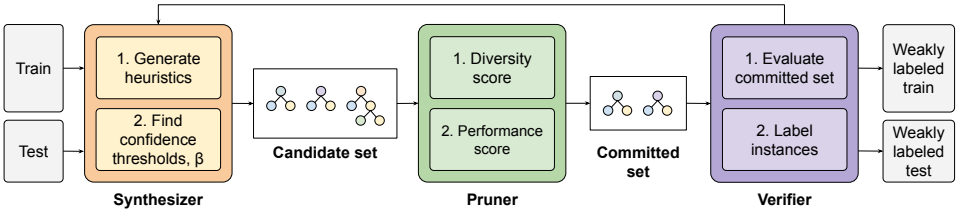

Figure 2.4: The pipeline of the automatic weak labeling system in Snuba (Varma and Ré, 2018). The system consists of three components: 1) The Synthesizer generates candidate heuristics and tunes the confidence thresholds used to decide when the heuristic will abstain from labeling. 2) The Pruner calculates each heuristic's diversity and performance score in the candidate set and adds the best heuristic to the committed set. 3) The Verifier evaluates the updated committed set and applies labels to the instances when the committed set is complete.

According to Varma and Ré (2018), Snuba consists of three main components, namely the Synthesizer, the Pruner, and the Verifier, as shown in Figure 2.4. Additionally, the system preserves a set of heuristics that will be used for the labeling of U_U after training, which is named the *committed set*. Simply put, the task of the Synthesizer is to generate candidate heuristics which are evaluated using the labels in U_L. The Pruner then selects the

best-performing heuristic of the candidates and adds it to the committed set. The Verifier then evaluates whether the performance of the committed set improved after the addition of the heuristic.

For each training iteration, the Synthesizer creates candidate heuristics from the smaller set of labeled instances, U_L. The heuristics can, in theory, be any classification model, but for this work, only the models that are pre-supported in Snuba are considered. Given a subset consisting of a user-defined number of features from U_L, the Synthesizer generates candidate heuristics which are either

- *Decision Trees*, which are small decision trees with depth limited by the number of features in the subset of U_L being evaluated. The label confidence is given by the fraction of labeled instances that belong to the same leaf as the unlabeled instance.
- *Logistic Regressors* that learn a linear decision boundary. The label confidence is found using a sigmoid function whose parameters are learned from the labeled instances.
- *k-Nearest Neighbor*, which relies on the distribution of data points in the subset for labeling instances. The label confidence is a function of the distance from the unlabeled instance to the labeled instances.

To avoid introducing noisy labels, Snuba allows the heuristics to abstain from labeling an instance if it has low confidence, resulting in a smaller labeled dataset but with high-confidence labels. A threshold β is found for each heuristic defining whether a heuristic should abstain from labeling an instance, such that

$$y_{i,j}^* = \begin{cases} 1, & P[y_{i,j}^* = 1] \geq 0.5 + \beta \\ 0, & |P[y_{i,j}^* = 1] - 0.5| < 0.5 \\ -1, & P[y_{i,j}^* = 1] \leq 0.5 - \beta \end{cases} \tag{2.6}$$

Note that the abstain value for Snuba is 0, which is not the case for Snorkel where it is set to -1.

Snuba only keeps the highest-ranking heuristic from each iteration, so the Pruner's task is to select the best heuristic from the candidate set and add it to the committed set. When selecting from the candidate heuristics, the Pruner must consider the trade-off between which instances each heuristic covers and its performance to avoid selecting heuristics that cover all instances but produces extremely noisy labels. An ideal setting is for the committed set to consist of highly accurate heuristics that each cover a small subset of the data, which in conjunction covers all or most of the data while still achieving high performance. The heuristics are evaluated on U_L in terms of diversity and F1 score to enable the selection of the candidates. The heuristic diversity is measured by the Jaccard distance between the instances labeled by a candidate heuristic and the set of instances labeled by the committed set. By weighting the F1 score of the candidate with the Jaccard distance using a simple average, the Pruner can select the best candidate while maintaining both diversity and performance.

Lastly, an automatic approach has to have a terminating condition of when to stop the generation of new heuristics to not introduce low-quality heuristics. The task of the

Verifier is thus to ensure that no heuristic is kept that will degrade the overall performance of the other heuristics in the committed set. Snuba does this by introducing a termination condition that is checked for each iteration. If the overall performance is worsened by adding another heuristic, Snuba omits this heuristic and terminates the process.

2.5 Classification Models

The goal of classifying news articles based on the features generated for the data is to create a classification model that can distinguish between the classes, namely fake and real. This section introduces the classification models chosen in this thesis for predicting the class of a news article.

2.5.1 Logistic Regression

Logistic Regression (LR) is a popular classification model similar to linear regression, except that the output label, or dependent variable, has to be categorical. Note that the input data, or independent variables, can still be high-dimensional and continuous. According to Kleinbaum and Klein (2010), the goal of the model is to use the independent variables $\mathbf{X}$ to predict the dependent variable Y. For a binary regression problem like the one in this work, the conditional probability of Y belonging to each class can be given by the independent variables $\mathbf{X}$, namely $P(Y = 1|\mathbf{X})$ and $P(Y = 0|\mathbf{X})$.

The goal of the model is to predict the occurrence of an event, e.g. an article belonging to the *fake* class ($Y = 1$), by fitting the training data to a logistic curve. The basis assumption is that the probability $P(Y|\mathbf{X})$ can be approximated as a sigmoid function σ applied to a linear combination z of the input features the following way:

$$P(Y = 1|\mathbf{X}) = \sigma(z) \tag{2.7}$$

where

$$\sigma(z) = \frac{1}{1 + e^{-z}} \; and \; z = \alpha + \sum_{i=1}^{m} \beta_i x_i. \tag{2.8}$$

The variables α and β in Equation 2.8 represent unknown parameters to be estimated, while i corresponds to the index of a specific variable, and m equals the total number of features in the data. Solving for the unknown parameters α and β has to be done numerically and is thus estimated with maximum likelihood estimation (MLE) to find values that maximize the probability. How well the model performs will depend heavily on the values chosen for the unknown parameters.

A regularized logistic regression model is employed in this work, which is a method to avoid overfitting by reducing variance in the model. The details of the regularization techniques applied can be found in the documentation of the implemented model[2].

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`, Last accessed: 14.06.2021

2.5.2 XGBoost

XGBoost, which is an abbreviation for *Extreme Gradient Boosting*, is a state-of-the-art algorithm within machine learning that has gained popularity in recent years for its efficiency and scalability to a wide range of tasks (Chen and Guestrin, 2016). As the name suggests, it is an implementation of a gradient boosted tree, which is an ensemble of *classification and regression trees* (CART) (Chen and Guestrin, 2016). A CART is a tree model where each node has a threshold for splitting instances based on its value for the feature. A condition is checked to be above or below a threshold in each node by traversing the tree until a leaf node is reached. Each leaf node holds a value that corresponds to the prediction. In boosted trees, each tree is built sequentially where subsequent trees aim to correct the error of the previous tree, ultimately making the overall object of the method to minimize the error between the prediction and the target (Friedman, 2002). Gradient boosted trees use gradient descent to minimize the error, which is fast, and the method is, therefore, able to handle large datasets even with limited computing power. The nature of tree-based models also makes them easy to interpret, which is advantageous when analyzing which features are of most importance for prediction.

2.5.3 BERT

BERT, which stands for *Bi-Directional Encoder Representations from Transformers*, is a language model developed by Devlin et al. (2018) at Google AI Language. It is a state-of-the-art language model which has in many ways introduced a new era of NLP research. What separates BERT from earlier language models is the use of bi-directional transformers, where text sequences had traditionally been analyzed sequentially word by word (Devlin et al., 2018). The bi-directional approach allows the model to process the entire text at once, analyzing a word based on its surroundings on both sides simultaneously, thus obtaining a deeper understanding of the context and flow of the text. The application of BERT extends not only to text classification but includes question answering, named entity recognition, language inference, word prediction, and more.

The BERT model uses a *transfer learning* approach by first pre-training in an unsupervised manner on a large corpus, producing a ready-to-use base model, which can later be fine-tuned on the specific problem domain task. When using BERT for a classification task, a classification layer can be added to the pre-trained base model, which is fine-tuned on the training data. The BERT base model is available as open-source and pre-trained versions, enabling easy access for anyone who wants to develop an NLP model.

Using BERT has many advantages, for example, the rapid fine-tuning enabled by the pre-trained models. In addition, the input data needs less preprocessing compared to other methods. There is, for example, no need for lowercasing or lemmatizing the text. A disadvantage of the model is that the predictions are not explainable like they are in the XGBoost model.

The Transformer

An essential building block of the BERT model is the encoder module from the *transformer*, which is another Google invention presented in the Vaswani et al. (2017). The

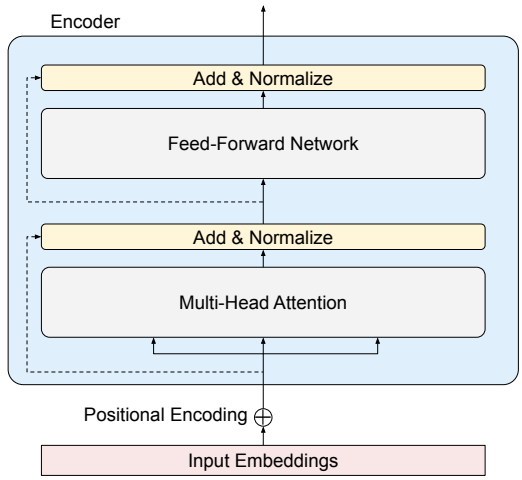

Figure 2.5: The Transformer Encoder. The encoder consists of a multi-head attention module and a feed-forward network, both followed by a normalizing layer. Input is passed on to the next encoder or decoder block. The figure is based on the work by Vaswani et al. (2017)

transformer model is a counterpoint to using recurrent neural networks (RNNs), which had until 2017 been one of the most popular methods for solving NLP tasks. Transformers have revolutionized the NLP field by instead focusing on something called *attention*. Avoiding the use of RNNs enables parallelization in the model, which as a result, increases the training speed. The initial usage for the transformer was to do machine translation, but it was soon discovered that the model could be modified to handle more NLP tasks.

The original transformer consists of components called encoders and decoders. Simply put, the encoders read and process the text input while the decoders decode the representation received from the encoders. The focus here will be on the encoder component, as this is the module that BERT uses. The transformer architecture contains multiple stacked encoders, each feeding their output to the next encoder. The structure of a single encoder is shown in Figure 2.5, a figure adapted from Vaswani et al. (2017). Each encoder consists of an attention module and a feed-forward neural network, both followed by a normalizing layer. The first encoder produces word embeddings from the input data combined with positional encodings, a way of inserting information about the word's position in the sequence. The next encoder then applies attention and propagates the neural network before passing the output onward to the next encoder, and so on.

Attention is a concept that allows the model to understand a word in the context of the surrounding words. The transformer uses attention in a way that is called *self-attention*, which incorporates the understanding of the relevant surrounding words into the embedding of the words itself. Have a look at the following examples:

- 'Server, can I have the check?'
- 'Looks like I just crashed the server'

The word ‘server’ has two different semantical meanings in these sentences, and without self-attention or other contextualized word-embeddings, they could be interpreted as having the same meaning. Self-attention allows the model to disambiguate words, do Part-of-Speech tagging, entity resolution, and more. Where the attention is put, for example, at the word ‘check’ in the first sentence, is learned from the training data.

The first step of calculating self-attention is to create three matrices by combining the current input token embedding with three pre-trained model weight matrices: a query, key, and value matrix. Next, attention scores are calculated by the scaled dot product for every other token in the sequence in relation to the query word matrix. These attention scores are passed on through a softmax function to decide how much each of the surrounding words should impact the current query word. The scores are summed and finally represent the attention of the current query token passed through to the feed-forward network. Multi-head attention, which is used in the transformer’s encoders, is an improvement of the self-attention that uses eight randomly initialized attention heads combined as the output.

The BERT Architecture

As previously mentioned, BERT uses the encoder from the transformer model. However, it has more encoder layers, larger feed-forward networks, and more attention heads than the original transformer. BERT also added two additional pre-training mechanisms called *Masked Language Modeling* (MLM) and *Next Sentence Prediction* (NSP). Both mechanisms are run when pre-training the model to minimize the combined loss functions for both strategies. The masked language model randomly masks 15% of the input words, and the model’s task is to predict the missing words based on their surrounding words. Next sentence prediction (NSP) is concerned with understanding and predicting whether two sentences are associated, i.e., if the second sentence follows the first sentence, or is unrelated.

When using BERT as a classification model, as done in this work, the classification layer is added on top of the encoder modules. The model can then be fine-tuned on the training data to produce classifications. The architecture for BERT as a classification model is shown in Figure 2.6. The model consists of the stacked encoders with input embeddings as input and a classification model on top. The classifier can, for instance, be a simple feed-forward neural network with a softmax activation function.

The input embedding is a combination of the current token embedding, segment embedding, and position embeddings. The segment embedding denotes which segment, e.g., which sentence the token is a part of. BERT can input a maximum of 512 tokens where the first token is a classification token ([CLS]) and each sentence is separated by a separation token ([SEP]). The output of the final hidden state at this position is used as the representation for the classification task, while the output from the rest of the positions will not be used directly for the classification task. However, information from all positions is incorporated into the output at the first position. A detailed explanation of how the input is combined with token, segment, and position embeddings and the flow of data through the model can be found in Devlin et al. (2018).

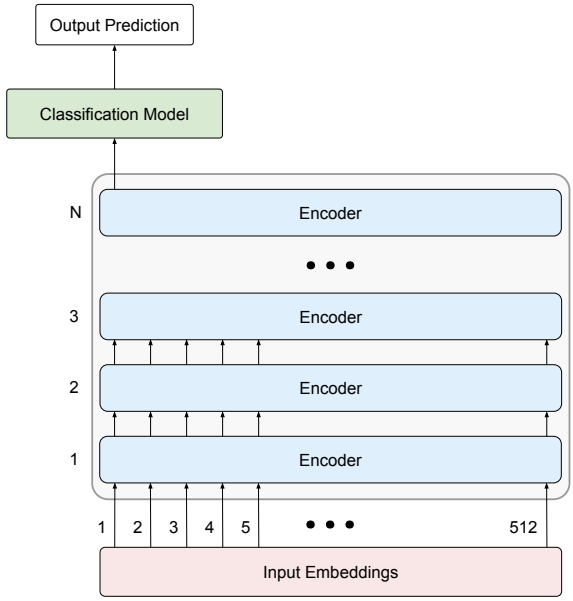

Figure 2.6: The BERT Classifier Architecture. The architecture consists of multiple stacked encoders with input embeddings as input. The final output at the first position serves as input for the classification model. The figure is based on the work by Devlin et al. (2018)

BERT-Based Models

Multiple BERT-based models or so-called BERT flavors have been developed, introducing advantages and modifications to the original architecture. The specific models used in this thesis are:

- *ALBERT (A Lite BERT)*: A light-weight BERT configuration developed by Lan et al. (2020). It presents parameter-reduction techniques to lower memory consumption and increase training speed, resulting in a model with fewer parameters and better scaling.
- *XLNet*: Proposed by Yang et al. (2020), XLNet is a generalized autoregressive pre-training method that overcomes some limitations of BERT. XLNet, for example, does not mask the input, as this neglects dependency between the masked positions. It also incorporates some mechanisms from the state-of-the-art autoregressive model TransformerXL, including capturing long-term dependencies by considering multiple sequences in relation.
- *RoBERTa (A Robustly Optimized BERT Pretraining Approach)*: A model developed by researchers at Facebook AI and the University of Washington (Liu et al., 2019). RoBERTa modifies some elements from the original BERT, e.g., removing next-sequence pre-training, using a larger dataset for pre-training and training over more iterations.

2.5.4 Hyperparameter Tuning

Hyperparameter tuning is the process of selecting the optimal hyperparameters for a machine learning model. In machine learning, hyperparameters control the learning process and are typically set by the user (Albon, 2018, p. 209). In contrast, parameters are values learned by the model during training, such as node weights. Tuning all the possible hyperparameters of a model can be a time-extensive task, thus a common practice is to only tune the most important hyperparameters. The best hyperparameters are typically chosen by training the classifier with different values for the hyperparameters repeatedly. The trained models are evaluated by either cross-validation or by using a separate validation set to prevent the model from overfitting on the training set. The best-performing model indicates the best values for the hyperparameters.

Different approaches to hyperparameter tuning exist. A manual trial-and-error approach is quite common, but often-most does not achieve optimal results. A more clever approach automates the process by performing a grid search of potential values for the chosen hyperparameters, providing an exhaustive search. Doing so can be computationally expensive as the number of combinations grows exponentially. However, using optimization methods such as Bayesian Optimization can avoid this problem, which limits the search to values that are more likely to improve the system by keeping track of the past evaluation results to form a probabilistic mapping from hyperparameter to evaluation score (Koehrsen, 2018).

2.6 Evaluation Metrics

There are three metrics used to measure how well a model assigns the correct label to instances compared to their ground truth labels in this work. This section introduces the metrics, namely accuracy, F1 score, and coverage.

In order to define these metrics, we must first look at the possible outcomes for a prediction task. In a binary classification problem an instance can either be *positive* or *negative*. The task proposed in this work is to detect fake articles, so a positive instance means it is fake, and a negative instance means it is real. When comparing an instance's predicted label to its actual label, the prediction can either be

- *true positive* (T_P): the model predicted an actual fake article as fake,
- *true negative* (T_N): the model predicted an actual real article as real,
- *false positive* (F_P): the model predicted an actual real article as fake,
- or *false negative* (F_N): the model predicted an actual fake article as real.

Now, why do we distinguish between correctly and incorrectly labeled instances based on their label? Why not combine the correctly classified instances, T_P and T_N, as well as the incorrectly labeled ones, F_P and F_N? The reason for this is that the consequences of incorrectly classifying a true instance as false may be more severe than incorrectly classifying a false instance as true. The classic textbook example for illustrating this concern is the task of detecting whether a patient has cancer; failing to detect a patient with cancer is

far worse than falsely detecting cancer in a healthy patient. While both cases introduce implications, in the former case, cancer might remain undetected and ultimately cause death. Other detection methods can be applied to validate that the patient indeed has cancer in the latter case. For fake news detection, the implication of misclassifying depends on the use case of the fake news detection system. If the system is used to pre-detect the credibility of an article to guide journalists in their fact-checking process, the consequences of misclassification are considered not to have severe implications for both cases. If the system is used directly to ensure the credibility of an article, misclassifying a fake article as real can cause unwanted bias in the readers. Misclassifying a real article as fake, on the other hand, may hurt the credibility of the author or publisher. Whichever is worse is hard to say as both cases create mistrust in the fake news detection system.

2.6.1 Accuracy

The *accuracy* of a model is the fraction of all instances that received their correct label during prediction (Chicco and Jurman, 2020) and is defined as

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}. \tag{2.9}$$

This metric is suitable for tasks where the class distribution is balanced, and the consequence of obtaining false positives and false negatives are equal, which is the case for generating weakly labeled training data in this work.

2.6.2 F1 score

When measuring a model's ability to correctly classify positive instances, in our case, the fake instances, one typically considers the *F1 score*. The F1 score is calculated by combining the *precision* and *recall* of a system. These metrics were originally defined in the field of Information Retrieval to measure a system's ability to retrieve relevant documents correctly and have later been adopted in machine learning. According to Mohri et al. (2018) precision is defined as the fraction of correctly labeled instances that are positive,

$$Precision = \frac{T_P}{T_P + T_N}, \tag{2.10}$$

and recall is defined as the fraction of all positive instances that were correctly labeled as positive,

$$Recall = \frac{T_P}{T_P + F_N}. \tag{2.11}$$

A well-performing system will achieve both high precision and recall.

F1 score is the harmonic mean of the precision and recall (Chicco and Jurman, 2020), and is defined as

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{2.12}$$

When the class distribution is imbalanced, like in most real-world problems, and the cost of having false negatives or false positives is high, F1 score is a better metric for evaluating a model.

2.6.3 Coverage

As previously mentioned, a weak labeling system is suspect to not being able to label all instances in an unlabeled dataset. To measure the ratio of covered instances, the *coverage* of a weak labeling system is the fraction of instances that have been assigned a label of all instances,

$$Coverage = \frac{N_l}{N}, \tag{2.13}$$

where N_l denotes all weakly labeled instances and N denotes all instances.

Chapter 3

Related Work

This section will present research related to fake news detection, including the characteristics of the news itself, the current state of fake news detection, supervised and weakly supervised approaches related to our approach, and present an overview of available fake news datasets.

3.1 Characteristics of Fake News

Multiple studies have shown that stylistic differences can separate fake and real news articles. In 2017, Horne and Adali conducted a study that compared known fake news articles to known real articles and found distinct differences within the content of the articles. The study showed that while real news content tends to employ factual arguments to persuade the reader, fake news content relies more on cognitive heuristics to gain the reader's attention, thus sharing more similarities with satire than real news. The study showed that the title structure is an important characteristic for separating fake and real articles, and that fake news tends to include as much information as possible in the title and often contains proper nouns to create more sensation. Consequently, the related fake news content often does not convey more knowledge nor present a nuanced debate. Other findings from this study include that real news tends to be longer and of higher linguistic complexity. Conversely, fake news tends to use fewer technical words, shorter words, and more lexical redundancy.

In a similar study, Rashkin et al. (2017) investigated the stylistic cues that can be used to assess the veracity of a text. Through experiments, they showed that fake news tends to include more second-person pronouns, first-person singular pronouns, adverbs, powerful subjectives and superlatives, as well as more *swear*, *sexual* and *negation* type words, as measured by the Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001). In contrast, real news seemed to use more comparative and assertive words, in addition to more *number*, *hear* and *money* type words, also measured by the LIWC framework. The study indicates that fake news is more likely to exaggerate and use personal language. In contrast, real news is more precise and more likely to use comparisons to present more

nuanced claims.

3.2 Current state of Fake News Detection

Since fake news became a global topic of interest, the most common approach for detecting false stories is manual fact-checking. Several fact-checking sites have emerged of which Politifact.com[1] and Snopes[2] are some of the most well-known internationally, and Faktisk[3] in Norway. Journalism experts govern these sites to debunk widespread misinformation, and their labels are considered the 'ground truth.' Other websites provide tools for a large population of non-experts to fact-check articles, known as crowd-sourced manual fact checking. This approach provides noisier labels but can give an indication of the articles' credibility if the population of fact-checkers is of a sufficient size (Zhou and Zafarani, 2020).

