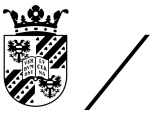


university of groningen
faculty of arts

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Robust Document Representations for Hyperpartisan and Fake News Detection

Author: Talita Anthonio, MA

Advisors: Dr. Rodrigo Agerri, Prof. Dr. Malvina Nissim

Erasmus Mundus Master in Language and Communication Technologies

Final Thesis

June 12, 2019

Department: Computer Systems and Languages, University of the Basque Country UPV/EHU

Abstract

Hyperpartisan news is characterized by extremely one-sided content from a left-wing or right-wing political perspective. This thesis is concerned with automatically detecting such news through supervised text classification. We work with data from the recent shared task on hyperpartisan news detection (SemEval-2019 Task 4). We use two classification techniques: Support Vector Machines (SVMs) and Recurrent Neural Networks. We experiment with document representations using bag-of-words, bag-of-clusters, word embeddings and contextual character-based embeddings. We also try to improve our classifiers by adding local features, such as POS n-grams, stylistic features and the sentiment of a text. Our aim is to build robust classifiers across tasks related to fake news, for different domains and text genres. Although local features help to model the task in-domain, this thesis shows that dense document representations work better across domains and tasks. We obtain very competitive results in the hyperpartisan news detection task and state-of-the-art results in an out-of-domain evaluation on fake news.

keywords: hyperpartisan news detection, fake news, supervised text classification

Contents

List of Figures

List of Tables

1 Introduction

> Respecting the truth and the public right to the truth is the first obligation of journalists. A journalist, in recognition of this commitment, defends the principles of freedom and the right to comment and critique, while finding and reproducing the news properly. A journalist reports only the facts obtained from a trustworthy source, he does not suppress important information and he does not falsify material. He only uses fair methods to find information, photos and other materials. In case he accidentally publishes news which later prove to be wrong, he will correct it.
> *-The International Federation of Journalists (1954).*

Recently, there has been a radical change in the way news is being consumed and produced in society. Namely, social media such as Twitter and Facebook have become popular places for people to read, distribute and discuss news. Unlike traditional media outlets, anyone can register as a news publisher on social media. Consequently, news publishers that do not obey the standards of professional journalism, which are formulated in the quote, can also produce and distribute news via social media (Ribeiro et al., 2018). One effect of this is the production of news articles containing false or biased information. This becomes problematic when such news stories affect the political opinions of readers, which may in turn influence the outcome of political elections. This is likely to happen because of the crucial role that news has in shaping citizens opinions, choices and thoughts.

Besides, a recent study of Bhatt et al. (2018) showed that there were a large amount of biased and false news websites during the presidential elections of 2016. The fact that a large proportion of those websites disappeared after the elections suggests that these news articles may have been specifically aimed at influencing the outcome of the elections. Furthermore, another study revealed that many supporters of Hillary Clinton received their news from mainstream sources, such as the *New York Times* and *The Washington Post*, while Trump supporters were more exposed to publishers of right-leaning biased news on social media (Marwick and Lewis, 2017).

Unfortunately, social media users are often unaware that they read biased or fake news. One reason could be that publishers of legitimate news are also active in social media websites. This makes it difficult for users to distinguish between legitimate and illegitimate news publishers. A possible solution would therefore be to monitor and limit the production of illegitimate news, such as done in traditional news media by scholars and watchdog groups (Ribeiro et al., 2018). However, this is a challenging, perhaps even unsolvable task on social media, where large amounts of textual contents are spread globally every second. A more effective solution might therefore be to build a computational systems that automatically detect whether a news article relies on false information, or uses subjective language that expresses approve or disapproval towards a political approach. The latter will be the main purpose of this thesis.

1.1 Hyperpartisan News

The most popular term to refer to politically biased news is hyperpartisan news. It has been defined as a type of news that is characterized by extremely one-sided content (Kiesel et al., 2019). The term was first mentioned in an article from the New York times Magazine[1], where it was formulated as a style of reporting that departs from the traditional notions of journalistic balance by providing a biased picture of one side of the political debate. The term was coined in context of the US Elections from 2016, implying bias that was either left-leaned, showing overt support of the Democratic Party, or right-leaned, which expresses support to the Republican Party. Hyperpartisan news has additionally been characterized as a specific type of fake news. Fake news is the kind of news in social media that spreads more successfully than the others and are typically extremely one-sided (hyperpartisan), inflammatory, emotional and composed of untruths (Potthast et al., 2017). Although related, we believe that hyperpartisan news and fake news are two distinct types of news. Reporting facts in a subjective manner is not the same as reporting false facts. However, since the two types of news report on similar topics and use similar styles to do so, we surmise that it is likely that features for hyperpartisan news detection are helpful for fake news detection.

In order to further illustrate what hyperpartisan news is, we provide an example of an article that is *left-learning* below.

> *President Donald Trump urged Congress Thursday morning to launch an investigation of the news media, wondering online why so much of our news is just made up.* ***He did not single out a specific story or media outlet that he believed to be guilty of inaccurate reporting. Trump's fake news complaints have been a staple of his political rhetoric, a label he often applied to stories that feature negative reporting about him or his presidency.*** *Most recently, Trump has railed against reports that have characterized his administration's hurricane recovery efforts in Puerto Rico as inadequate, as well as against an NBC News report that Secretary of State Rex Tillerson called the president a moron over the summer and nearly resigned.*

The phrases in the example, especially the ones in bold, are used to express disapproval of Donald Trump and his presidency. In particular, the underlying message of this article seems to be that Donald Trump is pretending that the negative reports about him or his presidency are examples of fake news. The author seems to make this point stronger by mentioning that Trump was not able to come up with an example to prove his statement, i.e., "He did not single out a specific story or media outlet that he believed to be guilty of inaccurate reporting". Moreover, the author uses certain bias-laden words to strengthen her point, for instance by using the words *urged* and *railed*.

Following this, one approach that could be used to detect hyperpartisan news automatically would be to develop a system that relies on a list of words that are frequently used

[1] J. Herman. (2016). Inside Facebook's (Totally Insane, Unintentionally Gigantic Hyperpartisan) Political-Media Machine. (24 August 2016). `https://nyti.ms/2k82R8I`

to express approval or disapproval towards an entity. Still, this method is only probable to be effective when the lexicon is large and when it is used by a system that is able to detect varieties of the same word. Even then, the lexicon is likely to be non-exhaustive. Another approach could therefore be to train a classifier in such a way that it knows which words occur more frequently in hyperpartisan news than in mainstream news by representing documents as bag-of-words. However, language is rich, and people can use different words to express disapproval or support towards an entity. Thus, when a system is trained to detect hyperpartisan news by means of a specific set of words, this system may not work well when it sees words that are not in the vocabulary of the training data. Moreover, even within hyperpartisan news, there are differences in the way how writers express bias. Let us consider another example:

> Donald Trump ran on many **braggadocios** and **largely unrealistic** campaign promises. One of these promises was to be the best, the hugest, the most competent infrastructure president the United States has ever seen. Trump was going to fix every infrastructure problem in the country and Make America Great Again in the process. That is, unless you are a brown American. In that case, you're on your own, even after a massive natural disaster like Hurricane Maria.

Similar to the other example, the author uses certain adjectives to express disapproval of trump, such as *braggadocious*. However, these adjectives did not occur in the previous example. Thus, it would perhaps be more effective to represent documents in such a way that words are grouped by their purpose or general topic, by using knowledge from additional sources than the training data. Ideally, a system that detects hyperpartisan news should be able to know that *Donald Trump*, *Hillary Clinton*, *Barack Obama* are political figures and that the words *braggadocious* and *unrealistic* are used to express disapproval.

A more sophisticated solution to detect hyperpartisan news would therefore be to use denser representations than bag-of-words, by grouping words with similar meanings and/or words that come from the same topic together in a cluster. On the other hand, this could cause us to miss important information. Furthermore, the extent to which we succeed in grouping those words may depend too much on the quality of the resources that are necessary to efficiently group those words. Thus, it is a challenging task to determine which features we should select.

In related studies about the automatic detection of linguistic bias in texts, the problem was usually tackled by building systems that relied on a large combination of local features from the documents, such as: the overall sentiment, the readability, the usage of certain POS n-grams and the occurrence of specific bias-laden words (see Chapter 2). These features could be interesting for hyperpartisan news detection as well, especially when they would be combined with features to capture sarcasm in documents. Nonetheless, because of the stylistic differences between hyperpartisan news articles, we believe that systems relying on such features will not work well on all texts. For instance, the second example that we showed contains more passages that indicate a negative sentiment or sarcasm than the previous article. Still, it could be the case that these features are helpful

when they are used in combination with an effective, dense document representation of a text. There are however, to the best of our knowledge, no studies that experimented with this combination of features.

1.2 Research Questions

The purpose of this research is to build a classifier that automatically identifies for a given news document whether it is an instance of hyperpartisan news or not. In particular, we approach this task by building a supervised machine learning classifier that learns appropriate textual features to discriminate between hyperpartisan and legitimate news. Our research can therefore be motivated by the following general question: *Is it possible to build an accurate supervised machine learning classifier that detects hyperpartisan news?*.

In the previous section, we surmised that the best way to approach this task would be to find an effective method to vectorize documents, rather than building systems that rely on a combination of local features. Therefore, the first step in answering our main research question will be to find the most effective document representation technique to vectorize documents for hyperpartisan news detection. We therefore formulate our first research question as:

> RQ1: What is the most effective method to vectorize documents for hyperpartisan news detection?

In order to answer this question, we will try to create a simple classifier that only relies on one document representation technique, for instance by averaging the word embeddings in a document. Despite its simplicity, we try to build a system that receives competitive results on a test set from a shared task on hyperpartisan news detection. However, some document representation techniques are perhaps more powerful when they are used in combination with other document representation techniques or when they are used together with local features, such as the sentiment or the readability of a text. Because of this, the next step of our study will be to add helpful local features to the classifiers that rely on one of the document representations techniques that we use. In this way, our second research question can be formulated as:

> RQ2: Can we improve our classifiers by combining document representation techniques and/or by adding local features?

In addition to building an accurate classifier on the test set of the competition, our aim is also to build a classifier that performs well on a different dataset of hyperpartisan news. In particular, we attempt to create a classifier that is able to perform similarly well across datasets, rather than being specifically tuned to one training set. In other words, we aim to build a system that provides robust results across datasets of hyperpartisan news. Therefore, we formulate the third research question as:

> RQ3: Is it possible to build a classifier for hyperpartisan news detection that is robust enough to perform well across different datasets of hyperpartisan news?

Finally, we try to take robustness one step further by evaluating our most promising systems on a different but related task, without using specific feature tuning on this task. The task that we select for this purpose is fake news detection. We have mentioned in the previous section that fake news is, to a certain extent, related to hyperpartisan news. Because of this, it would be interesting to develop a system that is able to detect both types of news. Thus, our last research question can be motivated by:

> RQ4: Is it possible to build classifiers for hyperpartisan news detection that can be applied to other related tasks?

We will answer our research questions by evaluating the performance of our systems through different set-ups. Moreover, we will experiment with different document representation techniques to address the sparsity problem in text classification. More specifically, we will work with dense word representations such as word embeddings that aim to compute semantic relatedness between words. These representations also help to tackle the out-of-vocabulary words problem given that they are pre-trained on large amounts of unlabelled data which contains a huge vocabulary size. Despite the promising aspects of these techniques, there is, to the best of our knowledge, few or no work available that has experimented with these type of semantic features for hyperpartisan/fake news detection. We therefore experiment with different types of document and word representations and highlight the extent to which they are robust and effective for hyperpartisan news.

1.3 Contributions

The main contributions of this thesis are the following:

- We provide a systematic comparative analysis of document representation techniques for hyperpartisan news detection that can easily be applied to other text classification tasks.
- We conduct a comparative analysis of document representation techniques for text classification.
- We present the first study to use a wide range of semantic features based on word representations obtained from both in-domain and out-of-domain data for hyperpartisan news detection.
- We provide examples of feature types that are specifically useful for hyperpartisan news detection, but also for tasks that concern the detection of linguistic bias.
- We conduct one of the first studies to experiment with Flair (Akbik et al., 2018), a novel NLP toolkit which allows to combine word embeddings and that uses several techniques to effectively use these word embeddings in text classification.
- We provide a comparative study of the effectiveness of current techniques to create word embeddings with Word2vec, GloVe, FastText and Flair.

- We describe and test an empirical method to develop robust models for fake news and related tasks.
- We obtain very competitive results for in-domain evaluation in the official hyperpartisan news detection benchmark.
- We outline a robust set of features that perform well across tasks related to fake news detection. Our systems outperform previous work in an out-of-domain evaluation on fake news without performing any specific tuning on the classifiers that we used for hyperpartisan news detection.

In the next chapter, we position our project in the existing body of research that has been conducted on hyperpartisan news detection and related tasks. In Chapter 3, we present the data that we used for this study. The Chapters 4 and 5 describe the methods and materials that we used to answer our research questions. In the former, we introduce the different document representations that we used and highlight their advantages and disadvantages. The latter describes the experimental set-up of the project. We subsequently present the results of our studies in the Chapters 6, 7 and 8. We finish our work with an analysis of the results in Chapter 9 and conclude our research in Chapter 10.

2 Related Work

In this chapter we present the theoretical framework that is related to hyperpartisan news detection. Hyperpartisan news is a relatively new phenomenon, and has therefore received little attention in computational linguistics. Therefore, we present studies concerned with tasks which aim to automatically detect linguistic bias. Moreover, since we also develop classifiers for fake news detection in this project, we additionally discuss studies concerned with this task. Despite the differences between the tasks and domains, we hypothesize that the features and systems discussed in these works aid to develop systems that identify hyperpartisan news. After all, the tasks can be approached as sub-tasks within detecting biased language.

2.1 Hyperpartisan Content Detection

This section consists of two parts. In the first part, we describe the related work that is conducted on automatically detecting hyperpartisan news. In the second part, we present a body of research concerned with the manual detection of political bias.

2.1.1 Automatic Detection

The study of Potthast et al. (2017) is one of the few available studies that is particularly focused on detecting hyperpartisan in addition to fake news. They approached fake news detection by investigating the writing style of fake news in relation to hyperpartisan news. Their corpus consisted of 1,627 articles. 826 of these documents were mainstream news articles and the remaining documents were hyperpartisan news articles. Furthermore, 256 of those articles were on the left-wing and 545 right-wing of the political spectrum. To detect hyperpartisan news, Potthast et al. (2017) used a writing style model that relied on commonly used stylistic features and some specific news domain features combined with dictionary-based features. The latter implied the usage of the General Inquirer Dictionaries (Stone et al., 2007). Domain specific features are based on the number of quoted words and external links, the number of paragraphs and their average length in an article. In order to avoid over-fitting, they discarded all features that occurred in less than 10 percent of the documents. Using these stylistic features, the authors managed to build an accurate classifier ($accuracy = 0.75$) that discriminates between hyperpartisan and mainstream news. Hence, it is possible to build a simple classifier that only relies on stylistic features. Nonetheless, they did not test their best system on fake news detection, which would have been interesting to test how specifically useful these features are for hyperpartisan news detection.

Another study that focuses on hyperpartisan news detection is the work of Hutto et al. (2015). Their study is particularly centered on bias detection in news reports and they aim to quantify the amount of the bias. The features that they considered in their study were based on a survey that investigated factors that affect the perception of bias in news stories. They worked with several types of features. First, they performed a structural analysis at

sentence level containing sentiment scores using VADER (Hutto and Gilbert, 2014). They also retrieved the subjectivity score, modality score and general mood of the text. Finally, they computed the readability score using Flesh-Kinchaid Grade (Kincaid et al., 1975). The second type of features that they used were motivated by Recasens et al. (2013), of which they used features such as factive verbs, implicative verbs, assertive verbs and hedges. In addition, they also used coherence markers and several type of words from the Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2015), such as causation words, certainty words, tentative words, third-person pronoun, achievement words, work words, conjunctions, prepositions and adverbs. They used a statistical linear regression model and both forward and step-wise (AIC) to measure the relative quality of each feature for the degree of bias. Their results showed that sentence levels of subjectivity were less meaningful, whereas sentence level measure for modality was a stronger indicator of bias than LIWC certainty words. In addition, their findings proved that the readability was unrelated to the degree of perceived bias. Other poor indicators were implicative verbs, degree modifiers, coherence markers, causation words, conjunctions, adverbs, prepositions and auxiliary verbs. By removing those features their system reached an accuracy greater than 97 percent and accounted for 85.9 of the variance in the human judgments of perceived bias in news.

Finally, when we started with this project, there was a shared task on hyperpartisan news detection organized by SemEval 2019 (Kiesel et al., 2019). We used the data of this shared task (see Chapter 3) for the current project and were able to continue submitting runs on the test set through the software TIRA (Potthast et al., 2019). Furthermore, before starting the thesis, we submitted a system to the competition which obtained an accuracy of 0.621 on the test set (30th out of 42 participants). We use this system as the baseline for this project (see Chapter 5). Moreover, the winning system scored an accuracy of 0.822 on the test set. Since the description papers of the submitted systems were not yet published during the thesis, we were unable to gain knowledge about the winning system and to build further on this knowledge during the thesis.

2.1.2 Manual Detection

The work of Yano et al. (2010) is another study that is concerned with identifying political bias in texts is the work of Yano et al. (2010). Nonetheless, this study is particularly aimed at identifying linguistic indicators of bias in American political blog posts of 2008. Arguably, this falls into a different text genre than news articles. Furthermore, the authors did not test the usefulness of their potentially relevant features by deploying a text classification system. Instead, Yano et al. (2010) created a corpus of texts with labeled by their overall bias. The first step that they conducted to built this corpus was to extract 261.073 sentences from their collected blog corpus, which contained an equal amount of conservative and liberal texts. To identify the sentences, the authors ensured that the sentences satisfied at least one of three conditions. The first was that the sentence should contain one of the “sticky” partisan bi-grams, which were defined as terms that are particular to one side in the political debate. In order to capture those terms from the sentences, the

authors created two lists of those bi-grams: one representing liberal terms (495 bi-grams) and one for the conservative side (539 bi-grams). The second condition implied that the sentence should have at least one of the words in the list of emotional words that they created. This list contained several items from the LIWC dictionary (Pennebaker et al., 2015), including words that could be used to express negative emotion, positive emotion, causation or anger. The final condition was concerned with the membership of "killing-related verbs". The potential relevance of those verbs for determining political bias was derived from Greene and Resnik (2009). Greene and Resnik (2009) observed that when comparing *Millions of people starved under Stalin* to *Stalin starved millions of people* the latter was generally viewed as being more negative towards Stalin. They concluded that killing-related verbs provide strong examples of this phenomenon because they exhibit a particular set of semantic properties that are associated with the transitive verb. Therefore, Yano et al. (2010) used the 11 "kill verbs" from the study of Greene and Resnik (2009), such as *slaughter, assassinate, shoot, poison, strangle, smother* and *suffocate* in their third condition.

Based on these three conditions, Yano et al. (2010) built a corpus of 'biased' sentences in which annotators had to quantify the bias as *none*, *somewhat* or *very*. The results led Yano et al. (2010) construct a list of biased words and their relative frequency of the bias mark. For instance, the words *administration*, *Americans*, *woman* and *single* were strongly associated with liberal bias. In contrast, the words *illegal*, *Obama's* and *corruption* were strong predictors of bias leaning towards the conservative political spectrum. However, the authors observed that although the participants managed to detect bias correctly, it was difficult for them to detect the type of bias. In particular, the annotators had an overall agreement of 0.55.

Another study which showed that determining the strength of the bias in an article is the work of Vincent and Mestre (2019). Annotators disagreed about one third of the articles when they had to label the amount of bias in a text. More specifically, the authors collected a dataset of 1,273 articles labeled by three annotators through crowd-sourcing. All articles were assured to be about political news. 50 percent of this dataset was used for the shared task on hyperpartisan news as a test set and the remaining was publicly made available for the participants to train their systems on. We therefore further explain this dataset in the next chapter.

2.2 Fake News Detection

On one hand, there is a body of research available on fake news detection that relied on traditional supervised classification techniques, such as SVM (Vapnik, 1995). We present a few of these studies in the first part of this section. On the other hand, there are several studies that used neural networks for this task. Because of recent technological advances and the success of deep learning in NLP, it is likely that future studies will continue to use them. We therefore present the studies that worked with neural networks in the second part of this section.

2.2.1 Feature-based Supervised Learning

One interesting study on fake news detection is the work of Pérez-Rosas et al. (2018). The authors worked with two fake news datasets. The first dataset was created by humans via crowd-sourcing. Each participant received several legitimate news articles coming from the following domains: business, entertainment, politics, technology, and education and sports. Then, the task was to rewrite a rewrite a shorter 'fake' version while emulating a journalistic style. The outcome of this procedure was a parallel corpus of 250 news articles and their respective fake versions (500 articles in total). The second dataset was a corpus of fake news that naturally occurs on the web, which was about celebrity news. It contained 500 articles, with an even frequency distribution for legitimate and fake news. Pérez-Rosas et al. (2018) used the following set of features:

- Uni-grams and bi-grams from the bag-of-words representation of each news article with tf-idf weighting.
- Punctuation features that denoted the type of punctuation that was used, derived from the Linguistic Inquiry (LIWC) and Word Count Software (Pennebaker et al., 2015).
- Psycho-linguistic features, also derived from the LIWC.
- Syntactic features representing the context-free grammar of a document.
- Readability features.

The readability features were mainly derived by extracting content features such as the number of characters, complex words, number of syllables, word types, number of paragraphs and long words. In addition, the authors calculated several readability metrics, such as Flesh Kinchaid Grade (Kincaid et al., 1975), the Automated Readability Index (Smith and Senter, 1967), the Gunning Fog Grade Readability Index (Gunning, 1952) and the SMOG readability index (McLaughlin, 1969). The features were used in a linear Support Vector Machine (SVM) classifier with default parameters and evaluated on 5-fold cross-validation. For both datasets, the most accurate systems were classifiers that relied on all features. In particular, their systems reached an accuracy of 0.74 on the fake news corpus that was created through crowd-sourcing and 0.76 on its counterpart.

Pérez-Rosas et al. (2018) also performed an across-domain evaluation in which the models were trained using the crowd-sourced fake news corpus and tested on the other dataset, and the other way around. In both cases, the accuracy of the systems dropped significantly, as they varied between 0.48 and 0.65. A further interesting finding was that readability features seemed to be equally relevant in both domains. Thus, Pérez-Rosas et al. (2018) proved that the domain of the article is an important aspect that should be taken into account when classifying fake news. However, it is crucial to bear in mind that the authors used small datasets and that the datasets differ in another aspect that their domain, which is that only one of them consists of 'natural' fake news. This could also be a valid explanation for the decreased performance.

Horne and Adali (2017) approached fake news detection in a similar manner. They used a similar set of features as Pérez-Rosas et al. (2018), such as the readability of the text, LIWC features and stylistic features involving the style of the writers and the syntax of the text, such as the amount of nouns and verbs. Besides, the authors also used a linear SVM classifier. A small difference is that Horne and Adali (2017) also used sentiment features to detect fake news, by means of the tool SentiStrength (Thelwall et al., 2012), which measures the intensity of positive and negative emotion in a document. However, the main difference with their study is that they also study the title of the articles and their importance to distinguish between fake and legitimate news. In particular, Horne and Adali (2017) observed that fake news articles are longer than legitimate news articles. They also contain simpler words. Furthermore, fake news titles use more capitalized words, significantly more proper nouns, whereas fewer stop-words and nouns overall. The authors surmised that writers of fake news attempt to squeeze as much content into the titles by skipping stop words and nouns to increase the amount of proper nouns and verb phrases. Overall, Horne and Adali (2017) were able to achieve an accuracy score of 0.71 when the body of the text was used and 0.78 when using only the title of the text on 5-fold cross-validation. Still, it is crucial to bear in mind that the authors used two relatively small datasets. One of them contained 36 instances of legitimate news and 35 fake news texts. The other dataset had 75 mainstream news articles and 75 fake news articles. Besides, all articles were about politics. We have seen in the study from Pérez-Rosas et al. (2018) that the predictive power of features can differ per domain. Thus, it remains questionable whether the authors would obtain a similar performance when there would be more documents and/or when they would work with different topics.

The previously discussed studies conducted by Pérez-Rosas et al. (2018) and Horne and Adali (2017) seem to suggest that fake news detection can best be approached by using a large set of different features. This makes us wonder whether it would also be possible to use a less sophisticated approach. This is addressed in the study of Ahmed et al. (2017), who solely used n-gram modeling with tf-idf weighting to detect fake news. They only performed a few data pre-processing steps, such as stop word removal and stemming. Another major difference is that Ahmed et al. (2017) used a large dataset of 12,600 fake news articles and 12,600 legitimate news articles. Besides, the authors also experimented with other classification techniques in addition to a linear SVM classifier: Stochastic Gradient Descent, K-Nearest Neighbour, Decision Trees and Support Vector Machines (SVM) with non-linear kernels. Still, the highest result was obtained by a linear SVM classifier, yielding an accuracy score of 0.92. This classifier used 1 as the value for n and 50,000 as the number of features for the bag-of-words model. Thus, it is possible to detect fake news with a simple linear SVM classifier that only relies on bag-of-word uni-grams.

2.2.2 Supervised Learning with Neural Networks

Within the theoretical framework of fake and hyperpartisan news detection, there are a few studies that have used neural networks to tackle the classification task. One example

of such a study is the work of Bajaj (2017), who experimented extensively with various neural network architectures. In contrary to the previously mentioned works, Bajaj (2017) worked with a large dataset of fake news (63,000 articles). The neural network structures that were used in this study are described below.

- A simple two-layer feed-forward neural network that used the average of all vectors corresponding to the average of all words in the news story. Thus allowed to normalize the vectors by their length while taking every content word into account.

- A bi-directional Recurrent Neural Network (RNN) to consider the entire length of each news article. To deal with the length of each article Bajaj (2017) used a limit of 200 time steps (and shorter examples were padded with zero vectors at the beginning). The author also experimented with Gated Recurrent Units (GRUs) (Cho et al., 2014) and Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997).

- Convolutional Neural Networks (Krizhevsky et al., 2012) with max pooling.

- Attention-augmented Convolutional Neural Networks.

In general, the Recurrent Neural Networks outperformed the other classification techniques. In particular, the highest scores were achieved with GRUs (*f-score=0.84*), LSTMs (*f-score=0.81*) and BiLSTMs (*f-score=0.81*). Thus, the study showed that GRUs and LSTMs are effective classification architectures for fake news detection.

Another study which proved that LSTMs are useful for fake news identification is the study of Ajao et al. (2018). They worked with three different types of LSTMs: LSTM without dropout regularization, LSTM with dropout regularization and LSTM with Convolutional Neural Networks (CNN). Their dataset consisted of 5,800 Tweets. Even though all LSTMs performed well, the highest accuracy was achieved by the LSTM that did not use dropout regularization (*accuracy = 0.8229*). However, the size of their data is smaller than ours and the length of the documents is shorter, since Ajao et al. (2018) are not working with news articles but with Tweets.

Singhania et al. (2017) proposed a different method, in which they built a hierarchical attention network to represent the internal structure of a news article. The model contained four levels: one for words, the headline (both using GloVe word embeddings and a bi-directional GRU), sentences (using a bidirectional GRU), and the headline. It constructed a news vector by processing an article from a hierarchical bottom-up manner. Furthermore, their model relied on three assumptions: words form sentences, sentences form the body and the headline with the body forms the article. The headline was viewed as a distinctive feature of the article that contains a concise summary of the article body and contains useful information in the form of its stance with respect to the body. To extract relevant words of a sentence, Singhania et al. (2017) created a sentence representation that is formed with an attention layer. The same was done to extract relevant features from the headline. On the final level, relevant sentences were identified in the formation of the body by using an attention layer. Using this hierarchical attention network, the authors were able to obtain

accuracy scores between 0.9481 and 0.9677. These scores outperformed the classifiers that simply modeled the structure of the article by concatenating the headline to the body of the article. However, the results were only slightly higher. For instance, a simple SVM model that used bag-of-n-grams with tf-idf weighting already reached an accuracy of 0.9247 on their corpus of 20.372 fake news articles and 20,932 mainstream articles. Other bag-of-words classifiers obtained accuracy scores between 0.9121 and 0.9247. Still, the fact that their attention model managed to outperform these high accuracy scores provides evidence that taking the hierarchical structure of the article into account yields a more accurate approach than bag-of-words.