Another common approach is to focus on detecting fake news spreaders rather than the fake news content itself. The definition of a spreader includes ordinary people, deceptive sites, and social bots which are autonomous programs interacting with other users. Shao et al. (2017) investigated the spread of fake news by social bots in relation to the 2016 U.S. election. They analyzed 14 million messages spread on Twitter and found that social bots played an essential role in spreading low-credibility content related to the election. With this in mind, detecting social bots might be an important step in mitigating fake news spread online. Morstatter et al. (2016) studied how these bots can be detected by proposing an approach that focuses on recall to increase the number of bots that are detected. The proposed system includes the BoostOR model, a basic boosted learner that optimizes its F1 score and achieved state-of-the-art results on the bot-detection task.

Shrestha et al. (2020) detected fake news spreaders in social networks by analyzing a combination of features, including the writing style of the author, sentiment analysis, and use of psycho-linguistic (LIWC) features. They found that detecting fake news spreaders is challenging because users sharing false information are often ordinary people and are hard to differentiate from regular users who never share fake news content. Thus they did not manage to achieve high accuracy on their test data. Shu et al. (2018) further investigated which user profiles tend to spread more fake news online and found that fake news spreaders are more likely to be humans than bots. They also found that human fake news spreaders are more likely to be older people than younger and more likely to be female than male.

To detect deceptive sites spreading fake news, Castelo et al. (2019) proposed a topic-agnostic classification method that uses web-markup, in addition to LIWC and stylistic features to identify fake news pages. A motivation behind their work was that the content of fake news constantly changes with new topics emerging. They instead focus on identifying the web pages spreading the fake news independent of the topic. Their approach proved to be successful at detecting fake news even over time without frequent re-training.

A similar approach to detect fake news by spreaders is to analyze the social networks and propagation of news articles on social media to detect fake news. Monti et al. (2019)

[1] `https://www.politifact.com`, Last accessed: 09.06.2021
[2] `https://www.snopes.com`, Last accessed: 09.06.2021
[3] `https://www.faktisk.no`, Last accessed: 09.06.2021

investigated such an approach in their work based on deep geometric learning for detecting fake news on Twitter. Their results showed that analyzing social network structure and propagation plays a vital role in detecting fake news on Twitter. They argue that propagation-based approaches have advantages over content-based counterparts because knowledge of the political and social context in the texts is difficult to learn for machine learning models. The method can detect stories within a few hours of propagation. They point out that the propagation-based method is a suitable addition to the content-based approaches.

A fourth approach is using knowledge graphs and networks to simulate how professional fact-checking journalists classify articles. Ciampaglia et al. (2015) showed that the use of knowledge graphs can approximate human fact-checking quite closely. Their approach is related to fact-checking the veracity of specific claims rather than the article as a whole. As an article can include multiple claims, both true and false claims might be present in the same article. In those cases, this approach can be beneficial.

Shi and Weninger (2016) also investigated the use of link-prediction in knowledge graphs for fact-checking, using a million node knowledge graph extracted from Wikipedia and PubMedDB. They found that the predictions are easily interpretable, which is advantageous regarding the model's explainability.

In this thesis, the focus is to exploit the content and title of the article to detect fake news. However, hybrid approaches that combine the different methods mentioned might be necessary to combat the fake news issue from multiple angles. We will further review supervised and weakly supervised approaches related to our chosen method for this task.

3.2.1 Supervised Approaches

Traditional supervised machine learning techniques have proven successful for handling various tasks and have therefore gained interest within the fake news detection field. Pérez-Rosas et al. (2017) extracted linguistic features such as n-grams, punctuation, psycho-linguistic features, readability, and syntax of news article content from datasets collected from seven renowned news outlets. Two datasets were collected, consisting of 480 and 500 instances, which were used to train a supervised Support Vector Machine (SVM) as a classifier to detect fake news stories. The proposed method achieved an accuracy of 76%, which they claimed is a promising result comparable to humans' performance at this task. Based on the result, the authors proposed that future efforts should include meta-features beyond linguistic ones, such as contextual and source-based features. However, we argue that a data basis of solely 980 articles is likely insufficient to capture the full extent of the existing linguistic features. Therefore, a content-based approach should be trained on a much larger and diverse dataset to realize its full potential, which again points us to the challenge of gathering enough training data for the task.

A similar approach was investigated by Reis et al. (2019), which extracted multiple features from fake news data and evaluated them on several classifiers, including k-Nearest Neighbors (KNN), Support Vector Machines (SVM), and XGBoost. The extracted features included various features categorized as either content-based, news source-based or network structure-based. The content-based features included syntax features, Part-of-Speech features, lexical features, psycho-linguistic and sentiment features. The news source-based features included the political bias, credibility, and the domain location of

the news source. The network structure-based features included engagement in the form of likes and comments and temporal patterns. In total, this resulted in 141 textual features. Through testing these features on different classifiers, they discovered the highest performance was achieved by the XGBoost classifier, which achieved an F1 score of 0.81.

As recently as 2020, Kaliyar et al. proposed a deep Convolutional Neural Network (CNN) named FNDNet that automatically extracts discriminatory features from the raw text of news articles to classify them. The model was trained and tested using a datasets from Kaggle which consisted of 20 800 instances, where the original features included in the data were the *id*, *title*, *author*, *text* and *label* of the news articles. FNDNet achieved a state-of-the-art result of 98.36% accuracy on test data, which is beyond what most fake news detection systems has achieved. However, some questions arise about the quality of the data used to achieve this result, as Kaliyar et al. does not offer a definitive source to the dataset. However, through the description of the data, the most probable source is a dataset named Fake News[4] which can be downloaded directly from the Kaggle API. The author of the dataset stated the data was collected by merging other datasets available on Kaggle, whose sources are unknown. He further admitted he could not vouch for its quality as the intended use of the dataset was an educational workshop[5]. In the Fake and real news dataset[6] published on Kaggle, most of the true articles include the keyword 'Reuters', creating a correlation between the keyword and the label. Additionally, the dataset contains duplicates and has multiple other weaknesses. Many users report an accuracy of around 99% on this dataset by exploiting these weaknesses. If the dataset used in FNDNet includes the Fake and real news dataset, it could be that the research achieved artificially good results. This argument exemplifies the importance of using high-quality data in the development of data-driven models.

Another supervised approach is presented by Shrestha (2018), which is a hybrid approach focusing on using a combination of sentiment analysis and network metadata to detect fake news. The proposed method uses Facebook's analytical metadata in conjunction with features generated from sentiment analysis, like the polarity and subjectivity of the text. The system was implemented using a random forest classifier, which showed promising results, achieving an F1 score above 88%.

3.2.2 Weakly Supervised Approaches

The use of weak supervision signals has become a popular substitute for training labels, which can be produced through techniques like distant supervision, crowd-sourcing, and expert-designed heuristics (Varma and Ré, 2018). As described in Section 2.4.1 Snorkel is a framework that simplifies the generation of heuristics and enables more efficient heuristic creation. According to a user study performed by Ratner et al. (2017b), the developers behind Snorkel, using the framework lets subject matter experts build models 2.8 times faster and increase the predictive performance with an average of 45.5% versus seven hours of hand labeling.

[4] `https://www.kaggle.com/c/fake-news/data`, Last accessed: 01.06.2021

[5] `https://www.kaggle.com/c/fake-news/discussion/71777`, Last accessed: 01.06.2021

[6] `https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset`, Last accessed: 01.06.2021

Another framework for generating heuristics is the Snuba framework (Varma and Ré, 2018) described in Section 2.4.2. When considering the performance of an end-model trained on the probabilistic labels generated by Snuba, this approach outperformed a model trained with labels generated by user-defined heuristics by 9.74 F1 points and crowd-sourced labels by 13.80 F1 points. The work in this thesis is greatly inspired by the philosophies behind the Snorkel and Snuba frameworks, namely to exploit a limited amount of labeled data to obtain larger amounts of weakly labeled data for training an end model. This section will present the work done using weakly supervised approaches within the fake news detection domain and other textual tasks.

Weak Supervision with Contextual Information

Weak supervision for fake news classification is typically used in combination with contextual features included in the dataset. *WeFEND: Weak Supervision for Fake News Detection via Reinforcement Learning* by Wang et al. (2019) and *Weakly Supervised Learning for Fake News Detection on Twitter* by Helmstetter and Paulheim (2018) are examples of weakly supervised fake news detection systems utilizing contextual features.

WeFEND is a weakly supervised fake news detection system proposed by Wang et al. (2019) that uses crowd-sourcing for weak labeling while incorporating reinforcement learning. The data used to develop the system were news articles published by WeChat[7] official accounts, which were weakly labeled based on users' reports of the articles via WeChat, a popular mobile messaging application. Unfortunately, the data used to develop the WeFEND system was not publicly available.

The system consisted of three components, 1) the annotator which assigned the weak labels based on users' reports, 2) a reinforced selector which chose high-quality samples from the data to avoid low-quality labels degrading the prediction performance, and 3) the weakly supervised fake news detector trained on the content of instances with high-quality weak labels. The WeFEND system achieved an accuracy of 82.4%, which is a great result compared to several supervised and weakly supervised methods. The proposed system thus became an inspiration for this work, and especially the reinforced selector inspired us to select high-quality labels for the weakly labeled training data to reduce noise.

Helmstetter and Paulheim (2017; 2018) studied the use of weak supervision for classifying fake news within Twitter data. A large dataset of tweets collected through Twitter's API was annotated with noisy source-based labels by classifying users as trustworthy or untrustworthy and assigning labels to tweets based on their author's credibility. The data was then comprehensively handled by text preprocessing techniques to extract textual features from the tweets themselves. The textual features included sentiment analysis and modeling of the news topics in the data. Additionally, the social context of the tweets such as shares, comments, retweets, and user-specific features were gathered, resulting in two types of features: textual and contextual. When making predictions based solely on the textual features of the tweet, the best result was achieved by the XGBoost model. XGBoost's performance resulted in an F1 score of 0.7758 when evaluated on a test set composed of tweets labeled by the user's credibility (user-based labels) and an F1 score of 0.7426 on a test set composed of hand-labeled tweets. When the contextual features

[7] `https://www.wechat.com`, Last accessed: 01.06.2021

were added to the textual features, the XGBoost model was again the best classifier on the test set with the user-based labels with an F1 score of 0.9256. In contrast, for the hand-labeled test set, the Neural Network model achieved the best result with an F1 score of 0.8996. The classifier benefited greatly from including the contextual features, indicating that content-based approaches are insufficient for fake news detection. However, as the work was based on Twitter data, the documents are short, meaning they include less linguistic style information than traditional news articles. Additionally, the tweet's social context and the tweet's author will always be available and in a standardized format, meaning the same features will exist for all tweets. In contrast, contextual features related to news articles can be diverse and non-existent in many cases, which poses a challenge for gathering enough data with the required context.

Weak Supervision without Contextual Information

A content-based approach to weakly supervised learning without contextual information would require a heuristics-based system to provide the weak labels. Unfortunately, no such methods developed in combination with the content of fake news articles was found. However, the approach has been applied to other text classification domains, including learning discourse structure and detection of anti-Semitic tweets.

Badene et al. (2019) uses weak supervision for learning discourse structures in dialogues by comparing a Snorkel-based approach to deep learning approaches such as BERT on this task. Discourse analysis for texts involves extracting causal, topical, and argumentative information from a text and is considered a difficult task within natural language processing (NLP). They show that their approach using domain knowledge provided by experts and Snorkel's generative model surpasses using BERT in a supervised manner on this task. Their generative Snorkel model achieved an accuracy of 93%, while the best supervised BERT model achieved 89% accuracy. The task of learning discourse structure is substantially different from predicting fake news, as it has more to do with the structure of the text than the credibility of its content. The use of domain experts on the fake news detection task was not available for this thesis, so using this approach for fake news detection was not carried out. The research by Badene et al. used a large annotated corpus that simulated a weak supervision scenario by only using a portion of this as labeled data, which is similar to what will be done in this thesis. Additionally, it was interesting to observe that Snorkel surpassed BERT, so our work will also compare Snorkel and BERT-based methods for the fake news detection task.

Starosta (2019) studies the combination of transfer learning and weak labeling to build a text classifier to detect anti-Semitic tweets cheaply, and compares this to a supervised approach. Weak labeling is applied using Snorkel, and transfer learning is done by using the ULMFiT language model (Howard and Ruder, 2018), which is similar to BERT. ULMFiT was pre-trained on a large tweet dataset in order to fine-tune the model to the Twitter domain. In addition to the ULMFiT model, several other models were used for comparison purposes, including TF-IDF vectors in combination with Logistic Regression, XGBoost, and Feed-forward Neural Networks. The weak labels were generated by Snorkel, with labeling functions based on less than 1,000 labeled tweet samples, and applied to an unlabeled set consisting of around 24,000 tweets. The best-performing end model was the ULMFiT classification model, which in the weakly supervised setting with transfer

learning achieved a precision score of 0.95 and a recall score of 0.39 on the test set, corresponding to an accuracy of 80%. The weakly supervised approaches achieved a better result than the supervised with an increase of 0.27 higher recall and 0.05 higher precision than the best model in the supervised setting trained on ground truth labeled tweets. Given the promising results for the weakly supervised setting, a similar approach will be adapted to the fake news domain in this thesis. Two of the simplest classifiers used in Starosta (2019), namely Logistic Regression and XGBoost, will be used as baselines with TF-IDF vectors used as textual representations in this thesis. However, instead of ULMFiT, BERT-based models will be utilized. The research on anti-Semitic tweets is the most similar to our binary text classification task using a content-based weak supervision approach discovered in the literature. We bear in mind that tweets are shorter than news articles and that detecting anti-Semitism is a more straightforward problem than fake news detection. The scores obtained in this research and the results of our adapted fake news system is therefore not directly comparable.

3.3 Fake News Datasets

Numerous fake news datasets exist, which have different features and characteristics, and most importantly, varying quality. Several of these were investigated in order to select a suitable dataset for this work. The following section will provide an overview of the investigated datasets and their properties. Only datasets containing articles in English were considered, and the dataset selected in this work is further described in Section 4.3

Table 3.1 shows an overview of the datasets considered. The first column represents the name of the dataset or the name of the repository containing the dataset. The *type of content* denotes the features included, i.e., whether the dataset contains full news articles, short statements, tweets, or social context. The *size* column shows the number of included samples, and the *classes* column gives the number of possible labels assigned to the samples, meaning how fine-grained the labels are. The *labeling method* denotes the way that the labels are annotated to the samples, and can be for instance source-based or manual annotation. The last column, *sources*, states the source(s) of the data.

The different datasets examined vary greatly with regards to especially size, features, and labeling method. Most of the datasets are relatively small in size, however five datasets contain over 500,000 samples, namely Fakeddit (Nakamura et al., 2020), FakeNewsCorpus (Szpakowski, 2018), NELA-GT-2018 (Norregaard et al., 2019), NELA-GT-2019 (Gruppi et al., 2020) and NELA-GT-2020 (Gruppi et al., 2021). The NELA-GT datasets have been released each year since 2018 and contain news articles gathered from various news outlets spanning through the whole of the corresponding calendar year, annotated by source-based labels. A NELA-2017 dataset (Horne et al., 2018) was also published. However, it did not include labels and is therefore not included in the overview. Since 2018, the collection method and label annotation have been updated and improved for each new release of the NELA-GT dataset. Especially NELA-GT-2020, released in April 2021, is more extensive and includes articles from more news outlets than its predecessors.

In the case of Fakeddit, the datasets containing above 500,000 samples are labeled by their source or subreddit. Information about the source or subreddit has been used to label all associated samples, thus providing a simplified but inexpensive labeling strat-

egy. As manually labeling is more time-consuming, we notice that the datasets containing manually annotated labels generally contain fewer samples than the larger source-based datasets. The smaller, manually labeled datasets include FakeNewsNet (Shu et al., 2019), FNID (Amirkhani, 2020), LIAR (Wang, 2017), and MisInfoText (Asr and Taboada, 2019b), which all contain between 300 - 16,000 samples. FakeNewsNet is the only one of these datasets containing contextual features in the form of social engagement information, while MisInfoText abd FakeNewsNet are the only ones that contain full articles. MisInfoText is a collection of three datasets, and two of these have been included in the overview, as the third only includes 33 samples. MisInfoText B denotes the dataset gathered from Buzzfeed, while MisInfoText S denotes the dataset gathered from Snopes.

A dataset that stands out from these smaller, hand-labeled datasets is FEVER (Thorne et al., 2018), a manually labeled dataset containing over 185,000 samples. However, FEVER consist of short fact-checked statements, not complete articles. The FA-KES dataset (Abu Salem et al., 2019) also stands out from the other datasets, as all the articles are related to the Syrian War, and it is the only one of the datasets that obtains labels by crowd-sourcing. Two of the datasets containing social engagement information are Getting Real About Fake News (Risdal, 2016) and the Twitter dataset (Helmstetter, 2017). While the Getting Real About Fake News dataset contains only fake news samples, the Twitter dataset contains tweets belonging to both classes. The Twitter dataset is the only one evolved around detecting fake news on Twitter, and the social engagement information includes retweets, favorites, and comments. The labels are annotated by assigning labels based on the reliability of the authoring Twitter user.

Table 3.1: Overview of examined fake news datasets.

Dataset	Type of content	Size	Classes	Labeling method	Sources
Fakeddit	Reddit posts incl. images, Social Engagement	1,063,106	2,3,6	Subreddit theme-based	Reddit
FakeNewsCorpus	Articles	9,408,908	10	Source-based	Opensources
FakeNewsNet	Articles, Social Engagement	1,056	2	Manual	Politifact
FA-KES	Syrian War Articles	804	2	Crowd-sourced	15 News Outlets
FEVER	Short statements	185,445	3	Manual	Wikipedia
FNID	Statements	15,212	2	Manual	Politifact
Getting Real About Fake News	Articles, Social Engagement	12,999	5	Source-based (BS Detector)	244 Websites
LIAR	Short statements	12,836	6	Manual	Politifact
MisInfoText B	Articles	1,380	4	Manual	Buzzfeed
MisInfoText S	Articles	312	5	Manual	Snopes
NELA-GT-2018	Articles	713,534	3	Source-based (8 assessment sites)	194 News Outlets
NELA-GT-2019	Articles	1,118,821	3	Source-based (7 assessment sites)	260 News Outlets
NELA-GT-2020	Articles	1,779,127	3	Source-based (Media Bias/Fact Check)	519 News Outlets
Twitter	Tweets, User Info, Social Engagement	401,414	2	User-based	Twitter

Chapter 4

Method

This chapter will present the method and implementations used in the proposed weak supervision system. First, the tools used for developing the system are presented in Section 4.1. Second, the overall system architecture is presented in Section 4.2. Third, the datasets used are introduced in Section 4.3, and the preprocessing techniques are described in Section 4.4. The feature engineering steps are given in Section 4.5, and sections 4.6 and 4.7 introduces the weak labeling systems, namely the automatic Snorkel and the automatic Snuba systems. The document representations used as the input to the end models are described in Section 4.8. Finally, the end models are presented in Section 4.9 and evaluated as explained in Section 4.10.

4.1 Tools

The Python libraries used for implementation in this thesis are listed and explained below.

Matplotlib[1]	Library for data visualizations and creation of plots in Python.
NLTK[2]	Library of NLP tools used for preprocessing, tokenization, lemmatization, sentiment analysis, POS-tagging and more.
Numpy[3]	Numerical Python library for scientific programming.
Pandas[4]	Data analysis library offering data structures such as data frames for containing data, with built-in methods for loading and saving data to CSV and Pickle formats.

[1] `https://matplotlib.org`, Last accessed: 07.06.2021
[2] `https://www.nltk.org`, Last accessed: 07.06.2021
[3] `https://www.numpy.org`, Last accessed: 07.06.2021
[4] `https://pandas.pydata.org`, Last accessed: 07.06.2021

PyTorch[5]	Machine learning library providing neural network implementations and tensor data structures optimized for GPU computing.
SentiWordNet[6]	Lexical resource for sentiment analysis and opinion mining.
Scikit-learn[7]	Machine learning library providing built-in implementations of a wide range of machine learning algorithms and methods for calculating evaluation metrics. Used for metric calculation and implementation of the Logistic Regression classifier in this work.
Simple Transformers[8]	Transformers library built on top of Hugging Face[9], which allows for quick implementation of a range of transformer models. Used for implementation of the BERT-based models in this work.
Snorkel[10]	A framework for programmatically building training sets from unlabeled data. Used in the manual and automatic Snorkel-based weak labeling systems.
Snuba[11]	A system for automatic labeling of training data based on a small labeled dataset. Generates a set of heuristics based on the labeled data that efficiently assign weak labels to the unlabeled data. The tool is used in the automatic Snuba weak labeling system in this work.
TextBlob[12]	Python library providing a simple API for textual processing such as sentiment analysis, POS-tagging, noun phrase extraction and more. Used to generate sentiment analysis-based numerical features in this work.
Weights & Biases[13]	Framework for visualizing model training and hyperparameter tuning, which is supported natively by the Simple Transformer models. Used for tuning hyperparameters in

[5] `https://pytorch.org`, Last accessed: 07.06.2021
[6] `https://github.com/aesuli/SentiWordNet`, Last accessed: 07.06.2021
[7] `https://scikit-learn.org/stable`, Last accessed: 07.06.2021
[8] `https://simpletransformers.ai`, Last accessed: 07.06.2021
[9] `https://huggingface.co`, Last accessed: 07.06.2021
[10] `https://www.snorkel.org`, Last accessed: 07.06.2021
[11] `https://github.com/HazyResearch/reef`, Last accessed: 07.06.2021
[12] `https://textblob.readthedocs.io`, Last accessed: 07.06.2021
[13] `https://wandb.ai`, Last accessed: 07.06.2021

	the BERT-based models with Bayesian optimization in this work.
XGBoost[14]	Library providing easy and efficient implementation of the XGBoost classifier. Used to implement the XGBoost end model in this work.

4.2 System Architecture

The proposed weak supervision system consists of two main components: the weak labeling systems, and the end models for the classification task. The overall system architecture and flow are shown in Figure 4.1. First, the fake news datasets was preprocessed, and feature engineering was applied to extract numerical features. Second, three weak labeling systems were evaluated, namely manual Snorkel described in Section 1.4, automatic Snorkel, and automatic Snuba. The best-performing weak labeling system was chosen to provide weak labels further used in the classification models. Third, the document representation of the articles was given as input to each classifier, namely Logistic Regression (LR), XGBoost, ALBERT, XLNet and RoBERTa, for either training or prediction.