2.3 Biased Language Detection

In addition to news, there is another genre of online texts in which bias can occur, which is Wikipedia articles. Wikipedia articles are a popular resource for related work on linguistic bias detection in general. An example of a work that is concerned with this is the study of Recasens et al. (2013). This work used a corpus of 7464 Wikipedia articles. All articles contained edits from authors that were associated with a neutral point of view (NPOV) tag. Thus, these edits were likely to be used for the purpose of removing bias. Furthermore, they categorized bias language into two types. The first, which was etymological bias, involves prepositions that are commonly agreed to be true or false and are presupposed, entailed, asserted or hedged in a text. In order to identify this type of bias, the authors used factive verbs, entailments, assertive verbs and hedges.

The other type of bias is framing bias, which occurs when subjective one-sided words are used to reveal the author's stance. This type of bias was detected by extracting subjective intensifiers, one-sided terms (e.g., liberated, captured, pro-life). They trained a logistic regression model that takes each word as a feature and its surrounding context using a 5-gram window capturing the two words to the left and the three words to the right. Given the word w, they extracted around 32 features, including the lemma, POS tag of the left word, POS tag of the left-most word before w, whether one or two words around w is an assertive, factive, report or implicative verb and whether the word w is in the list of Liu et al. (2005) list of positive words and/or negative words. A few of the most contributing features were: the word w under analysis, the POS-tag of the word after w, whether w is a report verb or a positive or negative word. They also computed the readability of the text, using the same metrics as Pérez-Rosas et al. (2018). The authors evaluated their system by outputting as the highest bias ranked word or the two or three lowest ranked words, on which they reached an accuracy score of 0.3435, 0.4652 and 0.5870 respectively. They received the best results by using all features, which showed that contextual, linguistic and sentiment-related characteristics are important.

The work of Hube and Fetahu (2018) also focused on bias detection in Wikipedia. Their set-up differs from Recasens et al. (2013) in that they detected bias in 1000 statements from NPOV Wikipedia articles. The most informative features, which were selected by the chi-square selection algorithm and in a Random Forest classifier, were related to the ratio of biased words in a statement and their context, LIWC features based on psycho-

logical and psychometric analysis and the context in which words from specific lexicons appeared that represented epistemological and framing bias. They used lexicons with report verbs, assertive verbs, factive verbs, positive words, negative words, implicative verbs. Furthermore, they created an automatic dictionary of biased words by crawling all articles under politics from Conservapedia (11,793), which is a wikipedia page according to right-conservative ideas including strong criticism and attacks especially on liberal politics and members of the Democratic Party of the United States. They used this data to train word representations with Word2vec (Mikolov et al., 2013; Mikolov, 2013). They worked with a seed word list and looked to the list of closest words in their word representation. By using these features, they were able to score an accuracy of 0.73 on the test set with a Random Forest classifier.

2.3.1 Summary

The related studies that we described in this chapter provided interesting features and classification architectures for (related tasks to) hyperpartisan news detection. For instance, we have seen in the previous studies concerned with hyperpartisan news detection that stylistic features are useful (Potthast et al., 2017). The studies that were concerned with bias detection in other texts genres than news, pointed out that using a combination of a large set of different features is particularly useful to detect bias linguistically. These features were mainly local features, such as the sentiment, POS n-grams, the usage of an assertive and/or factive verb and the lemma of a word (Recasens et al., 2013; Hutto et al., 2015). Some features that were mentioned in these studies were also effective for fake news detection, such as dictionary-based features from the Linguistic Inquiry and Word Count Software (Pennebaker et al., 2015) and readability features. Because of this, we will also conduct experiments with a few of these features the current work. We will refer to them as 'additional features' or 'local features', since we will use them in addition to our document representations.

Moreover, the works discussed in this chapter proved that a Support Vector Machine is an efficient classification technique for fake news detection, especially when it uses a linear kernel (Ahmed et al., 2017; Gilda, 2017). We therefore decided to work with Support Vector Machines in this study as well. Furthermore, the studies that used neural networks for fake news detection showed that Recurrent Neural Networks (RNNs) are the way to go when using neural networks. One study experimented with a variant of RNNs, namely Long short-term memory (Ajao et al., 2018), which obtained promising results for future work on fake news detection. In the work of Bajaj (2017) however, the author compared the performance of a LSTM to another variant of RNN, namely Gated Recurrent Units (GRUs). The latter performed better than LSTM. Because of the promising results with Recurrent Neural Networks in these related studies, we decided to experiment with them as well. We particularly experimented with GRUs, since they received better results than LSTMs (Bajaj, 2017).

3 Data

In this chapter, we present the two datasets that we will use in the rest of this thesis. The first is the hyperpartisan news dataset from SemEval-2019 Task 4 on hyperpartisan news detection (Kiesel et al., 2019). The second is the fake news dataset from the study of Pérez-Rosas et al. (2018), which we will use to test our models in an out-of-domain setting.

3.1 The SemEval-2019 Task 4 Hyperpartisan News Dataset

The complete dataset that was used for the shared task consists of two corpora: the by-publisher and the by-article corpus. The former is labeled by the overall bias has provided by Buzzfeed journalists or Mediabiasfactcheck.com. In contrast, the by-article dataset is labeled through crowd-sourcing (Vincent and Mestre, 2019). We describe both sets in Section 3.1.1 and 3.1.2 respectively.

3.1.1 By-publisher

The by-publisher dataset is automatically annotated based on a list of publishers of hyperpartisan news (Kiesel et al., 2019). It is further divided into three sets: a training set, a validation set and a test set. The training set was available for participants in the shared task. It contains 600.000 documents, and there are 60 different publishers. Half of these documents are labeled as hyperpartisan news (hyperpartisan=*true*). The remaining documents are labeled as legitimate news (hyperpartisan=*false*). Furthermore, the articles are labeled by their position on the political spectrum. This position is indicated by either one of the following classes: left, left-center, least, right-center and right. Articles that are labeled as being on the *right* or *left* side of the political spectrum are automatically labeled as being hyperpartisan. However, when using the by-publisher dataset, we focused on binary prediction (i.e., the hyperpartisan class: true or false), since this was also the main task in the competition.

The validation set has 150.000 documents. The frequency distribution of the classes is also balanced. Furthermore, this dataset was available for participants that signed up for the shared task, similar as the training set. None of the publishers that occur in this set also occur in the training set. In this way, the organizers ensured that the participants would try to extract features for hyperpartisan news, rather than modeling the features of the publishers themselves (i.e., publisher attribution).

Despite the attractive size of the by-publisher training and validation set, we decided to not train systems on the complete by-publisher training nor validation set. Since we were planning to conduct a large amount of experiments, it was more practical to use a smaller but representative dataset of the by-publisher corpus. This set contains articles from both the by-publisher training and validation set. We will further present this dataset in Section 3.2. Nonetheless, we used the by-publisher training set for other purposes in this study. We will describe this in more detail in the next chapters.

Finally, the test set of the by-publisher corpus contains 4000 articles: 2000 of them are hyperpartisan news and 2000 are mainstream news. This dataset was not used for the competition, but participants could test their classifiers on this dataset for further experimentation. However, the by-publisher dataset was hidden for the participants. It was also not published after the evaluation period of the shared task. However, it is possible to test systems on this dataset by using the virtual machine provided by the organizers. This system is called TIRA and further described in Potthast et al. (2019).

3.1.2 By-article

The complete by-article corpus contains 1,273 articles of political news from hyperpartisan and mainstream websites. The data was manually annotated by three annotators through crowd-sourcing (Vincent and Mestre, 2019). The annotators were asked to grade the bias of the article on a 5-point Likert scale:

1. no hyperpartisan content
2. mostly unbiased, non-hyperpartisan content
3. not sure
4. fair amount of hyperpartisan content
5. extreme hyperpartisan content

The organizers Kiesel et al. (2019) removed the articles of the dataset with a low agreement score and the articles that received a score of 3. The labels were binarized by labeling the articles with a score of 4 and 5 as hyperpartisan. The articles that scored 1 or 2 were labeled as not being hyperpartisan. The result of this procedure was a collection of 1,273 articles that were either labeled as being *hyperpartisan* or not. This dataset was used in the shared task and divided into two parts: one was used for training and the other one was used for testing the participants' systems. The available training set contains 645 articles, of which 238 (37%) articles are hyperpartisan and 407 (63%) are not. The second example that we presented in Chapter 1 was a part of one of the hyperpartisan articles in this dataset. We show the complete version of this article in Figure 1. We also outline the relevant statistics of the by-article dataset in Table 2. Furthermore, similar as for the by-publisher subset, we present the frequency distribution of the publishers of hyperpartisan news and mainstream news in Figure 6 and 7 respectively. It additionally presents the statistics of the other dataset that we used to train our systems, which is a subset of the by-publisher corpus. We describe this dataset in the next section.

The remaining 628 articles (50% hyperpartisan and 50% mainstream) from the by-article corpus was used as the official test set for the competition. Hence, the frequency distribution of the labels in test set is equally distributed, in opposition to the training set. In order to evaluate a system on the test set, participants had to use the same procedure as for the by-publisher test set. Thus, they had to submit their system through TIRA

Trump Just Woke Up & Viciously Attacked Puerto Ricans On Twitter Like A Cruel Old Man

Donald Trump ran on many braggadocios and largely unrealistic campaign promises. One of those promises was to be the best, the hugest, the most competent infrastructure president the United States has ever seen. Trump was going to fix every infrastructure problem in the country and Make America Great Again in the process. That is, unless you're a brown American. In that case, you're on your own, even after a massive natural disaster like Hurricane Maria. Puerto Rico's debt, which the Puerto Rican citizens not in government would have no responsibility for, has nothing to do with using federal emergency disaster funds to save the lives of American citizens there. The infrastructure is certainly a mess at this point after a Category 5 hurricane ripped through the island, and 84 percent of Puerto Rican people are currently without electricity. Emergency efforts after Hurricanes Irma and Harvey reportedly went very well and Trump praised himself as well and even saw his disastrous approval ratings tick up slightly as a result. However, the insufficient response in Puerto Rico has nothing to do with Trump, in his mind, and can only be blamed on the people there who do not live in a red state and have no electoral college votes to offer the new president for 2020. They're on their own. Twitter responded with sheer incredulity at Trump's vicious attack on an already suffering people. YouTube

Figure 1: An example of an article with hyperpartisan content from the by-article training set.

Hyperpartisan class	**Amount**
true	24066 49.65%
false	24401 50.35%
total	48467

Bias class	**Amount**
least	11361 23.44%
left	11659 24.06%
left-center	12272 25.32%
right	12742 26.29%
right-center	433 0.89%
total	48467

Table 1: The distribution of hyperpartisan and bias labels our by-publisher subset.

Metric	By-publisher subset	By-article training
Total tokens	29M	412K
Total types	14M	180K
Total sentences	1.5M	16K
Avg sentences per doc	29.18	25.35
Avg words per doc	598.45	638.55
Avg types per doc	286.69	279.21
Tokens in longest doc	3336	6389
Types in longest doc	1274	1338

Table 2: Corpus statistics of the datasets that we used for classification purposes.

(Potthast et al., 2019). Furthermore, to ensure that the classifiers would not profit from over-fitting to publisher style, the authors constructed the data in such a way that there were no articles from the same publishers in the training and test set.

3.2 By-publisher Subset

We used a subset of the by-publisher corpus that was created by members of the University of Groningen. This dataset contains 48467 articles. The documents come from the training and validation set. The frequency distribution of the hyperpartisan labels (either true or false) and bias labels were presented in Table 1. The table reveals that the frequency of the hyperpartisan labels is equally distributed (approximately 50-50). In contrast, the distribution of the bias labels is skewed, as only 0.89% of the documents are labeled as being right-centered on the political spectrum.

Moreover, the frequency distribution of the Top 20 most frequent publishers of hyperpartisan news and mainstream news are shown in Figure 4 and 5 respectively. Both figures reveal that for both classes, there is a relatively large amount of articles that come from one particular publisher. In case of hyperpartisan news, this publisher is *foxbussiness* (N=10747). For mainstream news it is *abqjournal.com* (N=7505). Because of this, the frequency distribution follows a long-tail pattern: there are many articles from a few publishers.

Nonetheless, it is worthwhile to mention that there are two sources of errors in the by-publisher data which had also permeated our subset. First, the presence of articles that are not about politics, such as the example in Figure 2. More specifically, this article is labeled as being on the right-side of the political spectrum. The second issue is that the articles are automatically annotated. Thus, we assumed that there would be errors in the dataset: some articles may have been labeled as being hyperpartisan, whereas they are not.

KISS band members got patriotic during Saturday night‘s concert in Louisiana, temporarily pausing the classic rock show to lead the crowd in the Pledge of Allegiance. After finishing up their signature song, (I Wanna) Rock and Roll All Nite at the Gretna Heritage Festival, guitarist Paul Stanley thanked the U.S. military and gave a shout-out to Army Maj. Steve Roberts, who was in attendance, The Times-Picayune reported. It‘s always cool to love your country, Mr. Stanley told the crowd. The concert in Gretna wrapped up the band‘s KISSWORLD 2017 Tour in North America and Europe, and it wasn‘t the first time the pledge was made part of the show.

Figure 2: An example of a hyperpartisan text (right-leaned) that is not about politics.

3.3 Fake News Dataset

We selected the fake news dataset presented in Pérez-Rosas et al. (2018) to test the performance of our systems in an out-of-domain evaluation set-up. We have described the general set-up of their study in the previous chapter (see Section 2.2). The most important reason why we decided to use this dataset for this purpose is that the authors evaluated their classifiers across datasets of fake news.

The fake news corpus of Pérez-Rosas et al. (2018) contains two sets: the FakeNewsAMT and the CelebrityNews dataset. We present the relevant corpus statistics of these datasets in Table 3. The fake news articles of FakeNewsAMT were written by participants in a crowd-sourcing set-up. We explained in the previous chapter how this data was created and that the news articles came from the following domains: sports, business, entertainment, politics, technology, and education.

Metric	**FakeNewsAMT**	**CelebrityNews**
Total tokens	68K	1.2M
Total types	45K	29K
Total sentences	2540	10K
Avg sentences per doc	5292	21.60
Avg words per doc	141	2467.58
Avg types per doc	93.89	58.88
Tokens in longest doc	3336	77371
Types in longest doc	1274	87

Table 3: Corpus statistics of the datasets of Pérez-Rosas et al. (2018) that we used for out-of-domain evaluation.