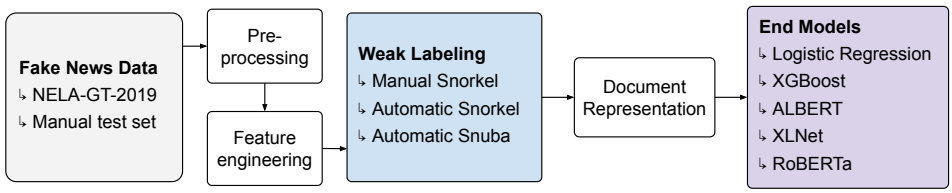

Figure 4.1: System overview. The two main components are the weak labeling system and the end models. First, the fake news datasets was preprocessed, and feature engineering was applied to extract numerical features. Second, three weak labeling systems were evaluated. Third, the document representation of the articles was given as input to each classifier, namely Logistic Regression (LR), XGBoost, ALBERT, XLNet and RoBERTa, for either training or prediction.

4.3 Data

In data-driven approaches such as machine learning, the importance of high-quality data can not be underestimated. Unfortunately, such datasets can be hard to obtain. For the experiments in this thesis, efforts were therefore made to acquire a suitable dataset. After discovering a lack of quality in many of the fake news datasets presented in Section 3.3, it was important to develop requirements for selecting a dataset.

When selecting a suitable dataset, mainly four properties were considered:

[14] `https://xgboost.readthedocs.io`, Last accessed: 07.06.2021

1. **The size of the dataset**, which should be large as most machine-learning models require sufficient amounts of data for training. The size needed will nevertheless vary and depend on the specific dataset and model used.

2. **The dataset features**, which are, for example, article title, article content, author information, metadata, and contextual information. The content-based features title and article content are the main focus of this work, and are therefore required in the selected dataset.

3. **The class balance**, should be balanced to make sure the model has seen enough samples of each class to be able to separate the classes. Simply having a dataset with only real news would not be sufficient for this task, as the model also has to learn the traits of the fake articles. Hence, heavily imbalanced datasets would likely give the model a tendency towards predicting the most common class.

4. **The labeling method** used to assign labels to the articles. Inaccurate labels degrade the performance of the machine learning model. Thus, manual labeling is the best approach, but methods like crowd-sourcing or assigning source-based labels is often applied as manually fact-checking articles is time-consuming.

No perfect datasets exist, so accepting a trade-off between the mentioned properties is necessary. All the investigated datasets are listed in section 3.3, and the chosen dataset for training is NELA-GT-2019. Additionally, a fact-checked test set was gathered by assembling multiple smaller fact-checked datasets and manually collected articles, as explained in Section 4.3.2.

4.3.1 NELA-GT-2019

The primary dataset used in this work is NELA-GT-2019 developed by Gruppi et al. (2020) and is the basis for the training and validation sets. The dataset consists of almost 1.12 million articles gathered from 260 news outlets, all published in 2019. During the work on this thesis, NELA-GT-2020 with articles from 2020 was also released. However, due to time limits, the 2020 version has not been employed in this work.

The NELA-GT-2019 dataset was chosen for multiple reasons. First, it offers substantial amounts of data within each class, allowing for creating a class-balanced subset. Second, it contains the title and content of the articles, which is crucial for a content-based approach. Finally, the dataset is very well documented, and the authors have clearly explained each step of how the dataset was gathered and annotated. The labels are obtained by gathering source-based veracity scores from seven different assessment sites, which are websites assessing the credibility of sources. Using multiple assessment sites increases the credibility of the source-based labels. There is still no guarantee that a source considered unreliable has not published true articles, or that a credible source has not published articles containing false information. This also depends on the definition used to define fake news, i.e., whether an article containing mostly factual information but some wrong facts should be considered fake or not. Nevertheless, we assume that the source-based labels approximate the ground truth, and are therefore considered to be ground truth labels for this task.

Dataset Characteristics

The dataset includes nine features for each article, as shown in Table 4.2. The features cover publication date, publication time, article source, title, content, author, and time of collection, and a unique id for each article. The news outlet source is combined with the veracity scores from each assessment site to obtain labels for each article. The assessment sites used are Media Bias/Fact Check [15], Pew Research Center[16], Wikipedia[17], OpenSources[18], AllSides[19], BuzzFeed News[20] and Politifact[21]. The seven assessment sites cover different dimensions of veracity, including reliability, political bias, transparency, consumer trust, and adherence to journalistic standards. There are four possible aggregated labels: reliable, mixed, unreliable, and unknown (i.e., no sites contained information about that source). A source will be labeled unreliable if the factual reporting is *low* or *very low*, mixed if the factual reporting is *mixed*, and reliable if the factual reporting is *high* or *very high*. In terms of the aggregated labels, 83 sources were labeled reliable, 50 were labeled mixed, 50 as unreliable, and 77 sources' factual reporting was unknown and remained unlabeled.

Table 4.2: Features of the NELA-GT-2019 dataset.

Feature	Type	Description
Id	Text	Article id
Date	Text	Publication date (YYYY-MM-DD format)
Source	Text	Name of the source
Title	Text	Title of the article
Content	Text	Content (body) of the article
Author	Text	Author of the article (if available)
Published	Text	Publication date as provided by source
Published_UTC	Integer	Publication time (unix time stamp)
Collection_UTC	Integer	Collection time (unix time stamp)

The complete NELA-GT-2019 dataset consists of 1,118,810 articles in total, and the distribution between labels is shown in Figure 4.2. AS can be seen, most articles originate from reliable sources and fewest from unreliable sources. Moreover, many articles are unlabeled or assigned a mixed label, meaning they can contain a mix of factual and false information. In the experiments of this thesis, only reliable and unreliable articles were used to simplify the evaluation of the weak labeling systems and end models. An article originating from an unreliable source is called a *fake* article, and an article from a reliable

[15] https://mediabiasfactcheck.com/, Last accessed: 28-05-2021

[16] https://www.pewresearch.org/, Last accessed: 28-05-2021

[17] https://en.wikipedia.org/wiki/List_of_fake_news_websites, Last accessed: 28-05-2021

[18] OpenSources website does no longer exist

[19] https://www.allsides.com/, Last accessed: 28-05-2021

[20] https://github.com/BuzzFeedNews/2017-12-fake-news-top-50, Last accessed: 28-05-2021

[21] https://www.politifact.com/, Last accessed: 28-05-2021

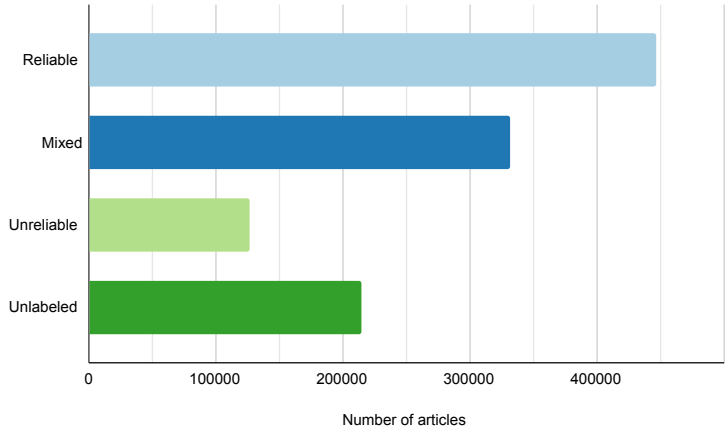

Figure 4.2: Articles per label class in NELA-GT-2019.

source is called a *real* article. For further work with weak supervision, the rest of the articles could have been weakly labeled to expand the dataset.

Before applying machine learning techniques, it is important to get familiarized with the dataset and its characteristics. Therefore, key characteristics were analyzed on a subset of the articles labeled fake or real. The average values of features from the most interesting findings of the data analysis are presented in Table 4.3. By comparing the average values for the fake and real articles, we notice that the average word count and the ratio of exclamation marks, uppercased words, and proper nouns in titles were more prevalent in fake news stories. Much of the same was observed for the content of the fake articles, but real news stories' content typically had a higher word and sentence count. The ratio of stop words was almost identical for the two classes, while there was an escalated use of quotes in the content of fake articles. These findings agree with previous research on the dissimilarities of fake and real articles, as presented in Section 3.1, and the NELA-GT-2019 dataset is therefore considered to be suitable for the task of fake news detection.

Dataset Examples

To get further familiarized with the dataset, some random samples of both classes were inspected. Below are five randomly selected article titles from the fake and real class. Examples from articles labeled as *real*:

1. Mailing Free Home HIV Tests Helps Detect More Infections (Source: U.S. News & World Report)

2. Global Wealth Gap Would Be Smaller Today Without Climate Change, Study Finds (Source: The New York Times)

3. Student paramedic dies in Burton ambulance collision (Source: BBC)

4. Chinese arthouse film breaks box office records after viewers mistake it for romcom (Source: The Guardian)

5. India: Teen who accused BJP leader of rape critical after crash (Source: Al Jazeera)

Examples from articles labeled as *fake*:

1. Is the 'Extinction Rebellion' a Scheme Cooked Up by the Alt-Establishment? (Source: Global Research)

2. Rapper Gets PO'd When Fan Won't Yell 'F**K Trump' On Command... Throws Him Out (Source: Clash Daily)

3. Accused Woman Beater and Farrakhan Supporter Keith Ellison Sworn In as Minnesota Attorney General #MeToo (Source: The Gateway Pundit)

4. A New World Order: Brought to You by the Global-Industrial Deep State (Source: Humans Are Free)

5. 80-year-old Israeli Rabbi beaten by right-wing Israeli settlers for trying to protect Palestinian farmers (Source: Signs of the Times)

The fake titles in these examples have a higher word count than the real ones. However, the titles are hard to separate without possessing knowledge of typical fake news topics. To the observant eye, proper nouns are more frequent for some of the fake examples, and the second fake example is distinguishable from the rest by the use of swear words, which might have the purpose of catching the reader's attention. This is a common strategy of fake news authors as a provocative title will generate more clicks on social media (Rashkin et al., 2017).

Table 4.3: Data analysis of NELA-GT-2019 articles, showing average values for each class.

		Real	**Fake**
Title	Word count	11.48	**12.00**
	Stop word ratio	0.25	0.25
	Exclamation mark ratio	0.005	**0.023**
	Uppercased word ratio	0.02	**0.04**
	Proper nouns ratio	0.33	**0.51**
Content	Word count	**535**	522
	Sentence count	**27.4**	24.0
	Words per sentence	21.47	**24.27**
	Stop word ratio	**0.42**	0.41
	Exclamation mark ratio	0.007	**0.02**
	Uppercased word ratio	0.01	**0.02**
	Quote marks ratio	0.012	**0.017**

Dataset Preparation

Some initial dataset-specific cleaning was applied before the text preprocessing described in Section 4.4. The specific text cleaning included removing articles from a newspaper called *Der Spiegel*, which contained articles written in German. Additionally, instances that were missing either title or content were removed, as we require both features for the experiments done in Section 5.

To avoid handling imbalanced datasets in the model, the subset of only reliable and unreliable instances was balanced by class. The balanced subset had a total size of 252,006 articles, i.e., 126,003 of each class. This size was chosen because 126,003 is the total number of unreliable news articles available after cleaning, which was the class with a limiting number of instances as seen in Figure 4.2. It is the balanced subset with 252,006 articles that is referred to when the NELA-GT-2019 dataset is mentioned later in this thesis.

4.3.2 Test set

A separate and independent test set was acquired to assess the final and realistic performance of the models. The articles in the test set needed to be similar to those in the training data and were therefore collected from similar sources. Another requirement was that the test set consisted of articles that have been manually fact-checked and labeled to ensure that the instances represented the ground truth. Ideally, the articles in the test set would be published in 2020 to test that the system could detect articles from succeeding time periods, but such a dataset was not available. Instead, articles published mostly in 2017 and 2018 were collected for the test set, to ensure that no articles were present in the training data, which consists of articles published in 2019. Articles were collected from the FakeNewsNet and MisInfoText datasets, as well as articles gathered manually, resulting in a balanced test set that consists of 434 articles. The number of articles from each source is shown in Table 4.4, and the rest of this section explains how the articles were collected from each source.

Table 4.4: Datasets and number of articles of each class used in the manually labeled test set

	FakeNewsNet	**MisInfoText**	**Manual Snopes**	**Total test set**
Fake articles	217	-	-	217
Real articles	73	58	86	217
Total	290	58	86	434

FakeNewsNet

FakeNewsNet[22] is a fake news dataset repository assembled by Shu et al. (2017a,b, 2019). The repository consists of two datasets, one from political and one from entertainment sources, including news content, social context, and spatiotemporal information. Both

[22] `https://github.com/KaiDMML/FakeNewsNet`, Last accessed: 14.6.2021

datasets have reliable ground truth labels from fact-checking websites. We used the political news dataset gathered from Politifact.com[23] as this was the dataset most similar to the training data. Social context from Twitter was available but not used, as tweets are not present in the training data. According to the dataset specifications, the political dataset contains 1056 articles, 432 fake and 624 real from 2017 and 2018. The complete dataset is not shared on GitHub and needs to be downloaded from the FakeNewsNet API. The articles are crawled at the time of download, and consequently, some articles had been removed or changed URL. Therefore, a considerable amount of cleaning was applied to prepare the data. After data cleaning, the dataset consisted of 217 fake and 73 real articles. The steps taken to clean the data are described below:

1. Remove empty articles that either has missing title, content, or both.

2. Remove samples that do not contain an article, e.g., sites that have only returned 'cookies need to be enabled' or 'this domain has been moved.'

3. Remove content that is not considered news articles, like factual articles and interview transcripts.

4. Remove articles that are not currently found on Politifact's pages.

5. Remove text that is not part of the article, e.g., advertisement.

MisInfoText

MisInfoText is a collection of fake news datasets with reliable veracity labels gathered by Asr and Taboada (2019a,b). The authors have created an overview of multiple fact-checked fake news datasets and assembled three fake news datasets from different sources. Their Snopes dataset (Referred to as MisInfoText S in Table 3.1) was the most suitable with regard to similar article topics as the training data. The articles in the dataset are fact-checked by Snopes and consist of 312 articles in total. Only articles from the *real* class were used from this dataset to make the total test set balanced by class. After manually cleaning this dataset, i.e., removing articles by the same approach as in Section 4.3.2, only 58 articles remained.

Manually Collected Snopes Articles

After joining the cleaned FakeNewsNet and MisInfoText datasets, the number of fake articles still exceeded the number of real articles. Therefore, in order to construct a test set with a balanced distribution of fake and real articles, we manually gathered the remaining real news articles from Snopes' fact-checking archives[24]. The result was 86 fact-checked articles belonging to the *real* class, making the final test set balanced by class.

[23] `https://www.politifact.com/`, Last accessed: 14.6.2021
[24] `https://www.snopes.com/fact-check/`, Last accessed: 29.05.2021

4.4 Preprocessing Data

The raw text data provided in the NELA-GT-2019 dataset is unstructured, and all articles are formatted as a single token. In order to transform the text into a more manageable format for later tasks, it was necessary to preprocess the data. This was done using several different Natural Language Processing (NLP) techniques provided by the Natural Language Toolkit (NLTK) package. Which processing techniques are necessary depends on the task at hand; for instance, the Part-of-Speech (POS) tagging task requires removing punctuation in the text. On the other hand, for a quote count task, it is necessary to include all punctuation. To provide requirements for subsequent tasks, the preprocessing was done in incremental steps where each intermediate state was retained for later use by adding it to the data as a new textual feature. The preprocessing techniques applied to each stage are shown in Figure 4.3. A total of seven new textual features were added, as represented by numbered circles in the figure, which we later will refer to as the seven *preprocessing stages*. Each step is applied to both title and content of an article.

The preprocessing of the data has previously been described in From and Netland (2020), which is the basis for this thesis. However, as the reader might be unfamiliar with this work, a brief repetition of the preprocessing stages is included to understand the system as a whole. This section will thus describe the preprocessing techniques applied to the data and explain the seven stages.

Stage 1. The first stage consisted of a single step which was to divide the raw text into word tokens. Word tokenization was performed using NLTK's word tokenizer[25], which splits text into word tokens while keeping punctuation as individual tokens.

Stage 2. For the second stage, the raw text was first split into sentence tokens. This step was performed using NLTK's sentence tokenizer[26] which creates sentence tokens by splitting the text on ending punctuation marks (period, question mark, and exclamation points). Similar to stage 1, sentence tokenization keeps all punctuation within the text. However, as stage one included all punctuation, the feature added in stage two is cleaned by removing punctuation using a regular expression. In the English language some words include punctuation to provide a new meaning to the word. Examples are possessive nouns indicated by the use of *'s* at the end of a noun, e.g., *word's meaning*, abbreviations indicated by the use of periods, e.g., *United States* $\rightarrow$ *U.S.*, and the use of '-' for combining existing words into new words. Additionally, the apostrophe is used to mark contractions of words such as *I am* $\rightarrow$ *I'm*. It is desirable to keep as much information as possible, so a regular expression is used to find the aforementioned cases and keep them in the text.

Stage 3. For stage three, the same process of punctuation removal was performed on the word tokens generated in stage one. Then all stop words were removed from the text. Which stop words to remove was defined by a stop words corpus provided in NLTK[27].

Stage 4. In stage four, the word tokens with punctuation removed in stage three were used. Instead of stop word removal, this stage performed case normalization on all words.

[25] `https://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.word_tokenize`, Last accessed: 18.05.2021

[26] `https://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.sent_tokenize`, Last accessed: 18.05.2021

[27] `https://github.com/nltk/nltk_data/blob/gh-pages/packages/corpora/stopwords.zip`, Last accessed: 18.05.2021

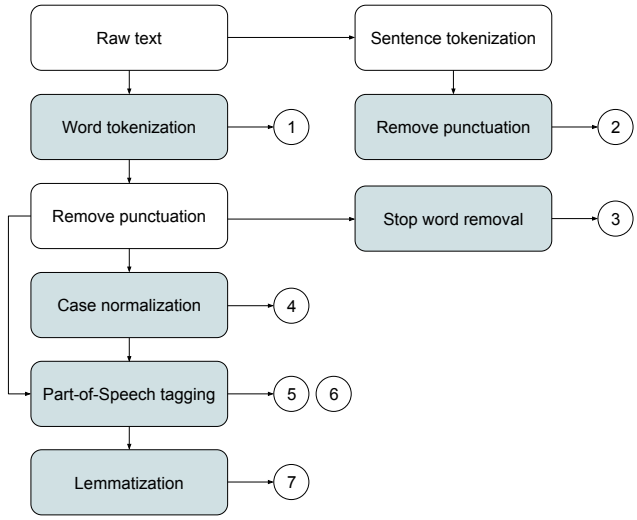

Figure 4.3: The natural language processing steps are conducted for preprocessing the data before feature generation. The rounded squares represent preprocessing steps. The numbered circles represent the resulting feature from a preprocessing step, referred to as a stage, and are kept as an additional textual feature in the data. A total of seven new textual features were added to the dataset.

Case normalization can be applied in several ways, but in this work, it was performed by lowering the case of all words using Python's built-in lowercasing function[28].

Stage 5 and 6. In both of these stages, NLTK's Part-of-Speech tagger[29] was used to assign POS tags to each word from the tokenized word input. Two different versions were created, one with case normalization (stage 5) and one without (stage 6).

Stage 7. For a simpler generation of TF-IDF vectors later, lemmatization was performed. Lemmatization was done using WordNetLemmatizer[30] on the POS-tagged words from stage 6 (POS-tagged word tokens with case).

4.5 Feature Engineering

The manual weak labeling system proposed in From and Netland (2020) is based on features extracted from the content of the articles. The same numerical features are used in this work, so this section will briefly cover the types of features generated in the previous work. Additionally, two new feature types were added in this work, namely adverb and SentiWordNet features. A complete list of all the generated features can be found in Appendix A. As all the generated features had numerical values, the features will later be referred to as the *numerical features*.

[28] `https://docs.python.org/3/library/stdtypes.html#str.lower`, Last accessed: 19.05.2021

[29] `http://www.nltk.org/_modules/nltk/tag.html#pos_tag`, Last accessed: 18.05.2021

[30] `https://www.nltk.org/api/nltk.stem.html#module-nltk.stem.wordnet`, Last accessed: 18.05.2021

The features generated in From and Netland (2020) were chosen based on the research presented in Section 3.1 that has studied the characteristics of fake and real news articles. Four types of features were explored: stylistic features, Part-of-Speech related features, sentiment analysis features, and complexity features. The stylistic features capture the author's writing style, such as the total length of an article, the average word length, and the use of exclamation marks and uppercase words. The POS-related features measure the frequency and ratio of the different word classes such as the number of verbs, adjectives, and nouns related to the total number of words. The sentiment features cover the text's polarity and subjectivity scores. Finally, the complexity features extract information from the content that is not readily available in the text, such as the type-token ratio, which measures the lexical redundancy in the text. For a more elaborate explanation of the methods used for generating these features, the reader is encouraged to read From and Netland (2020).

4.5.1 Additional features

In the weak labeling systems based on Snorkel and Snuba, some of the generated heuristics will cover a large part of the instances, while others cover very few. There is a probability that the coverage of heuristics overlaps, leaving some instances unlabeled by all the heuristics. To increase diversity in the instances labeled by the weak labeling systems, it is necessary to have heuristics that capture different characteristics of the data. The number of heuristics in the weak labeling system is directly linked to the number of numerical features generated from the content of the articles. To improve the number of possible heuristics, some additional numerical features were added in this work, resulting in a total of 68 extracted features.

Adverbs

As presented in 3.1, Rashkin et al. (2017) found that real news in general uses more comparative and assertive words than fake news articles. The only features related to comparative words in the manual weak labeling system were the adjective count and frequencies in the text, whereas adverbs were ignored. To gain more insight into the use of comparative words, the adverb count and frequency in the content and title of the articles were included in this work. The number of adverbs is found based on POS tags, and the frequencies are found by normalizing the adverb count with the word length of the title or the content.

SentiWordNet

The manual weak labeling system used TextBlob for sentiment analysis. Figure 4.4 shows the distribution of the articles in terms of the negative polarity score calculated by the TextBlob framework using word tokens. As one can see, the fake and real articles have almost identical distributions, thus the features generated by TextBlob proved to be of minor importance for assigning weak labels to instances. The reason for this was not fully explored. However, this might be due to the language style in movie reviews differing too much from the style of news articles.

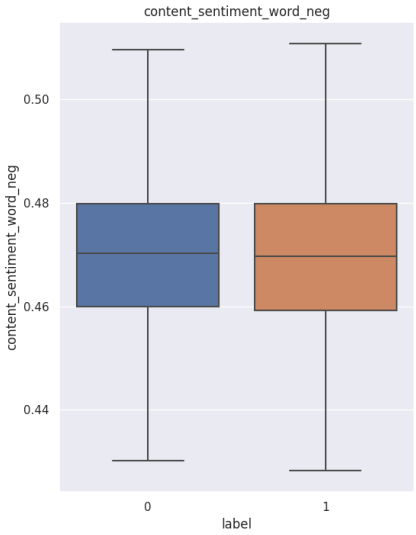

Figure 4.4: The distribution of negative polarity scores of the articles as calculated by TextBlob using word tokens.

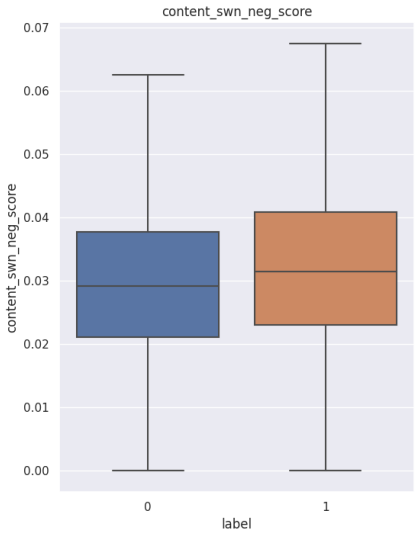

Figure 4.5: The distribution of negative polarity scores of the articles as calculated by SentiWordNet using word tokens.

As shown in Bhutani et al. (2019), the sentiment of a news article can be a good indicator of its reliability. Another method for calculating the sentiment of articles, based on the SentiWordNet lexica, was thus implemented to achieve separable distributions for the sentiment features. For example, Figure 4.5 shows the distribution of the negative polarity score of the articles as calculated by SentiWordNet using word tokens. Compared to the identical feature based on TextBlob, the SentiWordNet approach resulted in more separable distributions. This was also the case for the other sentiment features, namely the positive polarity and subjectivity scores.

4.6 Automatic Weak Labeling System with Snorkel

In order to gain an improvement in performance and efficiency of fine-tuning thresholds, adjustments were made to the manual weak labeling system. This resulted in a new Snorkel-based weak labeling system that automatically finds suitable thresholds for separating the data, later referred to as the *automatic weak labeling system in Snorkel* or the *automatic Snorkel system*. This section presents the improvements made.

4.6.1 System Overview

The pipeline of the automatic Snorkel system is shown in Figure 4.6. First, the instances within the training data were described through descriptive statistics, and thresholds for

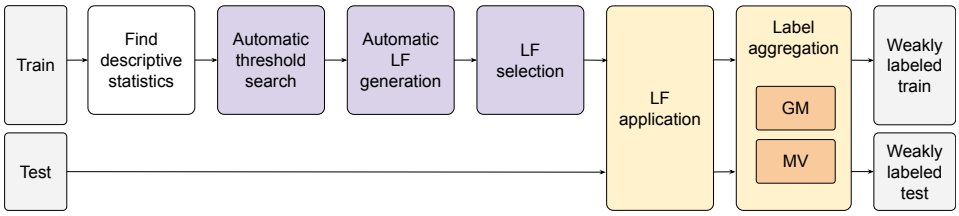

Figure 4.6: The pipeline of the automatic weak labeling system in Snorkel. The purple color marks the components developed in this work. The white component is preliminary processing, yellow components marks the processes handled by Snorkel and the gray components are the input and output of the system.

separating instances based on a feature value were found, and is described in Section 4.6.2. The thresholds were then used to automatically generate labeling functions and is described in Section 4.6.3. A set of labeling functions (LFs) was then selected as an ensemble of heuristics which was the basis for the weak labeling system. The selection of LFs and how the heuristics label new instances are described in Section 4.6.4.

4.6.2 Threshold Search

In the original weak-labeling system, the thresholds set for defining the labeling functions were found manually by analyzing the feature distributions of the real and fake instances, as described in From and Netland (2020). As the thresholds directly impact which weak label is assigned for an instance, threshold tuning is a vital part of improving the weak-labeling system. Manually tuning these thresholds is a time-consuming process, and the tuning of thresholds thus became a bottleneck for improving the performance of assigning weak labels. To overcome such a time-exhaustive task, we propose a way to automatically find and tune the thresholds by implementing a method we have called *automatic threshold search*.

As a prerequisite to threshold search, descriptive statistics of the distribution of all numerical features were generated for the real and fake instances. This was done using the `pandas.DataFrame.describe`[31] method provided by the Pandas package for Python. The statistics generated were the maximum, minimum and mean value, instance count, standard deviation and percentiles .05, .10, .15, .20, .25, .30, .50, .70, .75, .80, .85, .90 and .95 for each feature for both the real and the fake instances. In statistics, a percentile is a value that a given percentage of the values in the frequency distribution falls below. For instance, a feature with a 5% percentile value of 5 entails that 5% of instances has a feature value of 5 or below 5. An example of the descriptive statistics generated for a unique feature, in this case the *title_word_count* feature, is shown in Table 4.5.

Based on these statistics, the task is to select feature values that best define whether an instance belongs to the fake or real class. The selected values are called *thresholds* for the corresponding feature. It is possible to set many thresholds for each feature, so

[31] `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html`, Last accessed: 14.6.2021

Table 4.5: Descriptive statistics of fake and real instances for the feature *title_word_count*

title_word_count	Fake instances	Real instances
count	100802.0	421044.0
mean	11.986	11.400
std	3.932	3.678
min	1.0	1.0
5%	6.0	6.0
10%	7.0	7.0
15%	8.0	8.0
20%	9.0	8.0
25%	9.0	9.0
30%	10.0	9.0
50%	12.0	11.0
70%	14.0	13.0
75%	14.0	13.0
80%	15.0	14.0
85%	16.0	15.0
90%	17.0	16.0
95%	19.0	18.0
max	42.0	37.0

an automatic approach requires standardization of which thresholds to find. This work focuses on primarily finding an *upper* and a *lower* threshold, t_n^u and t_n^l, respectively, for each feature f_n. When selecting the thresholds, finding values that cover a considerable part of the data is desirable. However, there is often a trade-off between the coverage of instances and mislabeling of instances. To find a good balance between high coverage and few mislabels, the idea is to exploit that if the real and fake instances have different distributions, the density of one type of instance, e.g., the real class, will be greater within a certain range of feature values.

To measure the difference of the fake and real distributions for a feature, their percentiles' value was compared by calculating the absolute difference between them. For each feature f_n, 13 percentiles were evaluated for both the fake and real distribution of the feature. We denote a value of a percentile within the fake distribution as $q_{n,k}^f$, and a value for the real distribution as $q_{n,k}^r$, where n denotes the current feature, k is the percentile evaluated, f stands for *fake* and r stands for *real*. If the absolute difference between the same percentile value for the fake and the real distributions was above a limit, they were considered to be distributed differently, and the percentile was chosen as a threshold. There are four possible ways the real and fake instances can be distributed relative to each other. This will determine which percentile is chosen as the threshold and which label should be assigned based on this threshold. The four possible cases are shown in Figure 4.7.

The thresholds can be tuned by altering the value of the limit for the absolute difference, which we will refer to as the *relative difference limit* (RDL). A smaller RDL value means the distributions are more similar and will result in more LFs being generated but

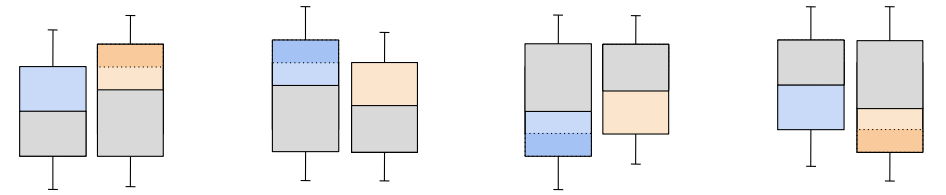

Figure 4.7: The four possible cases of percentile comparison for the chosen thresholds t_n^l and t_n^u. The colored parts of the box plots indicated which part of the distribution is being evaluated, where the real distributions are shown in blue, and the fake distributions are shown in orange. The grey areas are irrelevant for the current evaluation. The darker shades marks the $NAD_{n,k}$.

also introduce more mislabeled instances. A higher RDL value results in a more restrictive approach that only creates LFs for the features with distributions that differ significantly, resulting in fewer mislabels. If the distributions differ,

Consider Figure 4.8 which shows two box plots of distributions for an imaginary feature, one for the real and one for the fake instances. A box plot is a typical representation of the distribution of a dataset that marks the 25th, 50th, and 75th percentiles. These percentiles are commonly named the lower, median, and upper quartiles, and are are displayed for both distributions in the figure. Each distribution can be considered to consist of a lower and an upper part divided by the median. In this example the *lower difference* marks the absolute difference between the lower quartile of the real distribution, $q_{n,0.25}^{r}$, and the lower quartile of the fake distribution, $q_{n,0.25}^{f}$. If the lower difference is *above* the RDL, the percentile of a good separator and the value of either $q_{n,0.25}^{r}$ or $q_{n,0.25}^{f}$ is chosen as t_n^l. In the example, we have a case 3 scenario where the real instances are distributed across lower values than the fake instances, thus $q_{n,0.25}^{f}$ is chosen as t_n^l and the instances with a feature value below this threshold is then considered to belong to the real class. Considering the *upper difference* in the example in Figure 4.8, $q_{n,0.75}^{r}$ could be chosen as a threshold where all instances with a feature value above the threshold are considered fake. Using this approach will potentially find two thresholds for each feature, one for the upper part, t_n^u and one for the lower part, t_n^l.

Suppose the lower difference is *below* the RDL. In that case, the distributions are considered too similar, and the system will evaluate another percentile until the set of percentiles is exhausted. For the lower part, this set consisted of 6 percentiles which were the median and all percentiles below the median down to the 5th percentile. To ensure thresholds covering as many instances as possible, the system evaluated the median first, then the percentile closest to the median, and so on. The exact process was repeated for the upper part of the distribution, where the set consisted of the same number of percentiles, but included other percentiles, namely the median and all percentiles above the median up to the 95th percentile.

The range of possible values differed for each feature. Hence the percentile values had to be scaled to set a global RDL for choosing a threshold. To scale the features,

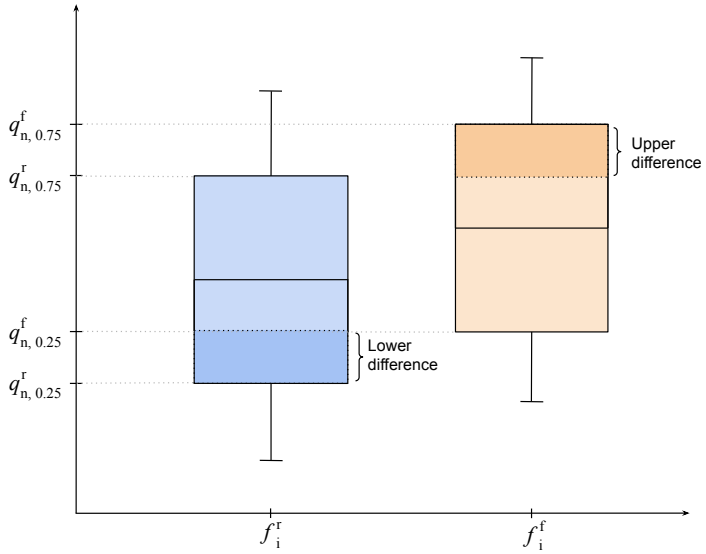

Figure 4.8: Box plots of an example distribution of a feature for real and fake instances. The lower, median and upper quantile, is shown, as well as the lower and upper difference of the distributions.

each percentile value was scaled to range between 0 and 1. Each $q_{n,k}$ is scaled by the difference of the maximum and minimum value of the feature, denoted as $\max_n$ and $\min_n$ respectively, so that

$$\tilde{q}^{*}_{n,k} = \frac{q^{*}_{n,k} - \min_n}{\max_n - \min_n}, \tag{4.1}$$

where $*$ denotes the distribution, either f for fake or r for real.

The scaled values were then used to find a normalized absolute difference (NAD) between the two distributions of a feature, where $NAD_{n,k}$ for a given feature for a given percentile value is defined as

$$NAD_{n,k} = |\tilde{q}^{f}_{n,k} - \tilde{q}^{r}_{n,k}|. \tag{4.2}$$

$NAD_{n,k}$ is then compared to the RDL. More formally, we can say that a percentile value $q_{n,k}$ is chosen as a threshold t_n for a feature f_n if $NAD_{n,k} > RDL$.

4.6.3 Labeling Function Generation

For each threshold found in the threshold search, a labeling function in Snorkel was generated automatically. The purpose of every labeling function, $p_j(x_i)$, was to check a condition, namely whether a feature value of an instance x_i was above or below a threshold and assign the appropriate label $y^{*}_{i,j}$ to that instance, so that

$$p_j(x_i) = y^{*}_{i,j}. \tag{4.3}$$

The label assigned was either REAL, FAKE, or ABSTAIN, denoted as the values 1, 0, and -1, respectively. As two thresholds could be set for each of the 68 numerical features extracted from the data, a total of 136 LFs could potentially be created.

Which label to assign was determined by how the real and fake instances of a feature are distributed in relation to each other. The label assigned for each possible case shown in Figure 4.7, which for the upper part are

- **Case 1:** $q^r_{n,k} < q^f_{n,k}$, so that $q^r_{n,k}$ is chosen as t^u_n and the label REAL or ABSTAIN is assigned so that

$$y^*_{i,j} = \begin{cases} 1, & x_i \geq t^u_n \\ -1, & x_i < t^u_n \end{cases} . \tag{4.4}$$

- **Case 2:** $q^r_{n,k} > q^f_{n,k}$, so that $q^f_{n,k}$ is chosen as threshold t^u_n and the label REAL or ABSTAIN is assigned so that

$$y^*_{i,j} = \begin{cases} 0, & x_i \geq t^u_n \\ -1, & x_i < t^u_n \end{cases} . \tag{4.5}$$

For the lower part, we have

- **Case 3:** $q^r_{n,k} < q^f_{n,k}$, so that $q^f_{n,k}$ is chosen as threshold t^l_n and the label REAL or ABSTAIN is assigned so that

$$y^*_{i,j} = \begin{cases} 0, & x_i \leq t^l_n \\ -1, & x_i > t^l_n \end{cases} . \tag{4.6}$$

- **Case 4:** $q^r_{n,k} > q^f_{n,k}$, so that $q^r_{n,k}$ is chosen as threshold t^l_n and the label FAKE or ABSTAIN is assigned so that

$$y^*_{i,j} = \begin{cases} 1, & x_i \leq t^l_n \\ -1, & x_i > t^l_n \end{cases} . \tag{4.7}$$

4.6.4 Labeling Function Selection

Some LFs may introduce extremely noisy labels. To counter the heuristics generating the noisiest labels, three sets of LFs were evaluated. The first included all LFs generated. The second included all LFs with an individual accuracy above 65%. The third consisted of the 25 LFs of all that performed the best in terms of accuracy. We will refer to the sets as *All*, *Acc* $>$ *65%* and *Top 25*, respectively. The limit of 65% was chosen because this limit for selecting LFs based on their individual accuracy proved to be the best set in the experiments conducted by From and Netland (2020).

Each set of selected LFs was evaluated using both the generative model (GM) and majority vote (MV) approach for aggregating their weak labels.

4.7 Automatic Weak Labeling System with Snuba

Two potential drawbacks of the automatic weak labeling in Snorkel are 1) the potential introduction of heuristics that generate extremely noisy labels and 2) the coverage of each LF not necessarily providing diversity in which instances are labeled. Snuba, however, compensates for these shortcomings when heuristics are selected to the committed set, and a weak labeling system using Snuba was therefore also implemented. The theory behind the automatic Snuba system is presented in Section 2.4.2. Snuba can use any classification model as its heuristic type, but given the time constraint of this work, only three classifiers were employed. These were the built-in models Decision Trees, Logistic Regressors, and k-Nearest Neighbour. Additionally, the user can set the number of features in the subset of the training data that is used to generate a candidate set of heuristics for each iteration. This number is called the *max cardinality* of the system. In the paper where Snuba is introduced, Varma and Ré (2018) found empirically that a max cardinality below 4 was sufficient for most real-world tasks. Thus, for each model used as the heuristic in Snuba, a max cardinality of 1, 2, and 3 was tested.

4.8 Document Representation

We want the end models to generalize beyond the weak labels and avoid the risk of developing an end model that reverse engineers the rules applied by the weak labeling system to the training data when applying the weak labels. Therefore, only the article title and content represented as numerical features was given as input to the end models. Which document representation is suitable depends on the end models used. For example, while Logistic Regression and XGBoost may need more text cleaning and normalization, the BERT-based models are designed to handle raw text input well. Therefore tokenized and lemmatized input text is used to create Term Frequency-Inverse Document Frequency (TF-IDF) vectors for Logistic Regression and XGBoost. Only minimal preprocessing is done on the raw text input for the BERT-based models.

4.8.1 TF-IDF

The theory behind TF-IDF vectors is explained in Section 2.2.1, and was implemented using `TfidfVectorizer`[32] from Scikit-learn. The input text was preprocesing stage 7 described in Section 4.4. To reduce dimensionality, only the N most important features were included, where N_{title} = 1000 and $N_{content}$ = 5000. The title and content features were then concatenated, resulting in an array of total size 6000 for each document. Because all values are between 0-1, further normalization of the features was not necessary.

4.8.2 Preprocessing for BERT-Based Models

The Simple Transformers package takes care of the BERT-specific preprocessing and can take raw text as input without further preprocessing. Thus only simple steps were taken

[32] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`, Last accessed: 21.05.2021

to prepare the data. First, the title and content were joined in a single text. Secondly, the resulting text was trimmed to the max length of tokens that the models can have as input, which is 512 tokens.

4.9 End Models

The five end models used in this work are Logistic Regression, XGBoost, ALBERT, XL-Net and RoBERTa. The models were trained for two learning scenarios, namely the weakly supervised and purely supervised scenario, resulting in ten end models. After training and hyperparameter tuning, the models were evaluated by the F1 score and accuracy metrics as explained in Section 4.10.

4.9.1 Logistic Regression

Logistic Regression was chosen as a baseline model for the other models implemented in this work due to its simplicity of implementation and widespread use for text classification tasks. It has achieved excellent results compared to other simple classifiers on text classification tasks, such as emotion recognition in tweets and short text classification (Wang et al., 2017; Yousaf et al., 2021). The `LogisticRegression`[33] model from Scikit-learn was used for implementation. The LR model inputs the aforementioned TF-IDF vectors and fits the classifier based on the training data. The classifier has multiple hyperparameters that are possible to tune, but in this work, only the regularization hyperparameter `C` was tuned while the other hyperparameters are kept constant. `GridSearchCV`[34] was used for the hyperparameter tuning.

4.9.2 XGBoost

As the work of Helmstetter (2017) showed that XGBoost performed well on a similar task, namely to categorize tweets as fake or real news, XGBoost was implemented in this thesis as well. For the implementation of the model, the `XGBoostClassifier`[35] provided in the Python package of the XGBoost library was used. The `XGBoostClassifier` provides functionality for configuring many hyperparameters that will alter the estimator's learning method. Performing hyperparameter tuning requires time and resources, but due to time constraints there was no opportunity to tune all the hyperparameters. Rather, a smaller set of hyperparameters that were considered to be the most important were tuned. Similar to the Logistic Regression model, `GridSearchCV` was used to find the best values for each hyperparameter. The chosen hyperparameters for tuning were:

- **max_depth**: the maximum depth of the boosted trees.
- **learning_rate**: the step size used when updating the feature weights.

[33] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`, Last accessed: 21.05.2021

[34] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`, Last accessed: 14.06.2021

[35] `https://xgboost.readthedocs.io/en/latest/`, Last accessed: 29.05.2021

- **colsample_bytree**: the ratio of subsamples of the columns. Subsampling occurs once for every tree constructed.

- **subsample**: the ratio of subsamples of the training instances. If the value is 0.5, the model randomly selects 50% of the instances in the training data before constructing the trees. Subsampling occurs once in every boosting iteration.

- **gamma**: the minimum loss required for continuing to partition from a leaf node. A larger gamma gives a more conservative model.

4.9.3 BERT-Based Models

BERT is a language model that has achieved state-of-the-art results for many NLP tasks (Devlin et al., 2018). A considerable advantage of the BERT architecture is transfer learning, which allows the model to require less training data to perform well compared to traditional neural network methods. Multiple improved models based on BERT have been developed in previous research that focus on improving either the prediction metrics or reducing the computational requirements of BERT. RoBERTa[36] and XLNet[37] were chosen based on their improvements in performance compared to BERT, while ALBERT[38] was chosen both based on both its performance increase and its parameter reduction techniques.

The BERT-based models were implemented through the Simple Transformers library, which provides a `ClassificationModel`[39] that can be customized to each BERT flavor. The Simple Transformers library is built on top of the powerful Transformers library by Hugging Face and offers a simple integration for implementing various BERT-based models. The `ClassificationModel` provides built-in methods for model fine-tuning, evaluation and prediction. The data input to the models was given as raw text as described in 4.8.2. Each of the BERT model flavors are fine-tuned over a pre-trained model, which for ALBERT was *albert-base-v2*, for XLNet *xlnet-base-cased* and for RoBERTa was *roberta-base*. The specifications of these pre-trained models can be found in the Hugging Face documentation[40].

The hyperparameter tuning of the BERT-based models was performed by Bayesian Optimization with the Weights & Biases (W&B) framework. The framework offers a web-based dashboard interface that allows tracking and visualization of each run with different hyperparameters and saves the evaluation scores obtained for each run. The hyperparameters chosen for tuning were batch size, learning rate, and epochs, as these hyperparameters are the ones used for tuning in the official BERT paper (Devlin et al., 2018).

[36] `https://huggingface.co/transformers/model_doc/roberta.html`, Last accessed: 14.06.2021

[37] `https://huggingface.co/transformers/model_doc/xlnet.html`, Last accessed: 14.06.2021

[38] `https://huggingface.co/transformers/model_doc/albert.html`, Last accessed: 14.06.2021

[39] `https://simpletransformers.ai/docs/classification-models/`, Last accessed: 16.6.2021

[40] `https://huggingface.co/models`, Last accessed: 14.06.2021

4.10 Evaluation Metrics

The evaluation metrics listed in Section 2.6, namely accuracy, F1, and coverage were used to evaluate the parts of the system.