The FakeNewsAMT dataset contains 240 instances of fake news and 240 of legitimate news. In contrast, the CelebrityNews dataset includes 250 samples of fake news and 250 samples of legitimate news. Another difference is that all articles come from the same domain, namely celebrity news. The articles of this dataset were manually collected by the authors. Thus, in contrast to the FakeNewsAMT dataset, the CelebrityNews dataset contains documents that occur naturally on the internet. All articles are about celebrities and other public figures. For further illustration, we show an example of a fake news article

about celebrities from this dataset in Figure 3.

Kanye West Wants to Enter Cosmetics Business Like Kylie Kanye West wants to go head-to-head with his famous sister-in-law ... diving head first into the cosmetic biz dominated by Kylie Jenner. Kanye‘s filed legal docs declaring his intention to produce DONDA brand makeup, perfumes, lotions and other cosmetics. Donda, of course, is Kanye‘s beloved mom who passed away in 2007. He‘ll be up against some stiff family competition. Kylie‘s cosmetics sell out within minutes ... some resell on eBay for 10 times the retail value. Kanye‘s application to snag the DONDA cosmetics line is currently being processed but our sources say at this time he‘s only filed the paperwork in case something develops.He‘s made it clear ... Kanye wants to be the new Martha Stewart, creating a lifestyle brand that includes credit cards, cars, wallpaper screens, furnishings, video games, amusement parks, hotels, fitness centers and healthy fast food. So which one‘s gonna give Kim lip?

Figure 3: An example of a fake news article from the CelebrityNews dataset of Pérez-Rosas et al. (2018).

Rank	publisher	freq.
1	foxbusiness.com	10747
2	counterpitch.org	4010
3	truthdig.org	3072
4	thedailybeast.com	1762
5	dissentmagazine.org	509
6	mintpressnews.com	426
7	dcclotheslines.com	233
8	therealnews.com	231
9	fair.org	222
10	bizpacreview.com	220
11	washingtontimes.com	199
12	joemygod.com	198
13	freebacon.com	195
14	forwardprogressives.com	185
15	leftvoice.org	168
	Total	22096

Table 4: The frequency distribution of the Top 20 most frequent publishers of hyperpartisan news in the by-publisher subset.

Rank	publisher	freq.
1	abqjournal.com	7505
2	nbcnews.com	4868
3	calwatchdog.com	2955
4	natmonitor.com	2500
5	reuters.com	2039
6	billmoyers.com	1130
7	ivn.us	941
8	factcheck.org	733
9	studionewsnetwork.com	390
10	reviewjournal.com	282
11	thetrace.org	206
12	foreignpolicyjournal.com	151
13	thewhim.com	142
14	greensboro.com	59
15	newsandguts.com	45
	Total	25069

Table 5: The frequency distribution of the Top 20 most frequent publishers of mainstream news in the by-publisher subset.

Rank	publisher	freq.
1	thegatewaypundit.com	17
2	opslens.com	14
3	realclearpolitics.com	13
4	nypost.com	10
5	salon.com	8
6	express.co.uk	7
7	opednews.com	7
8	dcclothesline.com	6
9	turtleboysports.com	6
10	pjmedia.com	6
11	trueactivist.com	5
12	video.gq.com	4
13	bearingarms.com	4
14	breakingisraelnews.com	4
15	washingtonexaminer.com	4
	Total	115

Table 6: The frequency distribution of the Top 20 most frequent publishers of hyperpartisan news in the by-article training set.

Rank	publisher	freq.
1	cbsnews.com	24
2	circa.com	21
3	express.co.uk	13
4	heavy.com	12
5	snopes.com	11
6	nbcnews.com	8
7	nfl.com	8
8	insider.foxnews.com	8
9	nytimes.com	7
10	businessinsider.com.au	7
11	king5.au	6
12	kiro7.com	6
13	nypost.com	5
14	kcra.com	5
15	bradenton.com	4
	Total	145

Table 7: The frequency distribution of the Top 20 most frequent publishers of mainstream news in the by-article training set.

4 Document Representations

In this chapter, we describe four different techniques that we used to create document representations in our systems: bag-of-words/characters, bag-of-clusters, word embeddings and contextual character-based embeddings. We highlight the advantages and disadvantages of these techniques. We present the resources that we used to create the document representations that we used as well.

4.1 Bag-of-words

When documents are vectorized as bag-of-words, the length of each vector is equal to the vocabulary size of the training data. Each document is then represented as a vector of its word frequencies. We illustrate the bag-of-words vectorization procedure for two documents in Table 8. Note that the illustration is simplified and that it only shows a proportion of the document vectors.

The bag-of-words approach is established upon the assumption that the relevance between documents can be indicated by the frequencies of their words. In text classification, vectorizing documents as bag-of-words is useful when there are words in the corpus that occur more frequently in documents from a specific class than in the other class. Thus, in our case, vectorizing documents as bag-of-words would be useful when there is a specific set of words that occur more frequently in hyperpartisan news than in non-hyperpartisan news.

There are several variants of the bag-of-words method. One of them is to use bag-of-n-grams, such as word bi-grams or tri-grams. This is a simple technique to represent documents by counting how often a sequence of words occur in a document. In some cases, it might be more effective to count how often sequences of words occur in a document, instead of single words (i.e., uni-grams). For instance, in the example in Table 8, it might be useful to count how often *Donald Trump* occurs in a document, instead of *Trump*. If *Melanie Trump* occurs four times in the next document, then the vectorizer will not add four to the amount of times that it has seen *Donald Trump*, which would happen if the vectorizer would have used word uni-grams.

Another popular variant is to use bag-of-character n-grams, which can be used to capture sequences of characters. In a bag-of-characters representation, the word *Trump* could for instance be represented by 1-to-3-grams as: *Tru*, *rum* and *ump*. In general, there are two advantages of character n-grams. The first is that they are able to detect the morphological units of a word. This is specifically interesting for highly inflected languages, but also for English. The second advantage is closely related to the first, which is that bag-of-characters are more capable of capturing misspellings and out-of-vocabulary words than bag-of-words. For example, suppose that we would train a classifier on the two documents in Table 8 using character 1-to-4-grams. If the word *large* would occur in the test set, then we would still be able to use this word in our classification procedure, as we have seen the word *largely* in the training data.

Document 1
Donald **Trump** ran on many braggadocios and largely unrealistic campaign **promises**. One of those **promises** was **to** be the best, **the** hugest, **the** most competent infrastructure president the United States has ever seen. **Trump** was going **to** fix every infrastructure problem in **the** country and Make America Great Again in **the** process.

Document 2
KISS band members got patriotic during Saturday night's concert in Louisiana, temporarily pausing **the** classic **rock** show **to** lead **the** crowd in the Pledge of Allegiance. After finishing up their signature song, (I **Wanna**) **Rock** and Roll All Nite at **the** Gretna Heritage Festival, guitarist Paul Stanley thanked the U.S. military and gave a shout.

	the	**trump**	**kiss**	**to**	**donald**	**members**	**rock**	**wanna**	**promises**	....
Document 1	6	2	1	2	1	0	0	0	2	
Document 2	6	0	0	1	0	1	2	1	0	
...										

Table 8: A simplified illustration of bag-of-word uni-grams for two documents.

4.1.1 Features

In our systems, we used the following features for bag-of-words/characters:

- Character 1-to-3 grams (uni-grams, bi-grams, tri-grams).
- Character 3-to-5 grams (tri-grams, four-grams, five-grams).
- Suffix of a word (the last 1-to-3 and/or 1-to-4 character n-grams of a word).
- Prefix of a word (the first 1-to-3 and/or 1-to-4 character n-gram of a word).
- Word uni-grams.

For the prefixes and suffixes, we tried several set-ups:

- Prefixes with 1-to-1, 1-to-2, 1-to-3 and/or 1-to-4 characters.
- Tri-gram prefixes (f.i., *Tru*, derived from *Trump*).
- Four-gram prefixes (f.i., *Trum*, derived from *Trump*.
- Both 1-to-3-gram prefixes and 1-to-4 prefixes. The former would be *T*, *Tr*, *Tru* and the latter *T*, *Tr*, *Tru*, *Trum* for the word *Trump*.

In order to vectorize the documents, we applied Term Frequency–inverse Document Frequency (tf-idf) (Jones, 1972) weighting on the word frequencies in the documents. This is a popular alternative to using the normal counts of the documents, because it allows to give a larger weight to rare words and to effectively ignore common words (i.e., stop words and function words). We used the tf-idf vectorizer in scikit-learn[2] and tuned the parameters of the vectorizer with a Grid-search. We describe this procedure in more detail in Chapter 5.

4.2 Bag-of-clusters

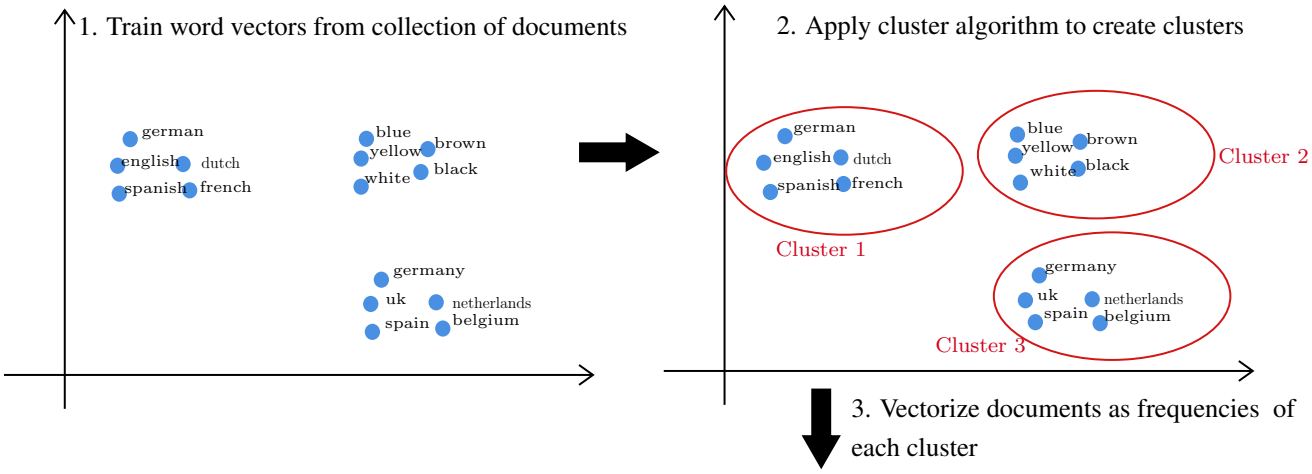
The second technique has been explained in the work of Kim et al. (2017) and Turian et al. (2010), where it is presented as an alternative vectorization method to bag-of-words. The authors referred to this technique as bag-of-concepts, but we will refer to it as bag-of-clusters. This term seemed more suitable to us because it is immediately clear that the method is about groups of words ('clusters').

We illustrate the procedure to vectorize documents as bag-of-clusters in Figure 4. The first step is to take a word embedding model (described in Section 4.3) trained on large corpus of texts, as showed in the left graph of the figure. The next step is to apply a clustering technique, such as k-means, on top of these word embeddings. The result of this procedure is shown on the right of the figure, which shows groups or 'clusters' of words that are related to each other. For instance, one of the clusters contains the words *german, english, dutch, spanish and french*. These words are related to each other because they refer to natural languages. Clusters such as this one can then be used to vectorize documents by counting for each cluster how many words of that cluster occur in a document. This is illustrated in the table in Figure 4.

There are several strong points of the bag-of-clusters method compared to the previous representation technique that we presented. One is that the procedure yields denser vectors, since the length of each vector will be determined by the amount of clusters that we use. Especially when large collections of documents are used, this will result to less sparse vectors than when using bag-of-words, where the length of the vector is equal to the vocabulary size. The second advantage is that the representations encode semantic features of a document, since we are grouping words by their semantic similarity/relatedness while preserving the same level of interpretability of the bag-of-words method. The third benefit is that this method is more robust towards unseen words than bag-of-words. There are two reasons for this. First, the clusters are generated on word embeddings in vector spaces which are created with large corpora. Therefore, it is likely that there are words in the clusters that do not occur in the training data. This tackles the problem of out-of-vocabulary words in the test data. Secondly, since we work with clusters, words that occur in the same cluster will be treated the same.

In addition to these advantages, there was another reason why we decided to use bag-of-

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`



	Cluster 1	Cluster 2	Cluster 3	Cluster 4	...
Document 1	18	1	2	4	
Document 2	0	5	6	3	
...					

Figure 4: An intuitive illustration of bag-of-clusters. The left graph represents a simplified word embedding model. The graph on the right shows the result of applying a cluster algorithm on the word embeddings. The table presents a fictive example of two documents vectorized as bag-of-clusters.

clusters as one of our document representation techniques. Namely, it seemed reasonable why vectorizing documents as bag-of-clusters could be effective for hyperpartisan news detection. The idea was that if there are clusters (i.e., groups of words) that occur more frequently in hyperpartisan than in mainstream news, then it should be possible to train an accurate classifier on bag-of-clusters. Besides, a similar effect has been noted in previous experiments with bag-of-clusters in other tasks, such as in opinion mining (Agerri and Rigau, 2018). Thus, bag-of-clusters seemed to be an interesting alternative vectorization procedure to bag-of-words.

4.2.1 Data and Methods

We present an overview of the three types of clusters that we used in Table 9. The clusters were trained on either one of the following datasets: the Wikipedia dataset using Wikipedia (20121208), the fake news dataset from Kaggle[3] or the by-publisher data from the shared task (both the training and validation set, see Chapter 3). We decided to work with these clusters because of their domain differences. The clusters of the by-publisher data are the most related to our data, whereas the remaining clusters are less related. Besides, the

[3]Dataset available on `https://www.kaggle.com/c/fake-news`.

Wikipedia clusters contain the most words, even more words than the clusters that we trained on the by-publisher data. By comparing the performance of these three clusters, we were able to find out whether it would be possible to use bag-of-clusters when the data is too small to train word embeddings and clusters. This could happen when there is classification problem of which there is only a small amount of data available for training and testing. With this set-up, we were also able to investigate the importance of the association between the task and the corpus of the clusters. In other words, we could find out what would be the most important component to effectively use bag-of-clusters: the domain that it used or the size of the corpus that we used to obtain clusters.