For the weak labeling system, the goal is for the system to be equally good at generating the correct label for both fake and real instances. Furthermore, it is desirable that the system assigns a label to most instances, i.e., have high coverage. Therefore, when choosing the best heuristics for the weak labeling system, the system's accuracy was the best metric to measure its performance, combined with the coverage.

For the end models, the main focus was to predict whether an article is fake, so the F1 score was most important when measuring their performance at correctly predicting the fake labels. However, as the test set is balanced, the accuracy is also a suitable metric. The F1 score was therefore considered in combination with accuracy for the end models.

As the datasets used for training and testing are balanced, a zero rule baseline that chooses the most frequent class in the training data will likely achieve an accuracy and an F1 score of 0.5. This is similar to a random guess model in this case, which is considered to not learn from the data. When evaluating the weak labeling systems and the end models they should thus perform at least as good as a zero rule model.

Chapter 5

Experiments

This chapter will describe the experiments conducted for evaluating the weak labeling systems and the weakly supervised and supervised end models. As stated in Section 1.3, the objective of this work is two-fold: First, to develop a weak labeling system that efficiently labels instances. Second, to create a machine learning end model trained on the weakly labeled data outputted from the weak labeling system. For evaluating the weak labeling systems and end models, one experiment was conducted for each objective, resulting in two main experiments:

- **Experiment 1:** Compare the weak labeling systems implemented in this work with each other as well as the baseline manual weak labeling system proposed in From and Netland (2020).
- **Experiment 2:** Compare the performance of classifiers for the weakly supervised learning scenario, and evaluate the effect of adding the weakly labeled instances to the training data in comparison to a purely supervised approach. This was done for two different scenarios, first with a realistic, limited amount of labeled data and later with an extensive amount of labeled data.

5.1 **Experiment 1:** Weak Labeling Systems

Experiment 1 is related to the first research question (RQ1), and studies which content-based weak labeling system performs best at labeling fake news articles. Experiment 1 consisted of two parts, one for each weak labeling system proposed in this work, namely the automatic Snorkel and automatic Snuba systems. The overall pipeline of the experiments conducted to evaluate the weak labeling systems is shown in Figure 5.1. The first and final steps of the pipeline were identical for both experiments, while the weak labeling system itself (marked in blue) was interchangeable. The pipeline starts with preprocessing the data using the techniques described in Section 4.4 in order to generate numerical features as described in Section 4.5. The data was then split into a training and a test set, as

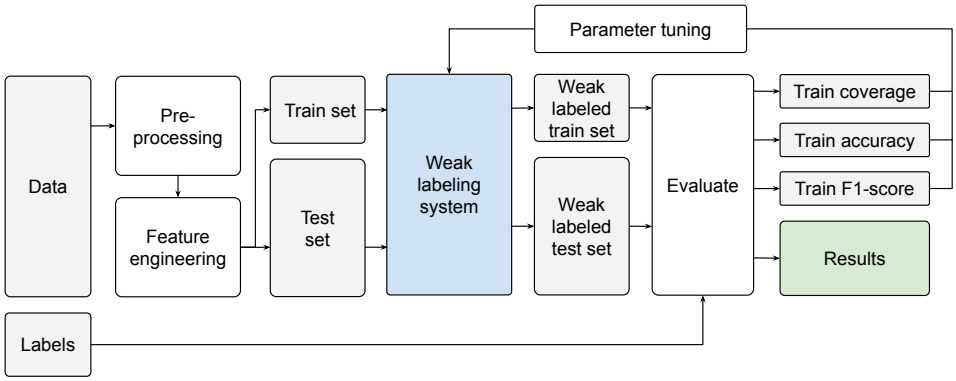

Figure 5.1: Pipeline of the experiments conducted to test and compare the weak labeling systems proposed in this work. The weak labeling system component, marked in blue, is interchangeable.

will be further described in Section 5.1.1. Heuristics were then created by the weak labeling system based on the training set. After, the heuristics were applied to both the training set and the test set, providing weak labels to the data. The weak labels of the training set were then compared to the ground truth labels to find the coverage, accuracy and F1 score of the system given the set of hyperparameters used. The metrics were used to tune the hyperparameters further. The best weak labeling system was selected by evaluating the performance of the weak labels on the test set. Note that this test set is not the same as the *Manually labeled test set* that will be used for the end models. Therefore, selecting a model based on this test set does not cause data leakage to the end model. Section 5.1.2 and 5.1.3 will further describe the specifics of the experiments done to evaluate each of the two weak labeling systems.

5.1.1 Dataset Splitting

The balanced dataset from NELA-GT-2019 with 252,006 articles, described in Section 4.3.1 was used in these experiments. According to Varma and Ré (2018), the Snuba system performs best when the unlabeled set is four times as large as the labeled set. This was assumed to be a good ratio for the Snorkel system as well, thus 20% of the instances were selected for the labeled set and the remaining 80% for the unlabeled set. In these experiments, the labeled set corresponds to the training set, and the unlabeled set corresponds to the test set. The number of samples in the training and test set is shown in Table 5.1, and both datasets are balanced by class.

Table 5.1: Partitions of the datasets used in Experiment 1.

	# of samples	**Origin dataset**
Training set	50 402	NELA-19
Test set	201 604	NELA-19

5.1.2 Automatic Weak Labeling System with Snorkel

For the automatic weak labeling system in Snorkel, the thresholds for all the features in the data were first found during an automatic threshold search. Which thresholds were found depends on the value of the relative difference limit (RDL), and different values for the RDL were tested and evaluated on the training set. Twenty different values were tested, where the first ten values ranged from 0.01 to 0.09 with 0.01 increments, and the last ten values ranged from 0.1 to 0.9 with 0.1 increments. Additionally, the performance of the system is dependent on which labeling functions (LFs) were selected for aggregating the labels. Thus the three types of LF sets (*All, Accuracy* $> 65\%$ and *Top 25*) were evaluated for each RDL value, resulting in a total of 60 experiments. Finally, the labels were aggregated using both the majority vote (MV) and the generative model (GM) for all 60 experiments. The best system for each of the three LF sets were compared. Then, based on the accuracy achieved on the test set, the best system was selected for comparison to the automatic weak labeling system in Snuba, which is explained in Section 5.1.3.

5.1.3 Automatic Weak Labeling System with Snuba

The Automatic Snuba system was tested using three types of heuristics, namely Decision Trees, Logistic Regression and k-Nearest Neighbor. For all the heuristics, an experiment was conducted using the max cardinality values 1, 2 and 3, which indicates the number of features the system should select for the subset evaluated at each iteration. This resulted in a total of 9 experiments. The best system was chosen based on the accuracy achieved on the test set. However, the coverage also played a role. To ensure consistency when comparing the best Snuba and Snorkel systems, it was decided that the chosen Snuba system should have a coverage at least as high as the best weak labeling system in Snorkel.

5.1.4 Comparison of Weak Labeling Systems

The best-performing settings for the Automatic Snorkel and the Automatic Snuba weak labeling systems were selected and compared to the *Manual weak labeling system*. The best of the three weak labeling systems were then chosen as the weak labeling system used further in Experiment 2.

5.2 Experiment 2: End Models

The second experiment consists of three sub-experiments. Experiment 2.A was related to RQ2 and studied which weakly supervised end model performs best for detecting fake news. Experiment 2.B was related to RQ3 and evaluated the effect of expanding the labeled training set with weakly labeled data when training a classifier in comparison to a purely supervised approach. Lastly, a third experiment, Experiment 2.C, was conducted to test the impact of the data size for training the models when comparing to the supervised approach. Additionally, preliminary experiments were carried out in order to conduct the main experiments, and are explained in Section 5.2.1. Experiment 2.A, 2.B and 2.C are further explained in Sections 5.2.2, 5.2.3 and 5.2.4, respectively.

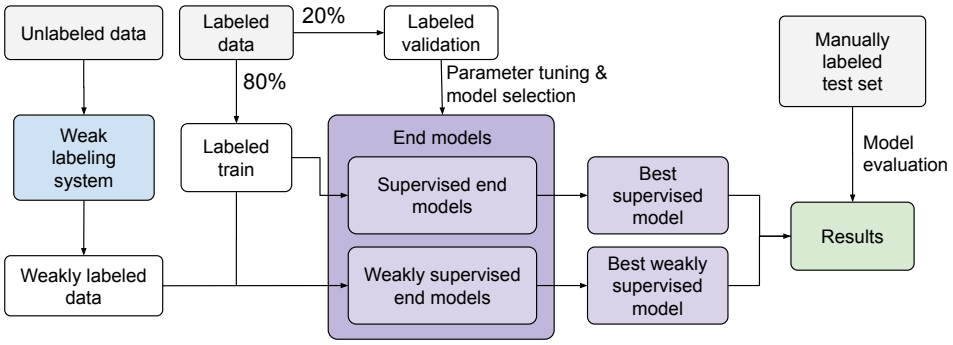

Figure 5.2: A pipeline of Experiment 2. The labeled data was first split into a training and validation set. Then, the weak labeling system, marked in blue, was used to apply weak labels to the unlabeled data. As input to the supervised end models was only the labeled training set, and as input to the weakly supervised end models were both the labeled training set and the weakly labeled data. All end models were tuned, and model selection was performed based on the labeled validation set. Lastly, the best supervised and weakly supervised models were compared on the *Manually labeled test set*.

The overall pipeline of Experiment 2 is shown in Figure 5.2. First, the labeled data was split into a training and validation set, with a ratio of 80% to 20%, respectively. Then, the weak labeling system chosen in Experiment 1 was pre-trained on the labeled training data, and weak labels were applied by the weak labeling system to the unlabeled data. The input to the supervised end models was the labeled training set, and the input to the weakly supervised end models were both the labeled training set and the weakly labeled data. Both the supervised and weakly supervised end models' hyperparameters were tuned with the values described in Section 5.2.2. Model selection was performed based on the labeled validation set. Lastly, the best supervised and weakly supervised models were compared on the *Manually labeled test set*. The pipeline is the basis for all experiments in this section.

5.2.1 Preliminary Experiments

The preliminary experiments included the hyperparameter tuning of the end models, model selection, and evaluating the best weak labeling system on the test set. The hyperparameter tuning is further explained for each experiment in Section 5.2.2 and 5.2.4. Model selection was conducted to compare the best-performing end model for the weakly supervised and the supervised learning scenario in Experiment 2.B and was selected based on the models' performance on the validation set. By selecting based on the validation set, it was ensured that the comparison of the two learning scenarios was evaluated on unseen data in the *Manually labeled test set*. In order to assess whether the end models actually generalized beyond the weak labeling system, weak labeling system evaluated on the test set was used as a baseline for the weakly supervised end models.

5.2.2 **Experiment 2.A:** Evaluation of End Models

Experiment 2.A follows the pipeline presented in Section 5.2, but without the comparison between the supervised and weakly supervised models. First, the data splits used are presented in the section below. Second, the model training and hyperparameter tuning are described in the following section.

Dataset Splitting

The data splits used in Experiment 2.A are shown in Table 5.2, and this section will explain how the splits were created. First, the balanced NELA-GT-2019 subset was split into a labeled and unlabeled set. The goal was to simulate a realistic setting where there is a limited availability of labeled data and more unlabeled data available. To provide this scenario, the number of labeled instances in the training data for Experiment 2.A was chosen to be the size of the largest manually labeled dataset containing full articles presented in Section 3.3, which contained 1,380 instances. The idea behind selecting this number of instances as the training data size was that a number of 1,380 instances is realistically the amount of manually labeled data available to train a supervised classifier. The *Labeled training set* was thus set to consist of 1,380 instances, and a validation set was chosen to be the size of 345 instances resulting in a size ratio of 80% to 20% for the training and validation set, respectively. The same validation set was used for both supervised and weakly supervised end models. Finally, the *Manually labeled test set*, as presented in Section 4.3.2, was used for evaluating the performance of the end models.

For the weak supervision scenario, the *Weakly labeled training set* was created using the best-performing weak labeling system from Experiment 1. To follow the idea of limited access to labeled data, weak labels were generated for the *Unlabeled training set* in Table 5.2 based on heuristics extracted from the 1,380 labeled instances available in these experiments. The *Weakly labeled training set* was chosen to be four times the size of the *Labeled training set*, resulting in a size of 5,520 instances, 2,670 of each class. Only the 2,670 instances with the most confident probability labels of each class were selected from the weakly labeled *Unlabeled training set* to extract the most high-quality labels.

Table 5.2: Partitions of the datasets used in Experiment 2.A and 2.B.

	# of samples	**Origin dataset**
Labeled training set	1 380	NELA-19
Unlabeled training set	201 604	NELA-19
Weakly labeled training set	5 520	NELA-19
Labeled validation set	345	NELA-19
Manually labeled test set	434	Manual

Model Training and Hyperparameter Tuning

The five classifiers used as end models, namely the Logistic Regression (LR), XGBoost, ALBERT, XLNet and RoBERTa models, were trained using both supervised and weakly

supervised learning. For the supervised scenario, the *Labeled training set* of 1,380 samples was used for training the models. As the *Labeled training set* was used for obtaining the weak labels, the ground truth labels used to extract the heuristics will also be available at the time of training. The datasets were therefore combined to measure the potential benefit of expanding the training data by adding weakly labeled data. Thus, for the weakly supervised scenario, the *Labeled training set* and the *Weakly labeled training set* were combined and used to train the classifiers, which consisted of 6,900 instances in total.

The hyperparameters of each model and learning scenario were tuned individually. The Logistic Regression and XGBoost models were tuned using GridSearchCV. Due to the default implementation of the grid search, the best parameters were evaluated and chosen based on the training set. A Bayesian optimization tuning strategy was used for the BERT-based models, where the validation set was used to choose the best hyperparameters. Ideally, the parameters for the LR and XGBoost models should also have been chosen based on the validation set to ensure consistency between all the models. However, due to the time limit, this was not implemented. The input and hyperparameter specifications for each of the models are described below.

Logistic Regression The LR classifiers take the Term Frequency-Inverse Document Frequency (TF-IDF) vectors as input. The models were tuned by altering the value for the regularization hyperparameter `C`, with values shown in Table 5.4. The other model parameters are kept constant for all LR models, and the hyperparameters that were set to a different value than its default are shown in Table B.1 in Appendix B for reproducibility purposes. The constant hyperparameters included the maximum iterations allowed for the solver to converge, the solver algorithm used, penalization norm for regularization, and the number of cross-validation folds.

Table 5.4: The value ranges tested of the hyperparameters tuned for the Logistic Regression models.

Parameter	Value Range
`C`	[0.001, 0.009, 0.01, 0.09, 1, 5, 10, 25]

XGBoost The XGBoost classifiers use the same input as Logistic Regression, namely the TF-IDF vectors. The XGBoost classifiers were tuned by testing the value ranges for the parameters as shown in Table 5.6. The other model parameters are kept constant for all XGBoost models, and the hyperparameter that was set to a different value than its default is shown in Table B.1 in Appendix B for reproducibility purposes. The constant hyperparameter was the parameter determining the number of boosting rounds.

Table 5.6: The value ranges tested of the hyperparameters tuned for the XGBoost models.

Parameter	Value range
max_depth	3 - 12
learning_rate	[0.01, 0.1, 0.3]
colsample_bytree	[0.3, 0.8, 1.0]
subsample	[0.8, 1.0]
gamma	[0, 1, 5]

ALBERT, XLNet, RoBERTa

The three BERT-based models uses raw text as input only minimally pre-processed, as described in Section 4.8.2. The hyperparameters of the three models were tuned by adjusting the number of epochs, the learning rate and batch size, and the value ranges for the tuned parameters are shown in Table 5.8. The other model parameters are kept constant for all the BERT-based models, and the hyperparameters that were set to a different value than its default are shown in Table B.1 in Appendix B for reproducibility purposes. The constant hyperparameters included the maximum sequence length and gradient accumulation step.

Table 5.8: The value ranges tested of the hyperparameters tuned for ALBERT, XLNet and RoBERTa.

Parameter	Value range
num_train_epochs	1 - 10
learning_rate	0 - 4×10^{-4}
train_batch_size	[16, 32]

5.2.3 **Experiment 2.B:** Comparison of Weakly Supervised and Supervised Learning

For Experiment 2.B, the best-performing classifier of each learning scenario in Experiment 2.A was chosen based on their performance on the validation set, by calculating the metrics specified in Section 4.10. The best classifiers were then compared based on their performance on the *Manually labeled test set* shown in Table 5.2.

As described in Section 5.2.1, to assess whether the end models generalized beyond the weak labeling system, the *Manually labeled test set* was also weakly labeled by the best weak labeling system to provide a baseline for comparison. The idea is that the end models should perform at least as good as the weak labeling system, presuming that if they do not offer an improvement, the weak labeling system could be used directly to detect fake news.

5.2.4 **Experiment 2.C:** Evaluation of Data Size and Weak Label Ratio

Experiment 2.C is also based on the pipeline shown in Section 5.2 and was conducted by training the models with considerably more data samples than for Experiment 2.A. This was done to study how the size of the datasets would affect the performance of the weakly supervised models compared to their supervised equivalents. Furthermore, to study how the ratio of weak labels to ground truth labels impacts the result, three versions of the weakly labeled dataset with altering ratios of labeled data to weakly labeled data were created. The dataset splits used in this experiment are shown in Table 5.9, and the reasoning for the splits are described further below.

The experiment was conducted by first training the best model for the supervised learning scenario found in Experiment 2.A on the *Labeled training set* from Table 5.9 with 50 204 instances. Next, the three versions of the weakly labeled dataset with different label ratios combined with the *Labeled training set* were used to train the best weakly supervised classifier found in Experiment 2.A. The resulting models were then evaluated and compared based on their performance on the *Manually labeled test set*.

Dataset Splitting

As the purpose of Experiment 2.C is to test the effect of training the models with large amounts of data, all the instances of the full balanced dataset described in Section 4.3.1 were used. The basis for this experiment was thus the same data splits as were used in Experiment 1, which included all the instances available and consisted of a training set and a test set as shown in Table 5.1. The training set included 50,402 instances and was set as the *Labeled training set* in this experiment. The *Labeled training set* was used to train the best weak labeling system found in Experiment 1, namely the automatic Snuba system, which was used to assign weak labels to the instances in the test set, consisting of 201,604 instances. This resulted in a weakly labeled set with 170,575 instances posterior to filtering out abstained instances and is referred to as the *Weakly labeled training set all* in Table 5.9.

The datasets used for testing the impact of label ratio were created by reducing the size of *Weakly labeled training set all*. Two ratios were considered, one with a ratio of 1:1 and 2:1 for the ground truth labels to weak labels, respectively, which resulted in the *Weakly labeled training set 50k* and *Weakly labeled training set 25k*. The *Weakly labeled training set 50k* set contains the 25,201 most confident labeled articles measured by the probability label of each class, resulting in a total of 50,402 articles. The *Weakly labeled training set 25k* set consists of the 12,600 most confident labeled articles from each class, 25,200 articles in total. The *Labeled validation set* was set to a size of 12,600 instances so that the *Labeled training set* and the *Labeled validation set* had a ratio of instances of 80% to 20%.

Model Training and Hyperparameter Tuning

For both learning scenarios in experiment 2.A, the best classifier turned out to be the RoBERTa classifier. The RoBERTa classifier required extensive computing power to perform hyperparameter tuning on the full training set of this size. The model was tuned using

Table 5.9: Partition of dataset for Experiment 2.C

	# of samples	Origin dataset
Labeled training set	50,402	NELA-19
Weakly labeled training set all	170,575	NELA-19
Weakly labeled training set 50k	50,402	NELA-19
Weakly labeled training set 25k	25,200	NELA-19
Labeled validation set	12,600	NELA-19
Manually labeled test set	434	Manual

30% of the training data for each learning scenario, which was assumed to be sufficient to represent the whole set. The same hyperparameters as in the previous experiments were tuned, namely learning rate, batch size, and number of epochs. The best parameters were selected by evaluation on the *Labeled validation set*.

Similar to how the models were trained in Experiment 2.A and 2.B, after hyperparameter tuning, the supervised classifier was trained on the full *Labeled training set* using the best parameters found. Next, three weakly supervised models were trained on the full *Labeled training set* in combination with one of the three *Weakly labeled training set* sets, resulting in three different weakly supervised models. Lastly, the models were evaluated on the *Manually labeled test set*.

5.3 Code

For these experiments, the computations were performed on the NTNU IDUN computing cluster (Själander et al., 2019). The cluster has more than 70 nodes and 90 GPGPUs. Each node contains two Intel Xeon cores, at least 128 GB of main memory, and is connected to an Infiniband network. In addition, half of the nodes are equipped with two or more Nvidia Tesla P100 or V100 GPGPUs. The storage is provided by two storage arrays and a Lustre parallel distributed file system.

In order to replicate the experiments, the following resources provide the code and datasets used in this work:

- The code for the text preprocessing, numerical feature extraction and weak labeling systems: `https://github.com/piiingz/fake-news-detection-weak-labeling`
- The code for implementation, training and hyperparameter tuning of the end models: `https://github.com/piiingz/fake-news-detection-classifiers`
- The manually fact-checked test set: `https://github.com/piiingz/fake-news-detection-test-set`
- The NELA-GT-2019 dataset used for training and validation: `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/O7FWPO`

Chapter 6

Results and Discussion

This chapter presents the results of the experiments described in Chapter 5. The results will follow the same structure as Chapter 5, with a discussion of the results included in each section. Experiment 1 is related to the first research question (RQ1), while Experiment 2 is related to the second (RQ2) and third research question (RQ3). Finally, a general discussion of the system as a whole with possible improvements follows.

6.1 **Experiment 1:** Weak Labeling Systems

The results and discussion of Experiment 1 consist of three parts, first the Automatic Weak Labeling System in Snorkel. Second, the Automatic Weak Labeling System in Snuba, and last, a comparison of the best performance of each system.

6.1.1 Automatic Weak Labeling System with Snorkel

Results

The best results for all the experiments run for each labeling function (LF) set on the automatic system in Snorkel are shown in Table 6.1, along with the relative difference limit (RDL) and number of LFs. The best RDL value for the *All* set was 0.2, resulting in a total of 33 LFs. For the *Acc >65%* set the best RDL value was 0.5 which included 6 LFs and for the *Top 25* set the RDL value was 0.01 and included 25 LFs. Regarding accuracy and F1 score, the aggregation of LFs by the majority vote (MV) performed better than the generative model (GM) for all the sets. In contrast, the GM produced higher coverage of instances for all the sets. As described in Section 4.10, the best weak labeling system was chosen based on accuracy as it is beneficial for the training data to provide both correct real and correct fake supervision signals. The weak labeling system based on the LF set *Acc >65%* achieved the best accuracy and was therefore chosen as the best Snorkel-based system.