Corpus	**Corpus size**	**Amount of clusters**	**Word embed. dimensions**	**Words**
By-publisher training + validation	498M	Min: 100 Max: 800	50, 300	661501
Wikipedia (20141208)	1700M	Min: 100 Max: 500	50,100,150,200	2053053
Fake news (Kaggle)	23M	Min: 100 Max: 800	50,300	68455

Table 9: A description of the clusters that we used to represent documents as bag-of-clusters. All clusters were applied on word embeddings trained with Word2vec. The amount of words represent the number of words that occur in each cluster file in either the wikipedia, hyperpartisan or fake news cluster files.

As mentioned in Table 9, all word embeddings that we used to create clusters were trained with Word2vec (Mikolov et al., 2013; Mikolov, 2013). We explain this technique later in this chapter (see Section 4.3.2). For now, it is sufficient to know that the dimensions that we presented in Table 9 refer to the dimensions of the word embeddings trained with Word2vec (using the Skip-gram algorithm). Furthermore, the clusters were created with k-means (MacQueen, 1967). This algorithm can be used to cluster data into k number of groups or clusters. The value for k is defined before obtaining the clusters (i.e., *apriori*).

We worked with 16 files for the clusters that we trained on the by-publisher data. Half of these files contained the clusters that were generated on the word embeddings with 50 as their dimension. The other half were generated on the word embeddings that had a dimension of 300. We had 8 files of each of them because we worked with files that contained 100, 200, 300, 400, 500, 600, 700 or 800 clusters. Each file contained 661501 words. The fake news clusters were divided in a similar manner, but contained 68455 words per file. The reason is the lower size of the fake news dataset. The Wikipedia clusters however were organized in a different way than its counterparts. We worked with six files of which each contained 400 clusters. Half of these clusters were obtained on word embeddings trained on a window of 10 tokens. The dimension of the embeddings of each file was: 100, 150 and 200. The remaining three files were obtained on word embeddings trained with a window of 5 tokens. The dimension of the word embeddings of the clusters

of each file was: 50, 100 and 150. In addition, we also worked with one file that contained 500 clusters, trained on embeddings with a dimension of 50 and a window size of 5. Each file of the Wikipedia clusters contained 2053053 words.

Finally, the clusters that we used in this project (see Table 9) were already available on the server of the university of the Basque Country. Using these files, we obtained the word representation for the current word w in document d by using a simple look-up system: we checked for the current lowercased word w if the word occurred in the clusters. If this was the case, then we took the cluster number as a feature (i.e., used in the header of the table in Figure 4). When there was no match, then we did not store any value for the word in the document-term matrix. In this way, we could encode for each document how often a cluster occurred in a document. After obtaining the counts, we applied tf-idf weighting to obtain a final vector representation of the documents. Thus, we used the same procedure as for the bag-of-words (described in Section 4.1).

4.3 Word Embeddings

Word embeddings are word representations of k-dimensional vectors of real numbers. The current techniques to create those word embeddings, such as Word2vec and GloVe, allow to encode the meaning of the words. When these word embeddings are used to vectorize documents, we are able to capture the meaning of the words. This has powerful benefits compared to bag-of-words, where there is no notion of the meaning of words and their semantic relatedness.

The rest of this section consists of two parts. In Section 4.3.1, we present the materials that we used to work with word embeddings in our experiments. In the other sections, we briefly explain the techniques which were used to create the word embeddings that we use in our study: Word2vec (in Section 4.3.2), GloVe (in Section 4.3.3) and FastText (in Section 4.3.4).

4.3.1 Materials

We present a complete overview of the word embeddings that we used in Table 10. We worked with two types of word embeddings. The first were pre-trained word embeddings, which were provided on the web and trained by other researchers. These word embeddings are described in the first five rows in Table 10. They were trained with Word2vec, FastText or GloVe. The second type of word embeddings that we used were the ones that we trained. These word embeddings are described in the last row of the table. We trained these word embeddings on the complete by-publisher training set (see Chapter 3) using FastText. We chose to train them with FastText because we obtained the highest results with this technique when we were using the pre-trained word embeddings (see Chapter 6). In particular, we trained our word embeddings with the same settings as the FastText word embeddings trained on Common Crawl. Therefore, we trained our word embeddings with CBOW. This could easily be done with the FastText package[4].

[4] `https://fasttext.cc/docs/en/unsupervised-tutorial.html`

Method	Training data	Words	Vocabulary size
Word2vec	Google News	100B	3M
FastText	Wikipedia (2017) stamt.org news dataset UMBC webbase corpus	1M 16B	1M
FastText	Common Crawl	600B	2M
GloVe	Wikipedia (2014) Gigaword 5th edition	6B	400.000
FastText	By-publisher training	374M	761.343

Table 10: An overview of the word embeddings that we used. All word vectors had a dimension of 300. We show the size in million or billion words.

We applied a simple procedure to use the word embeddings to create document representations. The first step involved obtaining a vector representation for each word w in document D. Then, we obtained a vector for the whole document by averaging the vectors of the words in a document (Kim et al., 2017; Xing et al., 2014). We repeated this procedure twice: once to experiment with stop words removal and once to experiment with lower casing the words.

There are two important advantages of using word embeddings to build document representations. The first is specifically related to the way how we create the document representations from the word embeddings. Since we obtain the document vector by averaging all the word vectors, we are able to obtain a dense vector representation. In particular, it is the most dense representation compared to the bag-of-words and bag-of-prefixes. Another advantage of using word embeddings is that it is more robust towards unseen words in the test set than for instance bag-of-words. Similar to the bag-of-clusters method, we relied on additional resources to obtain a vector representation. These resources were created from large corpora.

4.3.2 Word2vec

Word2vec (Mikolov et al., 2013; Mikolov, 2013) is a collection of tools that can be used to compute continuous vector representations of words from large datasets ("word embeddings"). The most important idea of Word2vec is that word vectors that share common contexts in the training data are located in close proximity to one another. In other terms, words that are surrounded by the same words have the same meaning, and should therefore be positioned closely together. Thus, the underlying principle of word embeddings that are generated by Word2vec is based on distributional semantics, which implies that the meaning of words can be derived from the context in which they appear in (where the context is defined as their surrounding words) and that words with a similar meaning occur in a similar context. Hence, words with similar contexts will have similar numerical representations.

In Word2vec, the distributed representations of words are learned by a two-layer feed-

forward neural network. The input of this neural network is a preferably large text corpus and the output is a set of vectors for words. To create the word embeddings, the network performs two steps. First, it is trained to predict words/a word for a certain task. There are two kinds of task that can be used for this purpose. In the first, the task is to train the neural network to predict the current word based on its context (i.e., the words surrounding the current word). The algorithm that uses this task is called continuous bag-of-words (CBOW). The other task is similar, but instead of predicting the current word based on the context, it tries to predict the surrounding words (i.e., the context) given the current word. This variant is called skip-gram. Once either of the tasks is completed, the weight matrix that was learned in the hidden layer is used to form the word embedding: each row for a word will be the word embedding of that given word.

There are two important parameters that the user needs to set when training word embeddings: the window size and the dimensions of the word embeddings. The former refers to the amount of words that would involve the context in the algorithms. Usually, this value is set to five, as based on a rule of thumb. Furthermore, the dimensions of the word embeddings is about the size of the embeddings. In general, the larger the vectors, the more information can be encoded. The most commonly used value is 300 (Yin and Shen, 2018).

4.3.3 GloVe

Both the CBOW and the skip-gram architecture of Word2vec have shown promising performances in NLP tasks. However, one of the weaknesses of Word2vec is that it does not effectively make use of global statistics, since its algorithms are trained to predict local contexts. One technique that does take advantage of global count statistics is GloVe (Pennington et al., 2014), which stands for Global Vectors for Word Representation. GloVe is a count-based model that learns the representation of a word by constructing a co-occurrence matrix which counts how frequently a word appears in a certain context in the corpus. This idea is based on techniques that use matrices with global count information to vectorize documents (i.e., global vectorization methods), as in Latent Semantic Analysis (Landauer et al., 1998). Here, the main idea is that words have 'a similar' meaning when they occur in the same contexts. This means that the ratios of the co-occurrence probabilities should be approximately the same. For instance, we would expect words related to water to appear equally in the context of ice and scream, and thus they would have a similar co-occurrence ratio. To account for co-occurrences that occur infrequent and tend to be noisy and unreliable, the authors apply a weight function to weight the occurrences. Then, the result can be used as a vector representation.

4.3.4 FastText

FastText (Bojanowski et al., 2017; Joulin et al., 2016) is another technique to create word embeddings. Word embeddings that are trained with FastText are robuster towards out-of-vocabulary words than Word2vec and GloVe, as words are represented as character

n-grams. In particular, words are portrayed as a bag of character n-grams, where the overall word embedding is the sum of these character n-grams. For example, for the word *where* and n=3, the representation consists of the following: *<wh, whe, her, ere, re>*. The symbols < and > are boundary symbols, thus indicating the beginning and the end of words. This allows to distinguish prefixes and suffixes from the other character n-grams. Moreover, the word itself, *where* is also used. Hence, in addition to character n-grams, FastText also learns a representation for the corresponding word. Then, to build a representation, FastText uses the same skip-gram algorithm with negative sampling from Mikolov (2013). The only difference is the method that is used to compute the similarity between the target word and the context word within the context. Instead of using the dot product between them, the dot product between two words represented as sums of n-grams is used.

4.4 Contextual Character-based Embeddings

One fundamental problem of the models that we presented to train word embeddings is that they generate the same embedding for the same word in different contexts. For instance, the word *bank* will have one vector, despite the different meanings of the word. The word *bank* can namely be used to indicate a *financial institution* or refer to a *river bank*. Nonetheless, this issue has recently been addressed by researchers that aimed to develop so-called contextual word embeddings (Peters et al., 2018). In our study, we work with another interesting type of embeddings that were developed to tackle this issue, namely Flair embeddings (Akbik et al., 2018). We briefly describe this technique in Section 4.4.1 and present the methods that we used in Section 4.4.2.

4.4.1 Flair Embeddings

The authors of Flair (Akbik et al., 2018) propose a method of which the end product is a collection of contextualized character-based embeddings. The most important aspect of their approach is that it models words and their context as sequences of characters. Thus, words are represented as sequences of characters in context (the characters of the word plus the surrounding words). In order to do this, characters are used as the atomic units of a language model. This is achieved by passing a sequence of characters into a Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), that predicts the next character for each character in the sequence. Akbik et al. (2018) use a forward and a backward language model. In the first, states of the LSTM aims to capture the probability distribution of the characters given the previous characters. In the latter, the predictions are computed the other way around: the LSTM learns how to predict the upcoming characters given the previous characters. Then, the authors concatenate the following properties:

- "From the forward LSTM, we extract the output hidden state after the last character in the word. Since the LSTM is trained to predict likely continuations of the sentence after this character, the hidden state encodes semantic-syntactic information of the sentence up to this point, including the word itself" (Akbik et al., 2018, p.4).

- "From the backward LSTM, we extract the output hidden state before the word's first character from the backward LSTM to capture semantic-syntactic information from the end of the sentence to this character" (Akbik et al., 2018, p.4).

The output states are subsequently concatenated to form the final embedding that captures the semantic information of a word and its surrounding context.

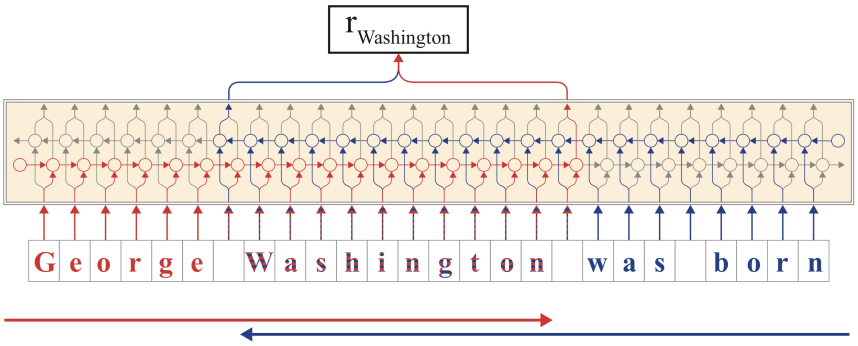


Figure 5: The procedure of creating the contextual string embedding for the word *Washington* from Akbik et al. (2018, p.4).

We show an example of how the embedding for the word *Washington* is build in Figure 5. The sentence is passed as a sequence of characters in a character-level language model to form embeddings. The forward LSTM is depicted by the arrow that points to the right. The output hidden state is extracted after the last character of the word. The backward LSTM is presented by the arrow that points to the left. Here, the authors extract the output hidden states before the first character. Then, the output states are concatenated.

4.4.2 Methods

We worked with pre-trained Flair embeddings in this project. In order to use these embeddings, we loaded them through the package Flair (Akbik et al., 2018). Flair is a recently developed toolkit that can be used to conduct several NLP tasks (e.g., text classification, sequence labeling) with state-of-the-art results. One interesting aspect is that Flair has a simple interface to load and combine static word embeddings (e.g., Word2vec, GloVe and FastText) and contextualized embeddings.

According to Akbik et al. (2018), Flair embeddings work the best when they are used in combination with other word embeddings, to which they can be concatenated. We therefore decided to use the best pre-trained word embedding model (either trained on Word2vec, GloVe or FastText) in combination with the Flair embeddings. We also decided to combine the Flair embeddings with the character embeddings from Lample et al. (2016). These character embeddings are similar to Flair embeddings, since they use characters to form representations of sequences. The main difference is that the character embeddings from Lample et al. (2016) are not pre-trained such as Flair, but that they are trained on

data of the current task. Therefore, they are able to capture task-specific features and can effectively complement the Flair embeddings. Besides, the developers of Flair, Akbik et al. (2018), conducted experiments with these character embeddings and Flair embeddings. They obtained better results with the systems that used Flair and character embeddings than the systems that only used Flair embeddings.

4.5 Summary

In this chapter, we presented several techniques that we used to vectorize documents for hyperpartisan news detection: bag-of-words/characters, bag-of-clusters, word embeddings and contextual character-based embeddings. We also showed the features and materials that we employed to create these representations. In the next chapter, we explain how we used the presented document representation techniques for hyperpartisan news detection and how we conducted feature selection to make efficient document representations.