Table 6.1: The best results achieved for each set of labeling functions using the automatic labeling function approach in Snorkel.

			Generative model			**Majority vote**		
LFs	**RDL**	**#LFs**	Acc	F1	Cov	Acc	F1	Cov
All	0.2	33	0.656	0.665	0.999	0.665	0.720	0.903
Acc >65%	**0.5**	**6**	0.665	0.684	0.922	**0.710**	0.740	0.860
Top 25	0.01	25	0.653	0.651	0.999	0.679	0.713	0.922

Discussion

The RDL values chosen for the LF sets differed greatly, as did the number of LFs included in the sets. As explained in Section 4.6.2, the RDL value dictates how strict the system should be when discriminating the classes, with a higher value resulting in a stricter system that achieves better accuracy but covers less of the instances. It makes sense that when all LFs created are included in the system, some of them will introduce noisy labels and the system compensates for this by employing a strict RDL value to achieve a good performance, as seen for the *All* set with an RDL value of 0.2. The *Acc >65%* had an even stricter value for the RDL. An individual performance of 65% is relatively high for an LF, and to create LFs that match this limitation, a strict RDL value is needed for the LFs to solely cover instances they are confident of. That the *Acc >65%* set excelled in terms of accuracy was expected as a combination of accurate LFs will result in a good performance. In terms of coverage, the *Acc >65%* set achieved the lowest result, which was expected as high accuracy leads to lower coverage.

For the *Top 25* set, the best accuracy was achieved by the use of the lowest value for the RDL, namely 0.01, which was an unexpected result. It was observed that a lower RDL value produced more LFs as more low-quality LFs are introduced. As the *All* set included only 33 of the 136 LFs that could potentially be created, a total of 25 LFs is a relatively high number of LFs. Therefore, it makes sense that a less strict RDL value was necessary to create enough LFs to choose the top 25. In retrospect, choosing the top 25 LFs might be an insufficient condition for creating an LF set as this increases the chances of including low-quality LFs. When it comes to the number of LFs included in the other sets, it was expected that the *All* set included the most LFs, and the *Acc >65%* included the fewest. However, it was surprising that only 6 LFs passed the bar of having an accuracy above 65%. This is probably due to the strict RDL value coupled with a further restriction on the accuracy.

It was interesting to study which features the 6 LFs of the best-performing system were based on, and the 6 labeling functions with their respective individual accuracies and coverages are therefore shown in Table 6.2. Three of the features were related to the content of the articles, and three were related to the title. The content-related LFs were based on the exclamation count and ratio features and the number of words per sentence. For the title-related LFs, all of them were based on features regarding proper noun count and ratio. These types of features were all listed as suitable separators by Horne and Adali (2017) in Section 3.1 and were expected to create good LFs. Still, it is notable that both

Table 6.2: The LFs that made up the best Snorkel-based automatic weak labeling system with their respective individual accuracies.

Labeling function	Accuracy	Coverage
lf_content_words_per_sentence_upper	0.741	0.090
lf_title_proper_nouns_count_upper	0.716	0.371
lf_content_exclamation_count_upper	0.693	0.153
lf_content_exclamation_ratio_upper	0.693	0.153
lf_title_proper_nouns_ratio_upper	0.680	0.465
lf_title_proper_nouns_count_lower	0.665	0.429

the count and the ratio of the exclamation marks and proper nouns were included as the best features, as these features capture much of the same information. From their identical accuracies and coverages, the LFs related to the exclamation count and ratio features of the content seem identical regarding which labels they assign. If this assumption is true, the outcome may be that a single feature's value has twice the impact on the result when the labels are aggregated by a majority vote as in this case.

By choosing a weak labeling system based only on these six relatively simple features, it could be possible for creators of fake news content to counteract the detection system by adjusting only a few of its characteristics. For preventing the reliance on only a few features for assigning labels, an improvement could be to include more thresholds for each feature to generate more LFs. Additionally, LFs should be selected for an LF set based on more aspects than simply their accuracies.

An improvement for the automatic Snorkel system could be to tune the accuracy limit of the *Accuracy >65%*, and the number of LFs included in the *Top 25* set as this directly impacts the aggregation of the labels and will have a significant impact on the result. Another improvement would be to extract additional numerical features from the data, which would augment the number of LFs in the system, and potentially increase the system's performance. For instance, the Linguistic Inquiry and Word Count (LIWC) framework (Pennebaker et al., 2015) could be used to extract more advanced numerical features based on linguistic, psychological and topical categories of the texts.

6.1.2 Automatic Weak Labeling System with Snuba

Results

The results for the experiments run on the weak labeling system in Snuba are shown in Table 6.3. Due to memory limitations, the experiment using k-Nearest Neighbor (k-NN) as the heuristic type with a max cardinality of 3 could not be run, and no result was achieved for this experiment. The best system was chosen based on the accuracy and coverage achieved on the validation set. The chosen Snuba system should have a coverage at least as high as the best weak labeling system in Snorkel, which achieved a coverage of 0.86. Four systems met this requirement, and of these four, the system that achieved the best accuracy was chosen as the best-performing Snuba system. The selected system was the one based on decision trees as the heuristic type with a max cardinality of 3, which

Table 6.3: The results achieved for the experiments done using the weak labeling systems in Snuba.

Model	Max cardinality	Accuracy	F1 score	Coverage
	1	0.836	0.911	0.077
DT	2	0.753	0.769	0.873
	3	**0.765**	0.765	0.902
	1	0.774	0.845	0.221
LR	2	0.766	0.816	0.384
	3	0.760	0.777	0.551
	1	0.610	0.483	1.0
k-NN	2	0.650	0.584	1.0
	3	N/A	N/A	N/A

achieved an accuracy of 0.765.

Discussion

When using decision trees (DT) as heuristics, an increase in the max cardinality from 1 hurt the system's performance, especially for the F1 score. However, DT with max cardinality of 1 achieved a coverage of only 0.077, which was a surprising result. The low coverage is most likely due to using decision trees based on a single feature is the same as checking whether the feature is above or below a thershold. As explained in Section 2.4.2, for the DT heuristics, Snuba calculates the confidence of the labels based on the number of labeled instances in the leaf nodes. Checking the value of each feature will result in a large number of heuristics where each will include few labeled instances in their leaf nodes. Thus, the labeled instances are thinly spread across the leaf nodes, causing the system to assign mostly low-confidence labels and thus abstain from labeling numerous instances. The system's coverage plays an important role, and the DT approach with max cardinality of 1 is concluded to be an inadequate approach for classifying fake news. When the subsets for generating heuristics are increased to include more features, the DT approach drastically increases the coverage. However, the decrease in accuracy indicates that more noisy labels are introduced.

For the Logistic Regression (LR) heuristics, the coverage was considered inadequate. As explained in Section 2.4.2, in the LR case, the confidence of the labels is measured by a sigmoid function where its parameters are learned from the labeled instances. Using the sigmoid function results in the instances close to the decision boundary being assigned labels with confidences close to 0.5 and thus be of low confidence. As the distributions of the real and fake articles overlap significantly for multiple feature combinations, many of the instances were likely close to the decision boundary, resulting in a low coverage for all the max cardinality values set.

It was observed that the systems using k-NN as the heuristic type performed poorly in both experiments, and would most probably not surpass the best system for a max cardinality of 3, either. Additionally, the k-NN approach achieved a coverage of 1.0 for all experiments, which was an interesting result. The high coverage achieved is most likely

due to the confidence of the labels being calculated as a function of distance from the labeled data points. For many subsets of the features, the distributions of the classes are, as mentioned, very similar, and the clusters of real and fake articles will therefore overlap. This causes an unlabeled instance to most likely be within a short distance of a labeled instance for all the heuristics. As all the instances were covered, it was no surprise that the k-NN approach performed poorly for the same reasons as discussed in Section 6.1.1, namely that a stricter system with lower coverage performs better in terms of accuracy. If the number of features used for each heuristic was increased, e.g., a higher max cardinality, the decision boundaries would be of a higher dimension and possibly separate the classes better. Unfortunately, due to limited computing resources, this was not tested.

An improvement for the Snuba system would be to analyze which features were used as the basis for the heuristics in the committed set and the size of the committed sets. This analysis would give valuable insight into figuring out the system's behavior and how to improve the heuristics. However, doing so was not supported by the framework itself, and due to time limits, this was not implemented. Additionally, following the arguments for improvement of the automatic Snorkel system, augmenting the data with additional numerical features could improve the Snuba system, as well.

6.1.3 Comparison of Weak Labeling Systems

Results

The results for the best automatic weak labeling system in Snorkel and Snuba are repeated in Table 6.4. The results for the manual weak labeling system are also shown as they provide a baseline for evaluating the weak labeling systems proposed in this work. It was shown in the previous work in From and Netland (2020) that the manual weak labeling system outperformed a zero rule based approach. Thus the manual weak labeling system is an adequate baseline for the automatic approaches. Both the automatic systems in Snorkel and Snuba performed better than the manual weak labeling system in terms of accuracy and F1 score. The automatic Snorkel system covered the same ratio of instances as the manual, whereas the Snuba system offered an improvement in coverage of 4.2 percentage points, later referred to as *points*. The best system was again chosen based on its accuracy due to the reasons described in Section 4.10. Thus of all the systems, the automatic weak labeling system in Snuba was considered to be the best, with an accuracy of 0.765 on the test set.

Table 6.4: The results of the best-performing systems within each type of weak labeling system.

	Accuracy	**F1 score**	**Coverage**
Manual system in Snorkel (baseline)	0.700	0.720	0.860
Snorkel, Acc >65%	0.710	0.740	0.860
Snuba, DT, 3	**0.765**	**0.765**	**0.902**

Discussion

That the automatic approaches were better than the manual one was expected as the automatic systems' parameters could be tuned to increase their performance on the training set. Additionally, it was expected that Snuba performed better than the Snorkel system due to the heuristics of the Snuba system, in general, being more complex than those in the Snorkel system. The labeling functions developed in Snorkel simply checked for a condition of a single feature, whereas the Snuba system could find more complex decision boundaries by aggregating several features. An exception is the Snuba experiments with max cardinality set to 1, where the heuristics were based on a single feature. However, as seen in Table 6.3 basing the heuristics on only one feature resulted in a poor performance in either coverage or accuracy for all heuristic types in Snuba.

The Snuba system achieved better coverage than the Snorkel system. This was also expected as Snuba takes the heuristic's diversity into account when selecting which heuristics to include in the committed set. Thus there will intentionally be less overlap between the coverage sets of each heuristic in this system.

6.2 Experiment 2: End Models

This section presents the results and discussion of Experiment 2 and consists of four parts. Section 6.2.1 presents the results from the preliminary hyperparameter tuning, model selection based on the validation set, and the weak labeling system baseline evaluated on the *Manually labeled test set*. Section 6.2.2 evaluates the performance of the end models for each learning scenario, namely Experiment 2.A. Section 6.2.3 compares the best-performing weakly supervised and supervised end model, namely Experiment 2.B. Finally, Section 6.2.4 presents and discusses the results of expanding the size of the data in Experiment 2.C.

6.2.1 Preliminary Experiments

Parameter Tuning

The increase in performance from hyperparameter tuning is shown in Table 6.5 and Table 6.6 for the weakly supervised and supervised models, respectively. The models that obtained a better performance after tuning are marked in bold.

For the weakly supervised end models, the XGBoost model's metrics performed best with the default parameters, thus the performance metrics remained unchanged for the tuned version. The LR model had a slight increase of 3 percentage points for both accuracy and F1 score after tuning. For the BERT-based models, a distinct improvement can be seen after tuning. The best improvement was seen in the ALBERT model, increased by 9.3 points for accuracy and 10.3 points for F1.

For the supervised end models, both the LR and XGBoost models performed better with the default parameters. Like the weakly supervised models, ALBERT was the model with the most significant overall improvement, with increased accuracy of 14 points and an increased F1 score of 13 points after tuning.

Table 6.5: Weakly Supervised end models before and after hyperparameter tuning, evaluated on the validation set. The values marked in bold denote the metrics that obtained an increased performance after hyperparameter tuning.

	Default		**Tuned**	
	Accuracy	F1 score	Accuracy	F1 score
Logistic Regression	0.791	0.786	**0.794**	**0.789**
XGBoost	0.748	0.731	0.748	0.731
ALBERT	0.736	0.713	**0.829**	**0.816**
XLNet	0.817	0.803	**0.884**	**0.870**
RoBERTa	0.806	0.785	**0.890**	**0.873**

Table 6.6: Supervised end models before and after hyperparameter tuning, evaluated on the validation set. The values marked in bold denote the metrics that obtained an increased performance after hyperparameter tuning.

	Default		**Tuned**	
	Accuracy	F1 score	Accuracy	F1 score
Logistic Regression	0.728	0.724	0.728	0.724
XGBoost	0.759	0.759	0.759	0.759
ALBERT	0.701	0.700	**0.841**	**0.830**
XLNet	0.790	0.790	**0.855**	**0.838**
RoBERTa	0.812	0.804	**0.913**	**0.905**

As mentioned in Section 5.2.2, the LR and XGBoost models were tuned by validating the model on the training set for implementation reasons, so there was no guarantee that the tuned parameters would perform better on the validation set. The LR and XGBoost models were still evaluated on the validation set to ensure a fair comparison between the models. Therefore, if the LR or XGBoost model performed better with the default parameters on the validation set, these parameters were chosen. The selected best hyperparameters are shown in Table B.2 in Appendix B for all the models.

End Model Selection

In order to compare the best-performing end model for the weakly supervised and the supervised learning scenario in Experiment 2.B, the best model for each scenario was selected based on the models' performance on the validation set. From the results presented in Table 6.5 and Table 6.6, RoBERTa was selected as the best model for both scenarios.

Weak labeling system baseline

In order to assess whether the end models have generalized beyond the weak labeling system, the *Manually labeled test set* was weakly labeled by the best weak labeling system

Table 6.7: The results from evaluating the best weak labeling system, namely the automatic Snuba system, on the *Manually labeled test set*.

	Accuracy	**F1 score**	**Coverage**
Snuba, DT, 3	0.646	0.668	0.956

to provide a baseline for comparison. The Snuba system with decision tree heuristics and cardinality 3 achieved an accuracy of 0.646, an F1 score of 0.668, and a coverage of 0.956 on the test set, as shown in Table 6.7.

6.2.2 **Experiment 2.A:** Evaluation of End Models

Results

This section presents the results of all the end models evaluated on the *Manually labeled test set* to gain insight into the real-life performance of the models. Table 6.8 presents the results of the weakly supervised end models. The table shows that the best-performing model was RoBERTa for both the validation and test set. On the test set, RoBERTa achieved an accuracy of 0.779 and an F1 score of 0.798. XLNet was the second-best model, with 0.733 in accuracy and 0.752 in F1 when evaluated on the test set. ALBERT was the poorest performing of the BERT-based models on both the validation and test set. Furthermore, both LR and XGBoost performed notably worse than the BERT-based models. The LR model achieved an accuracy of 0.641 and an F1 score of 0.653 on the test set.

Table 6.9 presents the results of the supervised end models evaluated on the validation and test sets. The table shows that the best-performing model was RoBERTa, with 0.753 in accuracy and 0.779 in F1 score on the test set. The second-best model was again XLNet, which achieved an accuracy of 0.718 and an F1 score of 0.742 on the test set. The simpler models, LR and XGBoost, performed notably worse than the BERT-based models.

Table 6.8: The weakly supervised end models evaluated on the validation set and the *Manually labeled test set*. The best value for each metric is marked in bold.

	Validation		**Test set**	
	Accuracy	F1 score	Accuracy	F1 score
Logistic Regression	0.794	0.789	0.641	0.653
XGBoost	0.748	0.731	0.618	0.623
ALBERT	0.829	0.816	0.696	0.717
XLNet	0.884	0.870	0.733	0.752
RoBERTa	**0.890**	**0.873**	**0.779**	**0.798**

Table 6.9: The supervised end models evaluated on the validation set and the *Manually labeled test set*. The best value for each hyperparameter is marked in bold.

	Validation		**Test set**	
	Accuracy	F1 score	Accuracy	F1 score
Logistic Regression	0.728	0.724	0.624	0.630
XGBoost	0.759	0.759	0.578	0.592
ALBERT	0.841	0.830	0.696	0.726
XLNet	0.855	0.838	0.719	0.742
RoBERTa	**0.913**	**0.905**	**0.753**	**0.779**

Discussion

First, we evaluate the performance of the weakly supervised models on the test set. We then compare them to the scores obtained by the baseline, namely Snuba's performance on the test set, which was an accuracy of 0.646, an F1 score of 0.668 and a coverage of 0.956 on the *Manually labeled test set*.

At first glance the LR model performed worse than the Snuba system on the test set. A possible explanation for this is that the Snuba system is designed to label articles it can predict with high confidence and will therefore abstain from labeling the most difficult articles. As a result, the types of articles that Snuba has abstained from labeling are not included in the *Weakly labeled training set*. Therefore, the end models have not been trained on such examples and will consequently be more likely to misclassify them, which is probably the reason for the decrease in performance of the LR model compared to Snuba.

However, to make a fair comparison with the LR model covering all the instances, we have to take coverage into account. On the test set, Snuba achieved a coverage of 95.6%, meaning that 19 of the total 434 articles in the test set were not labeled. An interval of what the Snuba system's realistic score would be if it labeled the 19 articles was estimated. Statistically, because the test set is balanced, the probable worst-case scenario is that the system randomly assigns labels to the remaining instances, which corresponds to an accuracy of 50% for the remaining instances and a resulting accuracy of 0.640 for all the instances. The theoretical best-case would be to label all the remaining instances 100% correctly. However, a more realistic best-case is for the weak labeling system to label the articles as correctly as the rest of the data, achieving an accuracy of 0.646. We chose the realistic best-case, which results in the weak labeling system achieving an accuracy within the interval 0.640-0.646 on the entire test set. With this interval in mind, we observe that the accuracy of 0.641 for the LR baseline is within this range. This finding indicates that the LR model's performance is similar to that of the best weak labeling system. A simple end model such as LR may therefore not generalize beyond the weak labels, and the weak labeling system could thus be used directly and achieve the same result. As the goal of the end models is to generalize beyond the weak labels, the LR model serves as a good baseline for measuring whether the other models have achieved this goal.

When comparing the weakly supervised XGBoost model to the LR baseline, a surpris-

ing finding was that the LR model performed better than XGBoost. This finding indicates that XGBoost was not able to generalize well beyond the training data. It was anticipated that XGBoost would discover more complex relationships in the data than the LR model. However, a possible explanation of the difference in performance between LR and XGBoost could be that the TF-IDF vectors' dimensions used in the experiments were more suitable for LR. The maximum number of features in the TF-IDF vectors could have been tuned to test the impact of the vector dimensionality. In addition, both models have potentially overfitted to the training data as the hyperparameter tuning was based on the training set and not the validation set. They may have performed better if other strategies, like cross-validation, were used for tuning.

The BERT-based models outperformed the simpler models LR and XGBoost, with the best-performing BERT-model, RoBERTa, surpassing the LR baseline with 12 points higher accuracy and 11 points higher F1 score on average. When compared to the baseline, RoBERTa appears to have generalized beyond the weak labeling system, indicating that the BERT-based language models are suited for this text classification task. This is perhaps not surprising, as the BERT-based language models utilize transfer learning by pre-training on large text corpora and are considered state-of-the-art classifiers for natural language processing (NLP) tasks. The language models already possess general knowledge about written language and need fewer data points to fine-tune the model to specific classification tasks, making the BERT-based models robust at predicting articles from different time periods and contexts, such as for the test set. In addition, the mechanisms of the BERT-based models are complex, allowing for better interpretations of complex tasks.

Even though the BERT-based models collectively perform better than the simpler models, a notable difference can also be seen between the different BERT flavors. While XLNet and RoBERTa performed similarly on the validation set, RoBERTa surpassed XLNet by a margin of 4.6 points for both accuracy and F1 score on the test set. This increase in performance indicates that RoBERTa is somehow better at adapting the knowledge obtained through training to samples from other time periods and contexts. Additionally, the ALBERT model scored 5-8 points lower than the RoBERTa model on all metrics, which may be caused by ALBERT's lite-weight parameter-reduction techniques that may be more suitable for other, less complex language tasks.

For evaluating the supervised models, the LR model was again set as a baseline by comparing the supervised LR model to a zero rule-based approach. The Snuba baseline was not used for comparison here, as the weak labeling system is not a part of the supervised approach. As the training data was balanced by class, the zero rule model achieves an accuracy of 0.5 on the balanced test set. As seen from Table 6.9, the LR model surpasses the zero rule-based model by 12 points in terms of accuracy on the test set, indicating that the LR model has learnt from the training data and is a suitable baseline for comparison with the other supervised models.

The supervised XGBoost performed better than LR on the validation set, yet worse than LR on the test set. As the model was chosen based on the validation set, the result on the validation set offers the 'best case result' and should not be considered as its final performance, whereas the test set offers the 'most likely result'. However, the difference in performance on the validation and test set indicates that the hyperparameters chosen for XGBoost were not the optimal ones. Like for the weakly supervised models, the metrics

obtained for both LR and XGBoost would have been more reliable if produced by cross-validation estimation.

The performance of the BERT-based models exceeded the simpler models, also in the supervised scenario. RoBERTa scored around 18 points higher on the validation and 13-15 points higher on the test set than the LR baseline, showing that the model has generalized well beyond the training data. When considering the lower training set size compared to the weakly supervised models, the results from the supervised BERT-based models are surprisingly high. This might be because the BERT-based models, especially RoBERTa, can perform very well even with limited data due to transfer learning mechanisms implemented.

Several improvements for the end models are common for both learning scenarios. First, the evaluation strategy used to measure the performance of the models could have been improved. Using multiple rounds of k-fold cross-validation reduces the variability of the results, which is especially useful when ranking model performances in cases where the models perform similarly. This way, the results obtained would be less sensitive to randomness.

The strategy for assessing the final performance of the models with the test set could have been improved. We observe that all the models scored lower on the test set than on the validation set. The average decrease in metrics on the test set was 14 points lower for accuracy and 11 points lower for F1 in the weakly supervised models, and 13 points lower on accuracy and 12 points lower for F1 in the supervised model. As the *Manually labeled test set* is not originally from the same dataset as the training data, a lower score is expected because models have not been trained on data from the same sources and time period. Consequently, different topics could be more frequent in the time period of the test data than the training set's time period. On the one side, the results on the *Manually labeled test set* give a realistic view of the system's performance in a real-life setting with changing contexts. On the other side, the validation set was the only way to test the model on data from the same original dataset as the training data. However, as the validation set was used to select the best hyperparameters, the scores obtained on the validation set are the best case results and are therefore biased. A better approach for testing how the end models performed on data similar to the training data would be to also include a completely unseen test set from the NELA-GT-2019 dataset.