5 Experimental Set-up

In this chapter, we describe how we employed the document representation techniques that we presented in the previous chapter to train classifiers for hyperpartisan news detection. We show the two supervised classification techniques and describe the general set-up that we used in this study. The latter implies that we explain how we selected the best classifiers and how we continued to evaluate them in other set-ups. We also outline the additional/local features that we used, which we selected based on the work presented in Chapter 2.

5.1 General Set-up

The aim of the current work is to find the best classifier that distinguishes between hyperpartisan and mainstream news using the datasets described in Chapter 3. We define our best model to be the system that has the highest accuracy score on the by-article test set, following the official shared task. In addition, we worked with two supervised classification approaches: Support Vector Machines (SVM) and Recurrent Neural Networks (RNN). We discuss each technique in Section 5.3 and 5.5 respectively. In the rest of the current section, we outline how we evaluated the classifiers within the two approaches.

We selected the best SVM classifiers via 5-fold cross-validation (see Section 6). We trained our classifiers twice: once on the by-article training set and once on the by-publisher subset. Then, we evaluated the best classifiers of each dataset through two set-ups: in-domain and across-datasets. The in-domain evaluation procedure implied that we tested the classifier on a test set derived from the same corpus. Thus, the classifiers that we trained on the by-article training set were evaluated on the by-article test set, while the systems trained on the by-publisher subset were evaluated on the by-publisher test set (see Chapter 7). In contrast, the evaluation across-datasets implied that we trained our systems on data from one corpus and tested it on data from the other corpus of the hyperpartisan news datasets. Hence, the classifiers that we trained on the by-publisher subset were evaluated on the by-article test set. In turn, the classifiers that we trained on the by-article training set were evaluated on the by-publisher subset (see Chapter 8).

We conducted a similar procedure to evaluate the performance of the Recurrent Neural Networks, as we evaluated them in-domain and across-data as well. However, there are three differences in the experimental set-up. The first is that we only trained the classifiers on the by-article training set. The second difference is that we used a different procedure than cross-validation to select the best classifier, which we further describe in Section 5.5. Thirdly, we decided to only evaluate the best classifier from our model development experiments for further evaluation in-domain and across-datasets. Therefore, we tested our best classifier in-domain by evaluating it on the by-article test set and across-datasets by testing it on the by-publisher test set.

Finally, after defining the best systems on the by-article test set, we evaluated the best SVM and Recurrent Neural Network classifier across-datasets by testing it on the by-publisher test set. We also performed an additional evaluation procedure, which we

refer to as out-of-domain evaluation. In this evaluation procedure, we used the features of the classifiers with the most promising results according to our observations on the fake news dataset of Pérez-Rosas et al. (2018). We further describe this experiment in Chapter 8.

5.2 Baseline and Evaluation Metrics

As mentioned previously, we evaluated our SVM classifiers through 5-fold cross-validation. For the in-domain evaluation set-up and the evaluation across-datasets, we additionally used the precision, recall and F1-score as shown below. It is important to bear in mind that we work with a binary classification task in which we either predict hyperpartisan news or non-hyperpartisan news.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{4}$$

As a baseline, we used the classifier submitted to the official shared task before starting this project. This classifier was trained on the complete by-publisher training data and only used the negative sentiment from VADER (Hutto and Gilbert, 2014). We describe this type of feature later in Section 5.4. We show the scores of this baseline system on the by-publisher and the by-article set in Table 11. It obtained the 30th place out of 42 participants on the by-article test set and the 20th position out of 28 submissions on the by-publisher test set. We mainly decided to use this system as our baseline because of its simplicity and the relatively low positions that it obtained on both test sets in the competition.

	by-article test set	**by-publisher test set**
Accuracy	0.621	0.589
Precision	0.582	0.575
Recall	0.860	0.681
F1-score	0.694	0.623

Table 11: The performance of the baseline system that only uses the negative sentiment of a text as features. The system is trained on the complete by-publisher training set.

5.3 Approach 1: Classification with a SVM

A Support Vector Machine (SVM) is a popular binary classification technique. The algorithm has been successfully employed in several text classification tasks, such as sentiment analysis (Jahdav and Vaghela, 2016; Mullen and Collier, 2004; Zainuddin and Selamat, 2014), stance detection (Küçük and Can, 2018), opinion mining (Sabuj et al., 2017), authorship attribution (Diederich et al., 2003) and named entity recognition (Ekbal and Bandyopadhyay, 2010). In Chapter 2, we have seen that they have also proved to be useful for fake news classification. Because of this, we decided to use this classification technique as one of the two main approaches in this thesis.

A SVM (Vapnik, 1995) performs classification by creating a line or hyperplane that separates the data into two classes. This is done by trying to find the best separating hyperplane such that the distances between the data points of the two classes is maximized. The best separating hyperplane is determined by support vectors, which are the critical elements of the training set. The C-parameter of the SVM constructs this hyperplane. In particular, the C-parameter of the SVM classifier determines how much this hyperplane is allowed to make misclassifications: the higher the value of the C-parameter, the more the optimization procedure tries to create a hard margin that does not make any errors. The lower the C-value, the more the SVM attempts to create a wide margin, and the less strict it is towards making errors. Thus, the C-parameter can affect the ability of the classifier to generalize on unseen data, and its robustness.

We use the `LinearSVC` implementation of Pedregosa et al. (2011) in `scikit-learn`[5]. Hence, we worked with Linear SVMs, which means that the hyperplane separates the data of the two classes in a linear manner. We decided to work with this type of SVM because they were also used in previous studies (see Chapter 2). We conducted a few experiments with non-linear SVMS using rbf-kernels. However, the results were not significantly higher than the ones obtained with linear SVMs. Besides, training SVMs with rbf-kernels is more time consuming than training linear SVMs.

5.3.1 Feature Selection for Document Representations

We tested three of our document representation techniques in SVMs: bag-of-words, bag-of-clusters and word embeddings. In particular, we started testing each technique in a SVM without using additional features. In these experiments, we conducted 5-fold cross-validation to select the best features and/or corpora to work with:

- Bag-of-words using the features described in Section 4.1, such as characters, prefixes and words.
- Bag-of-clusters, using the clusters described in Section 4.2.
- Word embeddings, using the materials described in Section 4.3.

[5] `https://scikit-learn.org/stable/index.html`

We trained classifiers on the by-article training data and on the by-publisher subset respectively. Afterwards, we selected the best bag-of-words, bag-of-clusters and word embeddings classifier of each dataset for in-domain evaluation and evaluation across-datasets. Once we found the best bag-of-words, bag-of-clusters and word embedding classifiers on the two datasets via 5-fold cross-validation, we tried to improve the accuracy of these models by adding the features that we present in Section 5.4 and/or by combining document representations.

5.3.2 Pre-processing and Parameter Tuning

Pre-processing The by-article training set was provided in XML-files of which we extracted the body of the documents and their corresponding labels. We did not further clean the documents, as the documents were reasonably clean. The data of the by-publisher subset already contained the extracted documents from the XML-files. Similar as to the by-article training set, we used the body of the documents. As there was still a substantial amount of noise in the documents of the by-publisher subset, we performed a few cleaning procedures to remove the remaining HTML tags in the documents. We additionally removed random occurrences of question marks in the body of the documents.

Parameter Tuning We tuned two types of parameters: the C-parameter of the SVM classifier and the parameters of the tf-idf vectorizer. We performed parameter tuning with a grid-search[6] using the values described in Table 12. Furthermore, we tuned the parameters of the following systems on the by-publisher subset and the by-article training set:

- The best bag-of-words classifier.
- The best bag-of-clusters classifier.
- The best classifier using linguistic features.

These classifiers were the systems with the highest accuracy on 5-fold cross-validation, in which they were using the default settings of the SVM and the tf-idf vectorizer.

Note that the linguistic features belong to the additional features and that we will describe them in Section 5.4. Since the linguistic features are represented by n-grams and tf-idf weighting, we decided to tune the parameters of these features as well. As a point of departure, we experimented with uni-grams, bi-grams and tri-grams via 5-fold cross-validation on both datasets. Once we found the best value for n via 5-fold cross-validation for the by-publisher subset and the by-article set, we used a grid-search to tune the tf-idf vectorizer parameters and the C-parameter to optimize the performance. The values that we used to perform the grid-search were the same as presented in Table 12.

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

Name	Values	Description
lowercase	True, False	Lowercases the words
min_df	1,2,3	Excludes terms that appear in fewer than n documents
sublinear_tf	True, False	Replaces term frequency with 1 + log(tf)
use_idf	True, False	Enables inverse-document-frequency reweighting
C	0.01, 1.0, 100	The C-parameter of the SVM

Table 12: The values over which we performed the grid-search to tune the parameters of the tf-idf vectorizer and the C-parameter of the SVM.

5.4 Additional Features for the SVMs

After obtaining the best SVM classifiers using either of bag-of-words, bag-of-clusters or word embeddings, we tried to increase the performance of the SVMs by adding of local features and/or by combining the most effective document representations. In this section, we describe the local features that we used for this purpose. We selected a collection of features that were mentioned to be effective for related tasks to hyperpartisan news detection presented in Chapter 2. The features that we continued to experiment with in our study can be divided into three distinct groups: sentiment features (see Section 5.4.2), linguistic features (see Section 5.4.2) and stylistic features (see Section 5.4.3).

5.4.1 Sentiment Features

In order to extract the sentiment of each document, we worked with dictionary-based features and automatic approaches. The latter implied that we used `NLTK`, in particular the VADER sentiment analysis tool (Hutto and Gilbert, 2014) to compute the sentiment score of each document[7]. Valence Aware Dictionary for Sentiment Reasoning (VADER) is a simple rule-based model for sentiment analysis that represents the intensity the sentiment of a document by calculating a negative, positive and neutral sentiment and a compound score. The tool was constructed by using a combination of qualitative and quantitative methods that led to an empirically validated gold-standard list of lexical features along with their sentiment intensity measures. Then, these features were used in five different rules that consider grammatical and syntactic aspects to further determine the intensity of a sentiment.

We conducted experiments with the positive, negative and compound score. We surmised that mainly the positive and negative score could help to indicate bias in the texts. In addition, the compound score seemed to be useful because it allows to compute the sentiment of a text in a single one-dimensional metric and encodes both the neutral, positive and negative scores. The latter was computed by summing up the valence scores of each word in the lexicon. Then, the result was adjusted according to the rules and normalized to be between -1 (most extreme negative) and 1 (most extreme positive). The positive and negative scores were expressed in percentages. Thus, with the neutral sentiment, the sum

[7] `https://www.nltk.org/_modules/nltk/sentiment/vader.html`

of the positive, negative and neutral sentiment should sum up to 1. We experimented with the following set-ups using:

- only the negative sentiment
- only the positive sentiment
- only the compound score
- both positive and negative score
- the positive score added to the negative score.

In addition to VADER, which proved to be useful for bias detection in news reports (Hutto and Gilbert, 2014), we also tried to capture the sentiment of texts by using lexicons with positive and negative words. More specifically, we worked with the lexicon of positive and negative words from Liu et al. (2005). We used this lexicon because it was also used in previous studies, such as in the work of Recasens et al. (2013).

We worked with two methods to use the sentiment of the texts as features. In the first, we followed the binary representation method described in Recasens et al. (2013). This implied that if a word from a document occurred in the list of positive words, then the value for positive sentiment would be set to 'true', and the same would be done for the negative words. The second type of feature representation involved counting the amount of positive or negative words. In particular, we represented the intensity of the negative sentiment by using the total number of negative words occurring in the document from the lexicon. We repeated the same procedure to calculate the negative sentiment by counting the total amount of words occurring in the lexicon of negative sentiment words.

5.4.2 Linguistic Features

The linguistic features that we used in this study involved experimenting with Part-of-speech (POS) n-grams. In previous studies, we have seen that tri-grams and uni-grams were particularly useful for this purpose (Horne and Adali, 2017; Recasens et al., 2013). These studies showed that writers can use several linguistic devices to express bias, such as the adjectives *great, beautiful, good* and intensifiers such as *very*. When these devices are used to express bias, they specifically occur in combination with nouns denoting names, organizations, objects, decisions, etc. On the other hand, in mainstream texts, authors could use specific words to denote objectivity, such as *might*, *could*. Furthermore, one could use specific POS tri-grams to use sources in an article, such as *is said to be*.

5.4.3 Stylistic Features

We additionally experimented with features that aimed to capture the stylistic elements of the texts. One method how we did this was by looking at the usage of certain words in the texts. In this light, we considered two groups of words: assertive verbs and factive verbs

(Hooper, 1975). Our decision to use them was based on their motivation and presented results of Recasens et al. (2013). We used a list of assertive and factive words that we took from Wyse (2009), since there was, to the best of our knowledge, no other way to obtain the lexica. The list of assertive verbs included 65 verbs such as *claim, predict, believe, think, suppose, expect, decide*. The list of factive verbs contained 63 words, such as *realize, remember, amuse*.

Another method that we applied to capture stylistic elements was by using readability scores. We decided to use these scores because biased texts tend to have an easier readability level than non-biased texts, (Hutto and Gilbert, 2014; Horne and Adali, 2017; Potthast et al., 2017). The same was mentioned about the readability level of fake news compared to legitimate news (Pérez-Rosas et al., 2018). Following these works, we decided to experiment with the following readability metrics: Flesh Kinchaid Grade (Kincaid et al., 1975), the Automated Readability Index (Smith and Senter, 1967), the SMOG readability index (McLaughlin, 1969) and the Gunning Fog Grade Readability Index (Gunning, 1952). We used the implementations of the package `textstat`[8] to compute the readability scores. We tested the performance of each metric individually on both the by-article and by-publisher subset. We subsequently conducted experiments on both datasets where we used all readability metrics.

5.5 Approach 2: Classification with Recurrent Neural Networks

Apart from that neural networks have revolutionized machine learning, there were three specific reasons why we decided to work with them in this project. The first was that it seemed interesting to test the best word embeddings from the experiments with SVM in a different classification technique. This would enable us to shed more light on the individual performance of the two classification techniques when using word embeddings. Secondly, using neural networks would allow us to build document representations from embeddings with a more sophisticated method than simply averaging the vectors. The third reason was that we were interested in working with contextual character-based word embeddings, in particular the Flair embeddings (see Section 4.4.1). The most efficient method to work with these Flair embeddings was by using the text classifiers provided in the Flair package, which rely on neural networks.

In Flair, there are several ways to use word embeddings to build document representations for text classification, including the method that we used to form document embeddings in SVMs (see Section 4.3). However, as mentioned previously, we decided to use a different method to create document representations. The procedure that we used involved the usage of document embeddings through Recurrent Neural Networks. The first step was to create document embeddings by using the document embeddings object from Flair[9] using the default settings. Because of this, the gated recurrent unit (GRU) was used as the type of the RNN. After obtaining the document embeddings, the next step by

[8] `https://pypi.org/project/textstat/`

[9] `https://github.com/zalandoresearch/Flair/blob/master/resources/docs/TUTORIAL_5_DOCUMENT_EMBEDDINGS.md`

default was to feed these word embeddings to a linear neural network layer that predicts the class[10]. We used GPUs to train the models, which were available at the University of the Basque Country.

Furthermore, training and evaluating a classifier with Flair requires the usage of a training, development and test set. Because of this, we divided the by-article set into a training set (516 documents, 80%), a development set (65 documents, 10%) and a test set (64 documents, 10%). We used the stratified sampling technique from `scikit-learn`[11] to have an approximately equal amount of instances from the hyperpartisan and mainstream class in the three sets. Furthermore, the performance was evaluated using the precision, recall, and F1-score as presented in Section 5.2. While Flair provides the micro and macro accuracy, we decided to use only the micro accuracy because of the imbalanced frequency distribution of the classes in the by-article training set.

Once we found the best classifier from the development experiments, we trained this classifier for another four times. Afterwards, we selected the best model out of these runs for further evaluation on the official by-article test set (in-domain evaluation). The best model was determined by the accuracy on the predictions of the test set derived from the by-article training set. We decided to use this procedure because deep learning architectures tend to produce different outcomes for each experiment.

5.5.1 Feature Selection

We trained three classifiers on each of the following set-ups:

- The most effective word embeddings from our SVM experiments. In our case, these were the FastText word embeddings trained on Common Crawl.
- Two character-based contextual Flair embeddings (`flair-news-forward-fast`) and (`flair-news-backward-fast`)[12] plus the previous features.
- The character embeddings from Lample et al. (2016) and the two types of Flair embeddings.

The first set-up allowed us to compare the effectiveness of the RNN architecture to the results of the SVM classifiers in which we averaged the word embeddings. The second set-up was useful to find out how well the best 'static' word embeddings (i.e., trained with Word2vec, GloVe or FastText) would perform when they would be combined with the two kinds of Flair embeddings, and the other way around. We performed the latter set-up because of the promising results of the experiments from Akbik et al. (2018).

[10] `https://github.com/zalandoresearch/Flair/blob/master/Flair/models/text_classification_model.py`

[11] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

[12] We had to select the 'fast' version of the embeddings due to lack of GPU memory

6 Model Development Results

In this chapter, we present the results of the experiments that we conducted to determine the most promising classifiers for in-domain evaluation and evaluation across-datasets on the test sets of the competition. The purpose of the experiments that we present here was to find the following classifiers:

1. The best SVM classifiers trained with any of the three document representations (bag-of-words, bag-of-clusters or word embeddings), presented in Section 6.1.

2. The best SVM classifiers trained with any of the three document representations that we improved with additional features, described in Section 6.3.

3. The best classifier that relied on contextual character-based embeddings using Recurrent Neural Networks, described in Section 6.4).

We first studied the performance of the additional features in isolation (i.e., without using them in combination with document representation techniques), to decide which features we should add to the classifiers in (1). We present these results in Section 6.2. We additionally describe the results of the experiments where we combined document representations and additional features in Section 6.3. Note that we only show the most relevant results of our experiments, as the purpose of this chapter is to present the most effective features to the reader.

6.1 Document Representations in SVMs

We report the results of the classifiers using any of the three document representations that we used in SVM and presented in Chapter 4. We present the performance of these document representations without using any other features in the classifier.

6.1.1 Bag-of-words

We show the results of the models using the bag-of-word features described in Section 4.1 in Table 13 and 14. The former represents the results on 5-fold cross-validation for the by-article training set. The latter outlines the results for the by-publisher subset. We additionally present the parameter settings of each classifier in the Tables 13 and 14. When we mention in the tables that we used the tuned parameter settings, we mean that we used other settings than the default values of the tf-idf vectorizer and the default value of the C-parameter of the SVM. These settings were the result of the grid-search procedure, described in Section 5.3.2. Moreover, when we mention that we used the default parameter settings, we mean that the default settings were the outcome of the grid-search.

Overall, the results in the tables show that we achieved higher accuracy scores on the by-publisher subset than on the by-article training set on 5-fold cross-validation. The scores of the by-publisher subset varied between 0.8871 and 0.9185, whereas the accuracy scores of the by-article training set varied between 0.7629 and 0.7891. The results also

show that word uni-grams, character 3-to-5 grams and prefixes (1-to-3 and 1-to-4) are powerful features for hyperpartisan news detection. More specifically, we obtained the best results with bag-of-prefixes on the by-article training data (*accuracy=0.7891*). In contrast, the best performance on the by-publisher subset was received by the classifier using word uni-grams (*accuracy=0.9185*).

Based on these results, we selected the best classifier using bag-of-words/characters features of each dataset for in-domain evaluation and evaluation across-datasets on the test sets. Thus, we continued to work with prefixes when training models on the by-article training set. We subsequently experimented with word uni-grams when we trained classifiers on the by-publisher subset. We also used prefixes and word uni-grams in combination with other features in the upcoming cross-validation experiments.

System	**Parameter settings**	**Accuracy**
Prefixes (1-to-3 and 1-to-4)	Default	**0.7891**
Word uni-grams	Tuned	0.7845
Character 3-to-5 grams	Tuned	0.7815
Prefixes (1-to-3)	Default	0.7752
Prefixes (1-to-4)	Default	0.7629

Table 13: The Top 5 bag-of-word classifiers trained on the by-article training set and evaluated via 5-fold cross-validation.

System	**Parameter settings**	**Accuracy**
Word uni-grams	Tuned	**0.9185**
Character 3-to-5 grams	Tuned	0.9013
Prefixes (1-to-3 and 1-to-4)	Default	0.9007
Prefixes (1-to-4)	Default	0.8908
Character 1-to-3 grams	Default	0.8871

Table 14: The Top 5 bag-of-word classifiers trained on the by-publisher subset and evaluated via 5-fold cross-validation.

6.1.2 Bag-of-clusters

In this section, we present the performance of our SVM classifiers using the clusters described in Section 4.2, Table 9. We use a specific format in our tables to refer to the clusters. In particular, the first component refers to the corpus on which the word embeddings were trained, which is either the Wikipedia dataset using Wikipedia (20121208) (`wiki`), the fake news dataset from Kaggle (`fake`) or the by-publisher training and validation set (`hyp`).

Then, the name is followed by an s-*number*, which refers to the dimension of the word embeddings on which the clusters were applied. For instance, the name `hyp-s50-15.400` describes a set of 400 clusters trained on the complete by-publisher training set, using 50 dimension word embeddings trained with a 5 token window.

The accuracy scores in Tables 15 to 17 reveal that bag-of-clusters is a helpful document representation technique for our task. However, their performance on 5-fold cross-validation is generally lower than the classifiers working with bag-of-words (see Table 13 and 14). This is specifically the case for the classifiers using the by-publisher subset, of which the scores varied around 0.90. Moreover, the results show that the clusters generated on hyperpartisan/mainstream news were the most effective. We obtained the second best results with the fake news clusters and the lowest results with the Wikipedia clusters.

When we selected the best cluster of each table for further parameter tuning, the default settings proved to be the best. Consequently, the hyperpartisan/mainstream news clusters were still the best ones. Because of this, we continued experimenting with these clusters in the remaining set-ups: in the experiments that used the best classifiers from Table 17 of each dataset combined with additional features, in the in-domain evaluation procedure and in the evaluation across-datasets.

System	**Accuracy**
Wiki-s50-w5.400	**0.7335**
Wiki-s50-w5.500	0.7241
Wiki-s50-w5.400	0.7164
Wiki-s100-w10.400	0.7149
Wiki-s100-w5.400	0.7134

(a) Results by-article training set.

System	**Accuracy**
Wiki-s100-w5.400	**0.7745**
Wiki-s100-w10.400	0.7613
Wiki-s200-w10.400	0.7600
Wiki-s50-w5.400	0.7502
Wiki-s50-w5.400	0.7498

(b) Results on by-publisher subset.

Table 15: The Top 5 results obtained with Wikipedia clusters on 5-fold cross-validation using default parameter settings.

Cluster	**Accuracy**
Fake-s300-w5-700	**0.7629**
Fake-s50-w5-600	0.7599
Fake-s300-w5-800	0.7551
Fake-s50-w5-700	0.7519
Fake-s300-w5-400	0.7505

(a) Results on by-article training set.

Cluster	**Accuracy**
Fake-s300-w5-700	**0.8123**
Fake-s300-w5-600	0.8076
Fake-s50-w5-700	0.8018
Fake-s50-w5-800	0.7963
Fake-s50-w5-600	0.7953

(b) Results on the by-publisher subset.

Table 16: The Top 5 results obtained with fake/legitimate news clusters on 5-fold cross-validation using default parameter settings.

Cluster	Accuracy
Hyp-s300-w5-700	**0.7645**
Hyp-s50-w5-700	0.7598
Hyp-s300-w5-500	0.7583
Hyp-s50-w5-400	0.7581
Hyp-s50-w5-800	0.7566

(a) Results on the by-article training set.

Cluster	Accuracy
Hyp-s300-w5-800	**0.8556**
Hyp-s300-w5-700	0.8498
Hyp-s300-w5-500	0.8465
Hyp-s300-w5-600	0.8457
Hyp-S50-w5-800	0.8429

(b) Results on the by-publisher subset.

Table 17: The Top 5 results obtained with hyperpartisan/non-hyperpartisan clusters on 5-fold cross-validation using default parameter settings.

6.1.3 Word Embeddings

In this section, we present the results of the SVM classifiers using the pre-trained word embeddings described in Table 10. The results on the by-article training set are described in Table 18. The results on the by-publisher subset are presented in Table 19.

In general, it can be noted that we received the best scores with pre-trained word embeddings from FastText on both datasets. More specifically, the best classifier on the by-article training set was the one that used FastText word embeddings trained on Common Crawl. These word embeddings also reached the highest score on the by-publisher subset. Still, the accuracy slightly improved when we removed the stop words of the documents (see Table 19).

Another observation is that our own trained word embeddings, which we trained on the by-publisher training set, received a fair accuracy score. The classifier that used these embeddings namely obtained the third best score on the by-publisher subset (see Table 19). Still, the results of these word embeddings were not among the Top 5 best classifiers that we trained on the by-article training set.

In addition, the results of the different pre-processing procedures that we applied show that removing stop words helped to increase the performance for some classifiers, but not for all. Moreover, lowering the case of the words did not generally improve the results. None of the results of the models that used this pre-processing procedure were among the Top 5.

For the in-domain evaluation and the evaluation procedure across-datasets, we selected the best classifier from Table 18, and the best classifier from Table 19. We also used these classifiers in combination with other features in the remaining experiments on 5-fold cross-validation.

6.1.4 Overview of the Best Classifiers

So far, we received promising results for all three document representation techniques on 5-fold cross-validation. We selected the classifiers with the best features of each document representation technique as a point of departure for further development and evaluation.

Technique	Corpus	Pre-processing settings	Accuracy
FastText	Common Crawl	None	**0.7754**
FastText	Common Crawl	Without stop words	0.7723
Word2vec	Google News	Without stop words	0.7708
GloVe	840B.300D	Without stop words	0.7662
GloVe	840B.300D	None	0.7647

Table 18: Results for the pre-trained word embeddings for the models that we trained on the by-article training set via 5-fold cross-validation.

Technique	Corpus	Pre-processing settings	Accuracy
FastText	Common Crawl	Without stop words	**0.8125**
FastText	Common Crawl	None	0.8100
FastText	By-publisher training	None	0.8097
Word2vec	Google News	Without stop words	0.8038
FastText	Wikipedia	None	0.8037

Table 19: Results for the pre-trained word embeddings for the models that we trained on the by-publisher subset via 5-fold cross-validation.

In particular, the results of our experiments led us select the following classifiers that we trained on the by-article training data:

- The best classifier using bag-of-words, which used prefixes (1-to-3/4) as features.
- The best classifier using bag-of-clusters which worked with the clusters `hyp-s300-w5-700`.
- The best classifier using word embeddings, which worked with pre-trained word embeddings trained on Common Crawl with FastText (`FT-Crawl`).

The highest performance among these classifiers was received by the classifier using prefixes (*accuracy = 0.7891*). The second best result was from the classifier using word embeddings, namely `FT-Crawl`. The lowest performance was obtained by the bag-of-clusters classifier, which scored an accuracy of 0.7629.

In addition, we continued experimenting with the following SVM classifiers when working with the by-publisher subset:

- The best classifier using bag-of-words, which used word uni-grams.
- The best classifier using bag-of-clusters, which is the classifier using the clusters `hyp-s300-w5-800`.

- The best classifier using word embeddings, which were the pre-trained word embeddings on Common Crawl from FastText, excluding stop words (`FT-Crawl-excl-stop-words`).

The best performance was from the bag-of-words classifier, which received an accuracy score of 0.9185 on cross-validation. The second best accuracy score was obtained by the classifier using word embedding (*accuracy = 0.8123*). The classifier that used bag-of-clusters yielded the lowest performance (*accuracy = 0.8123*).

Hence, on both datasets, the bag-of-words classifiers scored the best and bag-of-clusters the lowest when they were evaluated via 5-fold cross-validation. One one hand, this pattern seemed to be in contrast to what we expected, since it seemed more likely to use that the dense document representations would perform better than the sparse ones. One the other hand, we were surmising that the better performance of the sparse systems could be related to the fact that the systems were still evaluated on a test set derived from the same corpus as the training data. This implied that there was a low amount of out-of-vocabulary words in the test sets in the cross-validation procedure. Therefore, we hypothesized that the sparse representations would also receive better results than the dense representations in the in-domain evaluation procedure. However, we surmised that we would see the opposite pattern in the results of the evaluation procedure across-datasets.

6.2 Additional Features for SVM

Before we present the scores of the SVM classifiers in which we combined the best classifiers from the previous section with additional features, we show the results of the experiments in which we studied the individual contribution of the additional features. This will be done in the current section.