For the LR and XGBoost models, an improvement could be to implement more advanced word embedding techniques than the TF-IDF vectors. In TF-IDF, similarities between words are not represented, so more sophisticated word embedding techniques like Word2cec[1] and GloVe[2] that produce similar vectors for similar words could improve the system. Additionally, using larger embedding dimensions for TF-IDF could have improved the models as higher dimensions keep more information from the texts. Nevertheless, increasing the embedding dimensions can also increase memory requirements for training the models and requires more computational resources. However, as the BERT-based models were the most promising, such experiments may be unnecessary as the simpler models' performance will most likely never exceed those of the BERT-based language models.

[1] `https://radimrehurek.com/gensim/models/word2vec.html`, Last Accessed: 16.06.2021
[2] `https://nlp.stanford.edu/projects/glove/`, Last Accessed: 16.06.2021

Generally, the BERT-based models gave promising results, but some improvements could be made for these models, as well. When considering the variation of performance among the different BERT-based models, there is a possibility that other BERT-based models could perform even better than RoBERTa. A possible improvement could have been to compare more BERT flavors to discover the best one for this task. This also includes testing different pre-trained models for each flavor like using the *roberta-large* model, which has more layers and more trained parameters, instead of *roberta-base*. We bear in mind that doing so would require more computational resources, which can be a bottleneck for some tasks. Furthermore, tuning additional hyperparameters could have been applied by increasing the maximum sequence length in the models and tuning the models with different batch sizes. The sequence length and batch sizes were kept relatively low for these experiments to avoid exceeding memory limitations.

6.2.3 **Experiment 2.B:** Comparison of Weakly Supervised and Supervised Learning

Results

In order to compare the best supervised with the best weakly supervised end model, the best end model from each learning scenario was selected based on the evaluation on the validation set, as described in Section 6.2.1. RoBERTa was chosen for both learning scenarios, and their performance on the *Manually labeled test set* are shown in Table 6.10. A plot of the differences in the F1 score of each model for the weakly supervised and the supervised scenario as evaluated on the validation set and test set are shown in Figure 6.1 and Figure 6.2, respectively. When evaluated on the test set, the weakly supervised model achieved the best performance in terms of accuracy and F1 score.

Table 6.10: Comparison of the best end model within the weakly supervised and supervised learning scenario.

	Accuracy	**F1 score**
Weakly Supervised	**0.779**	**0.798**
Supervised	0.753	0.779

Discussion

Although the weakly supervised model achieved the best performance, it did not perform significantly better than the supervised model. As the *Labeled training set* used to train both models was limited to 1,380 instances, the performance could depend on which subset of the NELA-GT-2019 dataset was chosen as the labeled set. When splitting the data, the first 1,380 instances were selected to be included in the *Labeled training set* from the dataset created in Section 4.3.1. Suppose other instances were selected for the training data. In that case, the topics represented in the selected articles could have been more or less similar to the topics of the articles in the *Manually labeled test set*. This, in turn, could have resulted in the supervised approach achieving the best result. Whether the

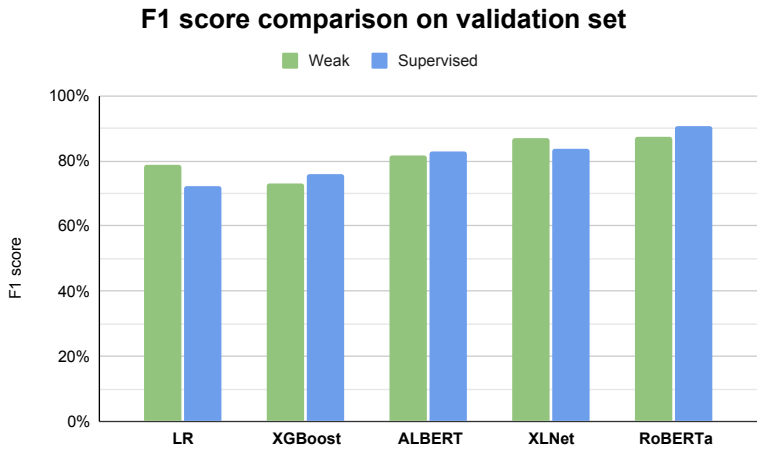

Figure 6.1: Bar plot of F1 score on the validation set for weakly supervised vs. supervised.

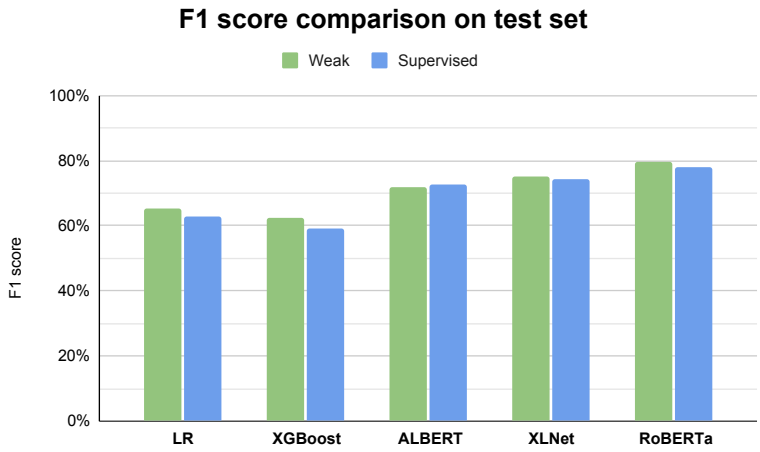

Figure 6.2: Bar plot of F1 score on the test set for weakly supervised vs. supervised.

results were dependent on the selection of instances exemplifies that training a model for detecting fake news based on the content of the articles should be done using large amounts of data to minimize this dependency.

An observation was that both the weakly supervised and the supervised model achieved a better F1 score than accuracy. This means the models are slightly better at identifying the fake articles than the real, which indicates that fake articles might have more prominent characteristics that reveal their intentions. When analyzing the plots of differences in F1 score for each model in Figure 6.1 and Figure 6.2, it was observed that the weakly super-

vised models did not consistently outperform the supervised models. Three of five supervised models achieved a better F1 score for the validation set than its weakly supervised equivalent. For the test set, however, four of five weakly supervised models outperformed its supervised equivalent. As previously mentioned, an essential distinction between the validation and test set is that the validation set is retrieved from the same dataset as the training set. In contrast, the test set is gathered manually from fact-checking sites. The test set articles may therefore differ from the training set in style and topics. Additionally, the labels in the validation set are source-based and might therefore be noisy, while the test set has ground truth labels assigned from expert knowledge.

The supervised models better classify the articles in the validation set than the test set, which might be due to the supervised models having overfitted to noisy labels in the training set as the source-based labels are not necessarily correct. Furthermore, as the labels are source-based, there might be a hidden correlation between the source and the content of the articles in the NELA-GT-2019 dataset that the models are able to learn. As the weak labels are assigned based on the article's content, the weakly supervised models might be guided to base their prediction more on the articles' content and not find this correlation. A correlation between the source and the articles' content has not been discovered in this work, but further work should investigate whether this is the case.

As previously mentioned, due to topics of news data being time-dependent, the results could have been affected by the fact that the training data retrieved from the NELA-GT-2019 dataset was gathered over a different time period than the *Manually labeled test set*. Such implications were not studied in this work due to limited time. However, the weakly supervised model could have performed significantly better than the supervised approach if the weakly supervised models were trained on newer topics. Further work could experiment with datasets from two different time periods to simulate a setting where ground truth labels are unavailable for new articles (as it takes time to label news). To conduct the experiment, we propose that the oldest dataset is labeled with ground truth labels, and the most recent dataset is weakly labeled by the weak labeling system. The supervised model is trained on the dataset gathered during the oldest time period, while the weakly supervised model is trained on both datasets. To study the significance of time-dependency of the topics in the data, the models are tested on a test set gathered from the most recent time period.

6.2.4 **Experiment 2.C:** Evaluation of Data Size and Weak Label Ratio

Results

As described in Section 5.2.4, the best models selected in Section 6.2.1 were trained on the data splits defined in Table 5.9. This was done to evaluate the effect of the data size and the ratio of labeled to weakly labeled instances in the training of a weakly supervised model. The results of experiment 2.C are presented in Table 6.11. The same experiments were also run on the baseline model to have a comparable result, and the results are shown in Table 6.12. The supervised model trained on 50,402 labeled instances achieved the best result with an accuracy of 0.959 and an F1 score of 0.959 on the validation set, and an accuracy of 0.793, and an F1 score of 0.813 on the test set. Of the three weakly supervised models trained using different ratios of labeled and weakly labeled data, the one trained

Table 6.11: The results of Experiment 2.C, where the best supervised and weakly supervised models were trained with more labeled samples and different ratios of labeled to weakly labeled instances.

	Validation		**Test set**	
	Accuracy	F1 score	Accuracy	F1 score
Weak$_{all}$	0.800	0.808	0.671	0.721
Weak$_{50k}$	0.900	0.901	0.753	0.778
Weak$_{25k}$	0.942	0.942	0.781	0.801
Supervised	**0.959**	**0.959**	**0.793**	**0.813**

Table 6.12: The results of Experiment 2.C where the baseline supervised and weakly supervised models were trained with more labeled samples and different ratios of labeled to weakly labeled instances.

	Validation		**Test set**	
	Accuracy	F1 score	Accuracy	F1 score
Baseline Weak$_{all}$	0.804	0.809	0.694	0.737
Baseline Weak$_{50k}$	0.843	0.845	0.703	0.731
Baseline Weak$_{25k}$	0.851	0.842	0.703	0.726
Baseline Supervised	0.859	0.860	0.698	0.734

on all the weakly labeled data achieved the worst result with an accuracy of 0.800 and F1 score of 0.808 on the validation set and an accuracy of 0.671 and F1 score of 0.721 on the test set.

Discussion

For the weakly supervised models, it appears that the weakly labeled data introduces more noise than 'help'. This reasoning might be justified when considering the improvement in performance in correlation to the ratio of weakly labeled data: the less weakly labeled data in the training set, the better the model's performance. The weakly supervised models *Weak$_{50k}$* and *Weak$_{25k}$* improved compared to their baseline score for F1, however the *Weak$_{all}$* model performed worse than its baseline on the test set. This is most likely due to the instances with the most confident labels being chosen for training the *Weak$_{50k}$* and *Weak$_{25k}$* models while including all the weakly labeled instances for training the *Weak$_{all}$* model introduced a magnitude of low-quality labels. Thus, these results show that including only high-confidence labels is the best practice for weakly supervised approaches. To increase the number of high-confident labels, a suggestion is to weakly label all the instances in the total NELA-GT-2019 dataset, including the mixed labeled and the unlabeled ones.

The supervised model achieved the best result with an F1 score of 0.813 on the test set, which compared to the baseline model with an F1 score of 0.734 was a considerable improvement. That the supervised model trained on 50,402 instances performed better than

the weakly supervised models in this experiment was expected. The higher performance is probably due to the diversity of topics that such a large dataset will cover while still providing high-quality supervision signals. However, even though the supervised model performed the best, it also performed surprisingly well. Its performance on the validation set, 0.959 for both metrics, was especially eye-catching. Again, a hidden correlation between the content of an article and its source-based label might be prevalent for the validation set. In that case, the model's performance on the validation set is artificially high.

6.3 General Discussion

In comparison to the baselines used in this work, the system as a whole achieved results beyond our expectations. It is clear that the models have learned patterns in the texts, with RoBERTa being the best model at the task. According to our findings, weak supervision can achieve better results in some situations, especially when the amount of labeled data is minimal, which was the case for Experiment 2.B. On the contrary, Experiment 2.C shows that using weakly supervised learning in scenarios with more labeled data actually worsens the model's performance. In the light of Experiment 2.B and 2.C, the choice of learning scenario should depend on the availability of labeled data and thus confirms that a weakly supervised approach is only favorable in cases where the labeled data is limited. A suggestion for further work is to study how the amount of labeled data affect the performance of weak supervision compared to a supervised model.

Another aspect to consider in this work is the choice of datasets. The work of developing large, high-quality, manually labeled datasets for fake news detection remains a challenge. Thus the labels used for training this system were source-based. The quality of the NELA-GT-2019 appears to be a good alternative considering the results obtained in Experiment 2.C on substantial parts of this data. It is essential to bear in mind that the machine learning algorithms ultimately are most dependent on the actual input to the model rather than the parameters used to tune. A model tuned to produce perfect predictions on a low-quality dataset is, in essence, not very useful in a real-world setting. By testing our models on the *Manually labeled test set*, the real-world performance of the system was assessed, which we consider a strength of this work. From this perspective, also considering the rules applied to annotate the manually fact-checked dataset should be considered, i.e., the criteria for labeling a news story as fake news. For example, simply assigning politically controversial news stories or unpopular opinions as fake news should be avoided. As a consequence of implementing models trained on such data, we incorporate these rules into the system, which can ultimately contribute to censorship if used in real-world applications. Therefore, the data used to evaluate the model should be fact-checked by approved fact-checking organizations to uncover such biases in the model. Ideally, the training set should also provide high-quality labels, but as previously stated, large manually labeled datasets are a limited resource and quickly becomes outdated.

On the same note, a related point to have in mind when considering the real-life applications of fake news detection systems is the model's explainability. The model's explainability is closely related to the previous matter regarding biases in the end model based on the training data. Out of all the models implemented in this work, only the LR and XG-

Boost models can provide explanations for the predictions. In contrast, the BERT-based models are based on neural networks, which are not explainable without applying special Explainable artificial intelligence (XAI) techniques, which are currently not widely supported. As the BERT models are the most promising models, solving this issue should be considered a priority. If not, we are risking language models that have unconscious biases being used in real-life applications. A possibility of making the BERT models more explainable would be to apply frameworks such as exBERT to the models (Hoover et al., 2020).

6.3.1 Comparison with Related Work

To put our work in the context of other research conducted within the fake news detection domain, we will compare this work to some of the studies presented in Section 3. We note that it is difficult to measure which systems perform best, as the test set used in this work has not been tested on the other systems.

The most similar study done in comparison to this work is that of Pérez-Rosas et al. (2017) where supervised learning was used to detect fake news within the content of mainstream news data. Although these works are not directly comparable as data, feature extraction and learning scenarios differed, their general approaches were similar. Therefore, they provided a good indication of this work's performance at detecting fake news. As stated in Section 3.2, Pérez-Rosas et al. achieved an accuracy of 76%, which is very close to the result achieved by the weak supervision system trained on 1,380 instances. This result shows that a weakly supervised method, although not excelling beyond the supervised method, achieves similar performance as other supervised methods trained on other data.

Another supervised approach was the FNDNet as proposed by Kaliyar et al. (2020), which achieved an accuracy of 98.36%. The only system developed in this work that achieved a comparable result to the FNDNet was the supervised model from Experiment 2.C, which accomplished an accuracy of 95.9% on the validation set. However, its result on the test set was considerably lower. The high accuracy achieved for FNDNet is most likely due to the test data in their work being very similar to the training data, which was not the case in our work. FNDNet could therefore perform poorly on data with new topics. As mentioned, the performance of the FNDNet might also be a result of the system being based on an untrustworthy data source. Thus their performance is not considered to be completely reliable.

When considering the related weakly supervised approaches, the weakly supervised model in Experiment 2.B achieved a similar result as the WeFEND system, which had an accuracy of 82.4%. Again, this confirms that the proposed approach can provide similar results to using contextual features. Additionally, the WeFEND system is designed for a specific use case, namely for the WeChat platform, and is thus not as scalable to new contexts as our proposed system.

The work done in Helmstetter (2017) was based on a nearly identical approach as this work for Twitter data but without the use of a weak labeling system. The results of their experiments using content-based features showed proximity in performance to the results achieved in Experiment 2.B. This was interesting as tweets and mainstream news articles have structural differences. Tweets are limited to 140 characters and usually consist of a single sentence, which indicates that fake news has characteristics inherent regardless of

the document type and that the length of the text is less critical for detection. It would have been interesting to study whether the length of the text was an essential feature for the best classifier, namely RoBERTa. However, as previously mentioned, BERT models are not directly explainable.

The key takeaway from the comparison to other fake news detection systems is that similar results are achievable by only using the content of the data through weak supervision.

6.3.2 System Improvements

When considering the proposed weakly supervised system as a whole, some possible improvements could be made. Seeing as some of the models, especially the BERT-based models, show decreased performance due to noisy labels, more noise-robust models could have been used in these scenarios to exploit the information in the noisy labels in a better way. Selecting only the weakly labeled articles with the highest confidence helped increase performance, as shown in Experiment 2.C, but further steps could be taken. There are several systems developed for handling noisy labels for deep learning in general (Han et al., 2018; Yi and Wu, 2019), but also for text classification (Jindal et al., 2019). Integrating such noisy label correction techniques with the BERT-based models could help increase the performance of the weak supervision system.

When considering all the end model experiments, including weak and supervised, a way of obtaining results with higher credibility would be to use a larger test set. The test set consisted of only 434 articles, thus using a larger test set would make the results less sensitive to randomness.

Chapter 7

Conclusion and Further Work

7.1 Conclusion

This thesis has explored using a weak supervision approach for detecting fake news articles, consisting of a weak labeling system and weakly supervised end models. The objectives of this thesis were to both find a way to efficiently weak-label news articles based solely on their content and to use the weak labels to train a weakly supervised machine learning model to distinguish between fake and real news content. To accomplish the objectives, this work first evaluated the performance of three weak labeling systems, namely the a manual Snorkel, automatic Snorkel and automatic Snuba system. Then, the five machine learning models Logistic Regression, XGBoost, ALBERT, XLNet and RoBERTa, were trained in both a weakly supervised and supervised learning scenario on two different sizes of the datasets. Finally, the models were evaluated using a manually labeled test set, which was gathered as part of this work.

The final contribution is three-fold: 1) A system that extracts features from the content of mainstream news articles and applies weak labels to generate training data for a machine learning model. 2) A thorough understanding of five machine learning models' performance at detecting fake news articles. The models are trained using a weak supervision approach on a combination of ground truth labels and weak labels, and then compared to a supervised approach using only ground truth labels. 3) A balanced test set gathered from existing datasets and fact-checking sites with manually labeled instances. The research questions (RQs) posed in the introduction are answered in the following manner:

RQ1 *What is the best weak labeling system that uses content-based features for fake news detection?*
Of the three evaluated systems, this work has identified the automatic Snuba system as the best weak labeling system for fake news data. The best Snuba configuration used decision trees created from subsets of three features as the heuristic type. The Snuba system performs better than both the manual and automatic Snorkel system in terms of accuracy, F1, and coverage and achieved an accuracy of 0.765, an F1 score of 0.765, and a coverage of 0.902

on a source-based test set. In this work, the Snuba system generates more complex heuristics than the Snorkel system. Therefore, the results show that a content-based approach for weakly labeling fake news should apply complex heuristics to create high-confidence labels.

RQ2 *Which weakly supervised machine learning model performs best at detecting fake news?*
The best weakly supervised machine learning model for fake news detection was found to be the RoBERTa model. The RoBERTa model achieved an accuracy of 0.779 and an F1 score of 0.798 on the manually labeled test set. As a comparison, the best Snuba weak labeling system achieved an accuracy of 0.646 and an F1 score of 0.668, with a coverage of 0.959 when assessing the correctness of applied weak labels on the same test set.

RQ3 *How is the performance of a machine learning model affected by expanding the training data with weakly labeled data?*
The difference in performance between weakly supervised and supervised models was smaller than anticipated for both data scenarios. In the scenario with limited labeled data, the weakly supervised model outperformed the supervised model by 2.6 accuracy points and 1.9 F1 points on the test set. However, when training the models with considerably more labeled and weakly labeled data, the supervised model outperformed the best weakly supervised model by 1.2 accuracy points and 1.2 F1 points. From these findings, we conclude that weak labels can improve a model's performance in scenarios where access to labeled data is limited, but may degrade the model's performance in scenarios where the labeled dataset is sufficiently large.

A limitation of this work is the lack of using noise-robust classification models to handle the noisy labels in the weak supervision system. As this work has shown, the weakly labeled instances can worsen the performance of the end model when noise-awareness is not implemented. Additionally, cross-validation was not used when evaluating and comparing the end models, thus making proper justifications of the comparisons difficult as variance and randomness could not be considered. Furthermore, the models were tested on an unseen test set that did not originate from the same dataset as the training data. The use of the manually labeled test set is considered a strength, but evaluating the end models on unseen test data from the original data as well could have provided further insight. Moreover, the NELA-GT-2019 dataset used to train the weak labeling systems and the purely supervised scenarios have source-based labels, which may already be considered noisy.

7.2 Further Work

As discussed in the previous chapter, some improvements could increase the performance of the weak supervision system. Considering the improvements and limitations previ-

ously discussed, the four most important points are suggested as further work (FW) and presented as FW1-FW4 below.

FW1 *Improve the weak labeling system.*
As the best weak labeling system was Snuba, we suggest going further with this framework for creating weak labels in the fake news domain. The Snuba weak labeling system could be improved by adding more numerical features extracted from the content and by creating more complex heuristics based on subsets of higher cardinality. Another improvement could be to analyze which features were used to create the heuristics. This would provide further insight into the behavior of the system, which could be exploited to improve the heuristics generated.

FW2 *Implement noise-aware end models.*
Using models robust to noise was not a focus in this work. However, some models are naturally more robust to noisy labels, and the performance of such models may be significantly better when trained on large datasets with weak labels than those tested in this work. A major improvement to the overall system could therefore be implementing noise-aware end models for the classification task.

FW3 *Study the amount of labeled data needed to determine whether to use a weakly supervised approach or supervised approach for fake news detection.*
This work found that the size of the labeled dataset determines which learning scenario should be utilized for fake news detection with content-based features. The weak supervision approach outperformed the supervised model for the scenario with limited availability to ground truth labels. Further work should experiment with different sizes of ground truth labeled data to investigate at which size of labeled data the weak labels start to degrade the model's performance.

FW4 *Determine the impact of time-dependency of topics.*
Considerably more work needs to be done to determine the impact of time-dependency of article topics for weakly supervised fake news detection. Figure 7.1 shows a proposed experiment for an approach to investigate how the time-dependency of topics will affect the performance of a weakly supervised model at this task. The idea behind the experiment is to utilize the newly released NELA-GT-2020. This dataset shares the same characteristics as NELA-GT-2019 but is gathered throughout the year 2020 and thus consists of articles from a more recent time period. In the suggested experiment, a labeled and an unlabeled dataset are extracted from the NELA-GT-2019 dataset. The labeled dataset is then used to train a supervised model. Another unlabeled dataset of instances is extracted from the NELA-GT-2020 dataset. The proposed weak labeling system applies weak labels to both unlabeled datasets, which is then combined with the labeled dataset to train a weakly supervised model. Furthermore, a test set is extracted from the NELA-GT-

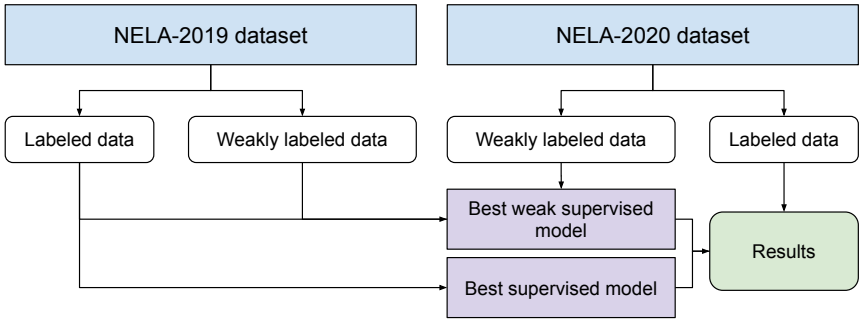

Figure 7.1: Proposed pipeline of an experiment for further work to investigate the impact of time-dependency of topics in news articles.

2020 dataset to evaluate both the supervised and weakly supervised model. This experiment simulates that manually labeling news articles takes time, and that manual labels of news articles published in 2020 are thus unavailable. Instead, the instances can be weakly labeled to train a model to capture new and changing topics. If the weakly supervised model performs significantly better than the supervised model, the experiment could show that the time-dependency of topics significantly impacts fake news detection, and that weak supervision can help combat this challenge.

Bibliography

Abu Salem, F.K., Al Feel, R., Elbassuoni, S., Jaber, M., Farah, M., 2019. FA-KES: A Fake News Dataset around the Syrian War. URL: https://doi.org/10.5281/zenodo.2607278, doi:10.5281/zenodo.2607278.

Albon, C., 2018. Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning. 1st ed., O'Reilly Media, Inc.

Allcott, H., Gentzkow, M., 2017. Social Media and Fake News in the 2016 Election. J. Econ. Perspect. 31, 211–236. URL: https://www.aeaweb.org/articles?id=10.1257/jep.31.2.211, doi:10.1257/jep.31.2.211.

Amirkhani, F.S.A.J.B.H., 2020. FNID: Fake News Inference Dataset. URL: https://dx.doi.org/10.21227/fbzd-sw81, doi:10.21227/fbzd-sw81.

Asiri, S., 2018. Machine Learning Classifiers. Towar. Data Sci. URL: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623.

Asr, F.T., Taboada, M., 2019a. Big Data and quality data for fake news and misinformation detection. Big Data Soc. 6. URL: https://doi.org/10.1177/2053951719843310, doi:10.1177/2053951719843310.

Asr, F.T., Taboada, M., 2019b. MisInfoText. A collection of news articles, with false and true labels. URL: https://github.com/sfu-discourse-lab/Misinformation_detection.

Baccianella, S., Esuli, A., Sebastiani, F., 2008. SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. Proc. 7th Conf. Lang. Resour. Eval. Lr. , 417–422URL: http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf.

Badene, S., Thompson, K., Lorré, J., Asher, N., 2019. Weak Supervision for Learning Discourse Structure, in: EMNLP/IJCNLP.

Bhutani, B., Rastogi, N., Sehgal, P., Purwar, A., 2019. Fake News Detection Using Sentiment Analysis. 2019 12th Int. Conf. Contemp. Comput. IC3 2019 doi:10.1109/IC3.2019.8844880.

Bird, S., Klein, E., Loper, E., 2009. Natural Language Processing with Python. 1st ed., O'Reilly Media, Inc.

Castelo, S., Almeida, T., Elghafari, A., Santos, A., Pham, K., Nakamura, E., Freire, J., 2019. A Topic-Agnostic Approach For Identifying Fake News Pages.

Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System, in: Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., ACM, New York, NY, USA. pp. 785–794. URL: https://dl.acm.org/doi/10.1145/2939672.2939785, doi:10.1145/2939672.2939785.

Chicco, D., Jurman, G., 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21, 1–13. doi:10.1186/s12864-019-6413-7.

Ciampaglia, G.L., Shiralkar, P., Rocha, L.M., Bollen, J., Menczer, F., Flammini, A., 2015. Computational Fact Checking from Knowledge Networks. PLoS One 10, e0128193. URL: http://dx.doi.org/10.1371/journal.pone.0128193, doi:10.1371/journal.pone.0128193.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding URL: http://arxiv.org/abs/1810.04805, arXiv:1810.04805.

Dey, N., Borah, S., Babo, R., Ashour, A.S., 2018. Social Network Analytics: Computational Research Methods and Techniques. Elsevier Science. URL: https://books.google.no/books?id=60vvuQEACAAJ.

Diseases, T.L.I., 2020. The COVID-19 infodemic. Elsevier Ltd. 20. URL: https://doi.org/10.1016/S1473-3099(20)30565-X.

Edgar, T.W., Manz, D.O., 2017. Machine Learning, in: Res. Methods Cyber Secur.. Elsevier, pp. 153–173. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780128053492000066, doi:10.1016/B978-0-12-805349-2.00006-6.

Friedman, J.H., 2002. Stochastic gradient boosting. Comput. Stat. Data Anal. 38, 367–378. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167947301000652, doi:10.1016/S0167-9473(01)00065-2.

From, A.R., Netland, I.U., 2020. Automatic Weak-Labeling of Fake News:A Content-Based Approach.

Goldberg, Y., 2017. Neural Network Methods in Natural Language Processing. volume 10. Morgan; Claypool Publishers.

Gruppi, M., Horne, B.D., Adalı, S., 2020. NELA-GT-2019: A Large Multi-Labelled News Dataset for The Study of Misinformation in News Articles. arXiv:2003.08444.

Gruppi, M., Horne, B.D., Adalı, S., 2021. NELA-GT-2020: A Large Multi-Labelled News Dataset for The Study of Misinformation in News Articles. arXiv:2102.04567.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., Sugiyama, M., 2018. Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels. arXiv:1804.06872.

Helmstetter, S., Paulheim, H., 2018. Weakly Supervised Learning for Fake News Detection on Twitter, in: 2018 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min., pp. 274–277. doi:10.1109/ASONAM.2018.8508520.

Helmstetter, S.F., 2017. Detection of Fake News on Twitter Using Machine Learning .

Hoover, B., Strobelt, H., Gehrmann, S., 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models, in: Proc. 58th Annu. Meet. Assoc. Comput. Linguist. Syst. Demonstr., Association for Computational Linguistics, Online. pp. 187–196. URL: https://www.aclweb.org/anthology/2020.acl-demos.22, doi:10.18653/v1/2020.acl-demos.22.

Horne, B.D., Adali, S., 2017. This Just In: Fake News Packs a Lot in Title, Uses Simpler, Repetitive Content in Text Body, More Similar to Satire than Real News. Proc. First Work. Fact Extr. Verif. , 40–49URL: http://arxiv.org/abs/1703.09398, arXiv:1703.09398.

Horne, B.D., Dron, W., Khedr, S., Adali, S., 2018. Sampling the News Producers: A Large News and Feature Data Set for the Study of the Complex Media Landscape. arXiv:1803.10124.

Howard, J., Ruder, S., 2018. Universal Language Model Fine-tuning for Text Classification. arXiv:1801.06146.

Huberman, B.A., Adamic, L.A., 1999. Growth dynamics of the world-wide web. Nature 401, 131. doi:10.1038/43604.

Jindal, I., Pressel, D., Lester, B., Nokleby, M., 2019. An Effective Label Noise Model for DNN Text Classification. arXiv:1903.07507.

Kaliyar, R.K., Goswami, A., Narang, P., Sinha, S., 2020. FNDNet – A deep convolutional neural network for fake news detection. Cogn. Syst. Res. 61, 32–44. URL: https://doi.org/10.1016/j.cogsys.2019.12.005, doi:10.1016/j.cogsys.2019.12.005.

Kao, A., Poteet, S.R., 2006. Natural Language Processing and Text Mining. Springer Publishing Company, Incorporated.

Kleinbaum, D.G., Klein, M., 2010. Logistic Regression: A Self-Learning Text. 3rd ed. ed., Springer, New York.

Koehrsen, W., 2018. A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning. URL: https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f.

Kornbluh, K., Goldstein, A., Weiner, E., 2020. New Study by Digital New Deal Finds Engagement with Deceptive Outlets Higher on Facebook Today Than Run-up to 2016 Election. Ger. Marshall Fund United States URL: https://www.gmfus.org/blog/2020/10/12/new-study-digital-new-deal-finds-engagement-deceptive-outlets-higher-facebook-today.

Krause, N., Wirz, C., Scheufele, D., Xenos, M., 2019. Fake News: A New Obsession with an Old Phenomenon? , 58–78doi:10.1093/oso/9780190900250.003.0005.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R., 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv:1909.11942.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.

Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Mohri, M., Rostamizadeh, A., Talwalkar, A., 2018. Foundations of Machine Learning. 2nd ed., The MIT Press, Cambridge, Massachusetts.

Monti, F., Frasca, F., Eynard, D., Mannion, D., Bronstein, M., 2019. Fake News Detection on Social Media using Geometric Deep Learning. ArXiv abs/1902.0.

Morin, A., 2020. What is Cognitive Bias? URL: https://www.verywellmind.com/what-is-a-cognitive-bias-2794963.

Morstatter, F., Wu, L., Nazer, T., Carley, K.M., Liu, H., 2016. A new approach to bot detection: Striking the balance between precision and recall. 2016 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min. , 533–540.

Nakamura, K., Levy, S., Wang, W.Y., 2020. r/Fakeddit: A New Multimodal Benchmark Dataset for Fine-grained Fake News Detection. arXiv:1911.03854.

Newman, N., Richard Fletcher, W., Schulz, A., Andı, S., Kleis Nielsen, R., 2020. Reuters Institute Digital News Report 2020 , 112.

Norregaard, J., Horne, B.D., Adali, S., 2019. NELA-GT-2018: A Large Multi-Labelled News Dataset for The Study of Misinformation in News Articles. arXiv:1904.01546.

Paul, K., Vengattil, M., 2020. Facebook removed seven million posts in second quarter for false coronavirus information. URL: https://www.reuters.com/article/us-facebook-content-idUSKCN25727M.

Pennebaker, J.W., Boyd, R., Jordan, K., Blackburn, K., 2015. The development and psychometric properties of LIWC2015. University of Texas at Austin. doi:10.15781/T29G6Z.

Pennebaker, J.W., Francis, M.E., Booth, R.J., 2001. Linguistic Inquiry and Word Count. Lawerence Erlbaum Associates.

Pérez-Rosas, V., Kleinberg, B., Lefevre, A., Mihalcea, R., 2017. Automatic Detection of Fake News URL: http://arxiv.org/abs/1708.07104, arXiv:1708.07104.

Perry, N.W., Greene, E., B., Willoughby, R.H., 1895. Electricity: A Popular Electrical Journal, Volume 8.

Rashkin, H., Choi, E., Jang, J.Y., Volkova, S., Choi, Y., 2017. Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking, in: Proc. 2017 Conf. Empir. Methods Nat. Lang. Process., Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 2931–2937. URL: http://aclweb.org/anthology/D17-1317, doi:10.18653/v1/D17-1317.

Ratner, A., Bach, S.H., Ehrenberg, H., Fries, J., Wu, S., Ré, C., 2017a. Snorkel: Rapid Training Data Creation with Weak Supervision. Proc. VLDB Endow. 11, 269–282. URL: http://dx.doi.org/10.14778/3157794.3157797, doi:10.14778/3157794.3157797.

Ratner, A., Varma, P., Hancock, B., Ré, C., 2017b. Weak Supervision: The New Programming Paradigm for Machine Learning. URL: http://ai.stanford.edu/blog/weak-supervision/.

Reis, J.C.S., Correia, A., Murai, F., Veloso, A., Benevenuto, F., Cambria, E., 2019. Supervised Learning for Fake News Detection. IEEE Intell. Syst. 34, 76–81.

Risdal, M., 2016. Getting Real about Fake News Dataset. URL: https://www.kaggle.com/mrisdal/fake-news.

Roh, Y., Heo, G., Whang, S.E., 2021. A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective. IEEE Trans. Knowl. Data Eng. 33, 1328–1347. doi:10.1109/TKDE.2019.2946162, arXiv:1811.03402.

Schudson, M., Zelizer, B., Derakhshan, H., Wardle, C., Fletcher, R., Nielsen, R.K., Lewis, R., Marwick, A., Watts, D.J., Rothschild, D., Freelon, D., Stroud, N.J., Thorson, E.A., Young, D., Kahan, D., Southwell, B., Boudeywns, V., Oh, S., Sippitt, A., Barker, M., Dias, N., Theis, J., Zollhöfer, M., Samminger, M., Theobalt, C., Nießner, M., 2017. Understanding and Addressing the Disinformation Ecosystem. Annenb. Sch. Commun. , 93URL: https://firstdraftnews.org/wp-content/uploads/2018/03/The-Disinformation-Ecosystem-20180207-v4.pdf.

Shao, C., Ciampaglia, G., Varol, O., Flammini, A., Menczer, F., 2017. The spread of fake news by social bots .

Shi, B., Weninger, T., 2016. Discriminative predicate path mining for fact checking in knowledge graphs. Knowledge-Based Syst. 104, 123–133. URL: http://dx.doi.org/10.1016/j.knosys.2016.04.015, doi:10.1016/j.knosys.2016.04.015.

Shrestha, A., Spezzano, F., Joy, A., 2020. Detecting Fake News Spreaders in Social Networks via Linguistic and Personality Features, in: CLEF.

Shrestha, M., 2018. Detecting Fake News with Sentiment Analysis and Network Metadata.

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., Liu, H., 2019. FakeNewsNet: A Data Repository with News Content, Social Context and Spatialtemporal Information for Studying Fake News on Social Media. arXiv:1809.01286.

Shu, K., Sliva, A., Wang, S., Tang, J., Liu, H., 2017a. Fake News Detection on Social Media: A Data Mining Perspective. ACM SIGKDD Explor. Newsl. 19, 22–36.

Shu, K., Wang, S., Liu, H., 2017b. Exploiting Tri-Relationship for Fake News Detection. arXiv Prepr. arXiv1712.07709 .

Shu, K., Wang, S., Liu, H., 2018. Understanding User Profiles on Social Media for Fake News Detection. doi:10.1109/MIPR.2018.00092.

Själander, M., Jahre, M., Tufte, G., Reissmann, N., 2019. Epic: An energy-efficient, high-performance gpu computing research infrastructure arXiv:1912.05848.

Spohr, D., 2017. Fake news and ideological polarization: Filter bubbles and selective exposure on social media. Bus. Inf. Rev. 34, 150–160. doi:10.1177/0266382117722446.

Starosta, A., 2019. Building NLP Classifiers Cheaply With Transfer Learning and Weak Supervision. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15577251.pdf.

Szpakowski, M., 2018. Fake News Corpus. URL: https://github.com/several27/FakeNewsCorpus.

Thorne, J., Vlachos, A., Christodoulopoulos, C., Mittal, A., 2018. FEVER: a large-scale dataset for Fact Extraction and VERification .

Tversky, A., Kahneman, D., 1973. Availability: A heuristic for judging frequency and probability. Cogn. Psychol. 5, 207–232. doi:10.1016/0010-0285(73)90033-9.

Varma, P., Ré, C., 2018. Snuba. Proc. VLDB Endow. 12, 223–236. doi:10.14778/3291264.3291268.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need. arXiv:1706.03762.

Wang, W.Y., 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. CoRR abs/1705.0. URL: http://arxiv.org/abs/1705.00648, arXiv:1705.00648.

Wang, Y., Yang, W., Ma, F., Xu, J., Zhong, B., Deng, Q., Gao, J., 2019. Weak Supervision for Fake News Detection via Reinforcement Learning URL: http://arxiv.org/abs/1912.12520, arXiv:1912.12520.

Wang, Y., Zhou, Z., Jin, S., Liu, D., Lu, M., 2017. Comparisons and Selections of Features and Classifiers for Short Text Classification. IOP Conf. Ser. Mater. Sci. Eng. 261. doi:10.1088/1757-899X/261/1/012018.

Wendling, M., 2018. The (almost) complete history of 'fake news'.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V., 2020. XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv:1906.08237.

Yi, K., Wu, J., 2019. Probabilistic End-to-end Noise Correction for Learning with Noisy Labels. arXiv:1903.07788.

Yousaf, A., Umer, M., Sadiq, S., Ullah, S., Mirjalili, S., Rupapara, V., Nappi, M., 2021. Emotion Recognition by Textual Tweets Classification Using Voting Classifier (LR-SGD). IEEE Access 9, 6286–6295. doi:10.1109/ACCESS.2020.3047831.

Zhang, X.D., 2020. Machine Learning, in: A Matrix Algebr. Approach to Artif. Intell.. Springer Singapore, Singapore. volume 45, pp. 223–440. URL: https://books.google.ca/books?id=EoYBngEACAAJ&dq=mitchell+machine+learning+1997&hl=en&sa=X&ved=0ahUKEwiomdqfj8TkAhWGslkKHRCbAtoQ6AEIKjAAhttp://link.springer.com/10.1007/978-981-15-2770-8_6, doi:10.1007/978-981-15-2770-8_6.

Zhou, X., Zafarani, R., 2020. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. ACM Comput. Surv. 53. URL: https://doi.org/10.1145/3395046, doi:10.1145/3395046.

Appendix A

Numerical Features

Overview of all numerical features generated in the feature engineering in Section 4.5, that are used as input to the weak labeling systems. The features are grouped by feature type, i.e. stylistic, part-of-speech, sentiment analysis or complexity type features.

A.1 Stylistic Features

Table A.1: Overview of all stylistic features. The * is a substitute for *title* and *content*, indicating that the feature has been generated for both cases.

Feature name	Feature description
*_word_count	Number of words.
*_word_count_with_punctuation	Number of words and punctuation.
*_sentence_count	Number of sentences.
*_capital_word_count	Number of uppercase words.
*_capital_word_ratio	Ratio of uppercase words.
*_stop_word_count	Number of stop words.
*_stop_word_ratio	Ratio of stop words.
*_exclamation_count	Number of exclamation points.
*_exclamation_ratio	Ratio of exclamation points per sentence.
content_quote_marks_count	Number of quote marks.
content_quote_marks_ratio	Ratio of quote marks per sentence.
content_url_count	Number of URLs.

A.2 Part-Of-Speech Features

Table A.2: Overview of all part-of-speech features. The * is a substitute for *title* and *content*, indicating that the feature has been generated for both cases.

Feature name	Feature description
*_verb_ratio	Ratio of verbs per word.
*_past_tense_verb_ratio	Ratio of past tense verbs per word.
*_past_tense_verb_ratio_of_all_verbs	Ratio of past tense verbs per verb.
*_adverb_ratio	Ratio of adverbs per word.
*_adjective_ratio	Ratio of adjectives per word.
content_personal_pronouns_count	Number of personal pronouns.
content_personal_pronouns_ratio	Ratio of personal pronouns per word.
title_nouns_count	Number of nouns.
title_nouns_ratio	Ratio of nouns per word.
title_proper_nouns_count	Number of proper nouns.
title_proper_nouns_ratio	Ratio of proper nouns per word.

A.3 Sentiment Analysis Features

Table A.3: Overview of all sentiment features. The * is a substitute for *title* and *content*, indicating that the feature has been generated for both cases.

Feature name	Feature description
*_sentiment_word_sub	Document subjective score based on words.
*_sentiment_word_pos	Document positive score based on words.
*_sentiment_word_neg	Document negative score based on words.
*_sentiment_sentence_sub	Document subjective score based on sentences.
*_sentiment_sentence_pos	Document positive score based on sentences.
*_sentiment_sentence_neg	Document negative score based on sentences.
*_sentiment_text_sub	Document subjective score based on document.
*_sentiment_text_pos	Document positive score based on document.
*_sentiment_text_neg	Document negative score based on document.
*_swn_pos_score	SentiWordNet positive polarity score.
*_swn_neg_score	SentiWordNet negative polarity score.
*_swn_obj_score	SentiWordNet subjectivity score.

A.4 Complexity Features

Table A.4: Overview of all complexity features. The * is a substitute for *title* and *content*, indicating that the feature has been generated for both cases.

Feature name	Feature description
*_ttr_score	Type-token ratio.
*_avg_word_length	Average word length.
*_avg_word_length_no_stop_words	Average word length including stop words.
content_words_per_sentence	Words per sentence.

Appendix B

Hyperparameter Tuning

B.1 Constant Hyperparameter Values

Table B.1 shows an overview of the constant hyperparameters that were set to a different value than the default, for each model used in this work.

Table B.1: The constant parameters set for the end models.

Model	Parameter	Value
Logistic Regression	`max_iter`	4 000
	`solver`	Liblinear
	`penalty`	L1
	`CV`	5
XGBoost	`n_estimators`	100
ALBERT, RoBERTa, XLNet	`max_seq_length`	256
	`gradient_accumulation_steps`	2

B.2 Best Hyperparameter Values

An overview of the best hyperparameters obtained after performing parameter tuning, as shown in the results in Section 6.2.1.

Table B.2: Best tuned hyperparameter values for end models

	Parameter	Weak Supervised	Supervised
Logistic Regression	C	5	1
XGBoost	max_depth	6	6
	learning_rate	0.3	0.3
	colsample_bytree	1.0	1.0
	subsample	1.0	1.0
	gamma	0	0
ALBERT	num_train_epochs	8	10
	learning_rate	3.246×10^{-5}	4.609×10^{-5}
	train_batch_size	32	32
XLNet	num_train_epochs	9	9
	learning_rate	1.685×10^{-5}	7.139×10^{-5}
	train_batch_size	16	16
RoBERTa	num_train_epochs	10	9
	learning_rate	7.17×10^{-6}	1.474×10^{-4}
	train_batch_size	32	32

Anja R. From & Ingvild U. Netland

Fake News Detection by Weakly Supervised Learning: A Content-Based Approach

NTNU
Norwegian University of
Science and Technology