6.2.1 Linguistic Features

The results of the experiments with linguistic features (i.e., POS tag n-grams) are shown in Table 20. The tables reveal that POS n-grams were helpful features for hyperpartisan news detection. In particular, the classifiers showed a similar behavior as the ones that used bag-of-words as features. Moreover, POS uni-grams were the best linguistic features to predict hyperpartisan content in the by-article training data, whereas POS bi-grams worked better on the by-publisher subset. Because of these promising results, we decided to use linguistic features as the major method to improve the accuracy of classifiers by means of additional features.

POS sequence	Parameter settings	Accuracy
Uni-grams	Default	0.7830
Bi-grams	Tuned	0.7768
Tri-grams	Tuned	0.7349

(a) Results on the by-article training set.

POS sequence	Parameter settings	Accuracy
Bi-grams	Default	0.9188
Tri-grams	Tuned	0.9150
Uni-grams	Tuned	0.9088

(b) Results on the by-publisher subset.

Table 20: The Top 5 results on 5-fold cross-validation of the SVM classifers using POS features (linguistic features) with tf-idf weighting.

6.2.2 Sentiment Features

Among the sentiment features, the ones that we obtained with VADER were the most useful. Nonetheless, the results in Table 21 show that sentiment features had minimal predictive power when they are used without additional features (i.e., in isolation). However, at this stage of the experimental procedure, this did not rule out the possibility that they could improve the performance of other models. Hence, despite the low predictive power of the sentiment features on their own, we continued experimenting with sentiment features in the experiments where we trained SVM classifiers on document representations and additional features.

System	Accuracy
Negative sentiment	**0.6326**
Positive and negative sentiment	0.6326
Positive plus negative sentiment	0.6326
Compound score	0.6310
Positive sentiment	0.6348

(a) results on the by-article training set.

System	Accuracy
Positive and negative sentiment	**0.5769**
Positive sentiment	0.5703
Positive plus negative sentiment	0.5617
Compound score	0.5268
Negative sentiment	0.5105

(b) results on the by-publisher subset.

Table 21: The Top 5 most predictive sentiment features for the classifiers that we trained and evaluated via 5-fold cross-validation. All features were computed with VADER (Hutto and Gilbert, 2014).

6.2.3 Stylistic Features

The stylistic features are more effective on the by-article data than on the by-publisher subset (see Table 22). In particular, the average accuracy scores on by-publisher varied around 0.50, whereas they fluctuated around 0.63 on the by-article training data. Furthermore, we could not increase the predictive power of the readability features by combining the

different metrics. The generally low scores discouraged us to work with stylistic features in the classifiers that would combine the best classifiers of Section 6.1 and the additional features presented in this section.

System	Accuracy
Assertives	**0.6310**
Smog	0.6310
Automatic readability index	0.6295
Kinchaid grade	0.6279
All readability metrics	0.5833

(a) Results on the by-article training set.

System	Accuracy
Flesh Easy	**0.5039**
Smog	0.5035
Assertives	0.5035
All readability metrics	0.5015
Kinchaid grade	0.5012

(b) Results on the by-publisher subset.

Table 22: The Top 5 most predictive stylistic features for the classifiers that we trained and evaluated via 5-fold cross-validation. The usage of assertives was represented by binary representation.

6.3 Adding Local Features to Document Representations with SVM

In the previous section, we presented the individual performance of the local/additional features. In the current section, we describe the results of the experiments in which we tried to find the three best SVM classifiers that used the most promising local features and at least one of the three document representation techniques. However, as we mentioned in Chapter 5 (see Section 5.4), we also built classifiers that combined several document representation techniques. We will present the results of these experiments here as well.

6.3.1 Bag-of-words

We begin by showing the results of our experiments aimed at increasing the accuracy of the best bag-of-words model on the by-article training set in Table 23. We started by adding the best linguistic feature from Table 20a, which was POS uni-grams. This addition increased the performance of the bag-of-prefixes classifier from 0.7891 to 0.7924. Afterwards, we tried to increase the performance by adding the sentiment feature with the highest accuracy from Table 21a. However, this was decremental to the accuracy of the classifier that used prefixes and POS uni-grams. As we did not obtain promising results with the stylistic features, we decided to add the best bag-of-clusters features `hyp-s300-w5-700`. This procedure neither outperformed the results of the classifier using POS uni-grams and prefixes. Thus, in the end, the best classifier was the one that used prefixes and POS uni-grams.

The results of the classifiers that we trained on the by-publisher subset are depicted in Table 24. Since we obtained accuracy scores around 0.90 on the by-publisher subset with the classifiers using bag-of-words, we only conducted two experiments to optimize

System	Accuracy
Prefixes (1-to-3/4)	0.7891
Prefixes (1-to-3/4) + POS uni-grams	**0.7924**
Prefixes (1-to-3/4) + POS uni-grams + negative sentiment	0.7908
Prefixes (1-to-3/4) + POS uni-grams + hyp-s300-w5-700	0.7877
Prefixes (3-to-4) + POS uni-grams + hyp-s300-w5-700 + negative sentiment	0.7861

Table 23: The results of the models that use prefixes and additional features on 5-fold cross-validation using the by-article training data.

the performance of the classifier. In the first, we tried to increase the performance of the SVM classifier by adding the sentiment feature with the highest accuracy from the cross-validation experiments (see Table 21b). Similarly, we added the best linguistic feature from Table 20b in our second experiment. The latter resulted to the highest accuracy score on cross-validation (*accuracy=0.9261*). Hence, this was the best model that relied on bag-of-words plus additional features. Therefore, we continued to evaluate this model on the by-article (across-datasets) and by-publisher test set (in-domain).

System	Accuracy
Word uni-grams	0.9185
Word uni-grams + positive + negative sentiment	0.9190
Word uni-grams + POS uni-grams	**0.9261**

Table 24: The results of the models that use prefixes and additional features on 5-fold cross-validation using the by-publisher subset.

6.3.2 Bag-of-clusters

The performance of the classifiers on the by-article training data are shown in Table 25. The highest accuracy score was achieved by the model that combined several types of features: prefixes, POS uni-grams, bag-of-clusters (using the clusters `hyp-s300-w5-700`) and the negative sentiment of a text. With this model, we were able to score an accuracy of 0.7954 on 5-fold cross-validation. Thus, we decided to use this classifier for further evaluation in the next chapters.

In the next table, we show the results of our experiments conducted on the by-publisher subset. When we added POS bi-grams to the clusters, we increased the accuracy from 0.8556 to 0.9178. We also tried to increase the results by adding the best sentiment features on the by-publisher subset, which were the positive and the negative sentiment. However, this classifier did not outperform the current best score of 0.9178. Since the results of the stylistic features were unpromising, we decided to experiment with the best bag-of-word features from Table 24. This increased the accuracy to 0.9194. Thus, the best

System	Accuracy
Hyp-s300-w5-700 +	0.7645
Hyp-s300-w5-700 + POS uni-grams	0.7846
Hyp-s300-w5-700 + POS uni-grams+ prefixes + negative sentiment	**0.7954**
Hyp-s300-w5-700 + character 3-to-5 grams	0.7815

Table 25: The results of the models that use additional features and bag-of-clusters on 5-fold cross-validation on the by-article training set.

model was the classifier using bag-of-clusters (using the cluster `hyp-s300-w5-800`) and bag-of-word uni-grams. Consequently, we used this classifier for evaluation on the test sets of the competition.

System	Accuracy
Hyp-s300-w5-800	0.8556
Hyp-s300-w5-800 + POS bi-grams	0.9178
Hyp-s300-w5-800 + word uni-grams	**0.9194**
Hyp-s300-w5-800 + positive + negative sentiment	0.8560

Table 26: The results of the models that use additional features and bag-of-clusters on 5-fold cross-validation on the by-publisher subset.

6.3.3 Word Embeddings

The results of the experiments in which we tried to find the best classifier that uses word embeddings and additional features are presented in Tables 27 and 28. We show the results on the by-article training data in the former, and the results on the by-publisher subset in the latter.

We used two ways to improve the results of the classifier using `FT-crawl` and the by-article training set. The first was by adding the features of the best classifier that relied on either one of the three document representation techniques. These features were prefixes. The second method was by adding the best linguistic feature from Table 20. As the latter decreased the results, we discontinued using them in the next experiments. Instead, we tried to increase the performance of the classifier using `FT-crawl` features and prefixes by adding the negative sentiment as features. Even though the negative sentiment was the best feature from Table 21, we did not manage to increase the performance in the current experiment. We were also unable to improve the results of the classifier using `FT-crawl` by combining it with the best cluster features from the earlier experiments.

Because of this, the best result from Table 27 is derived from the classifier that uses two document representation techniques: word embeddings and bag-of-prefixes. With

these features, we were able to obtain an accuracy of 0.8093 on 5-fold cross-validation. This is also the highest accuracy that we obtained in general on the by-article training set.

System	Accuracy
FT-crawl	0.7754
FT-crawl + prefixes (1-to-3/4)	**0.8093**
FT-crawl + POS uni-grams	0.7909
FT-crawl + prefixes (1-to-3/4) + negative sentiment	0.8093
FT-crawl + hyp-s300-w5-700	0.7831
FT-crawl + word uni-grams + hyp-s300-w5-800	0.7985

Table 27: The scores on 5-fold cross-validation for models that use FastText pre-trained word embeddings and additional features trained and evaluated on by-article training.

Furthermore, the results in Table 28 show that the performance of our best classifier using word embeddings (`FT-crawl-excl-stop-words`) could be improved by adding word uni-grams. In particular, we were able to increase the accuracy from 0.8125 to 0.9206. We initially decided to experiment with word uni-grams because it was used in the best bag-of-words classifier. In contrast, we subsequently experimented with prefixes in another experiment because they proved to be useful on the other dataset and the third best bag-of-words features on the by-publisher subset. However, the prefixes did not increase the performance as much as the word uni-grams. We finally attempted to optimize the performance of the classifier using `FT-crawl-excl-stop-words` and word uni-grams by adding both the positive and the negative sentiment as features. These sentiment features achieved the best score when they were used in isolation on 5-fold cross-validation (see Table 21). Nonetheless, this only increased the accuracy slightly. Hence, the best classifier of the experiments with additional features and `FT-crawl-excl-stop-words` was the one that we combined with word uni-grams.

6.4 Experiments with Recurrent Neural Networks

We show the results of the experiments that we conducted with the Flair embeddings in Table 29. It is important to recall that we used a different procedure for feature selection than the SVM classifiers (see Section 5.5) and that there are two types of Flair embeddings: backward and forward embeddings (see Chapter 4).

The results in the table show that we did not manage to obtain promising results with the neural networks. In particular, we were unable to obtain high results with the set-ups that we initially used (described in Section 5.5.1), The results of these set-ups are presented by the results of the first three classifiers in Table 29. Besides, the results reveal that the FastText word embeddings did not yield high results when they were used with Recurrent Neural Networks. The same can be said about the classifier that relied on FastText word embeddings and Flair embeddings. Furthermore, even the classifier that used character embeddings and Flair embeddings received only a micro accuracy of 0.5422. This result

System	Accuracy
FT-crawl-excl-stop-words	0.8125
FT-crawl-excl-stop-words + prefixes (1-to-3/4)	0.9063
FT-crawl-excl-stop-words + word uni-grams	0.9206
FT-crawl-excl-stop-words + word uni-grams + positive + negative sentiment	**0.9208**

Table 28: The scores on 5-fold cross-validation for models that use FastText pre-trained word embeddings (without stop words) and additional features trained and evaluated on the by-publisher subset.

is lower than the classifier which used FastText embeddings and the two flair embeddings (*micro-accuracy=0.5802*).

Because of the results of our initial set-ups, we decided to experiment with other pre-trained word embeddings, such as the ones from GloVe. When we experimented with GloVe, we obtained higher results than with FastText. The highest accuracy is 0.4828, which was from the system using GloVe embeddings and the character embeddings of Lample et al. (2016). Furthermore, the lowest accuracy score was received by the classifier that only used the embeddings from FastText. Recall that we obtained the best results with these word embeddings in SVM (see Section 6.1.3).

Features	Precision	Recall	F1-score	Accuracy	Micro Accuracy
FT-Crawl WE	0.6667	0.5000	0.5714	0.4000	0.5610
FT-Crawl WE + Flair (backward) + Flair (forward)	0.6842	0.5417	0.6047	0.4333	0.5802
CharacterEmbeddings (Lample et al., 2016) + Flair (backward) + Flair (forward)	0.6190	0.5417	0.5578	0.4062	0.5422
GloVe 840B.300D + CharacterEmbeddings (Lample et al., 2016)	**0.7368**	**0.5833**	**0.6511**	**0.4828**	**0.6203**
GloVe 840B.300D + Flair (backward) + Flair (forward)	0.7000	**0.5833**	0.6363	0.4667	0.6000

Table 29: The performance of the classifiers that use recurrent neural networks on evaluated on 10% of the by-article training data and trained on 80% of the by-article training set.

6.4.1 Best Classifier

Despite the low results, we decided to use the second best model of Table 29 for further evaluation. According to the micro accuracy, the second best classifier used GloVe embeddings and the two Flair embeddings. However, we mainly decided to continue experimenting with this classifier because we were interested in the performance of the Flair embeddings combined with one of the pre-trained word embeddings that we evaluated in SVMs. Because of the low results with the FastText embeddings, we decided to use the GloVe embeddings for this purpose.

We trained the classifier that used GloVe embeddings plus the FastText embeddings for four other times and selected the model with the highest micro accuracy over the five runs for further evaluation. We show the best results of this procedure in Table 30. The performance of this classifier is similar to the best classifier that we presented in Table 29. In addition, we visualize the performance of this classifier over 120 epochs in Figure 6. The graphs show that the accuracy and F1-score on the development set increased over the amount of epochs. Nonetheless, this pattern is not visible in the most upper graph, which indicates that the loss on the training set slowly increased over epochs. Ideally, we would have seen the opposite pattern. Despite the unpromising results, we decided to evaluate this classifier on the by-article test set (in-domain evaluation) and on the by-publisher test set (evaluation across-datasets).

Metric	**Score**
Precision	0.7000
Recall	0.5033
F1-score	0.6365
Accuracy	0.4667
Micro accuracy	0.6363

Table 30: The performance of the best classifier over five runs that use recurrent neural networks on predicting hyperpartisan evaluated on 10% of the by-article training data and trained on 80% of the by-article training set.

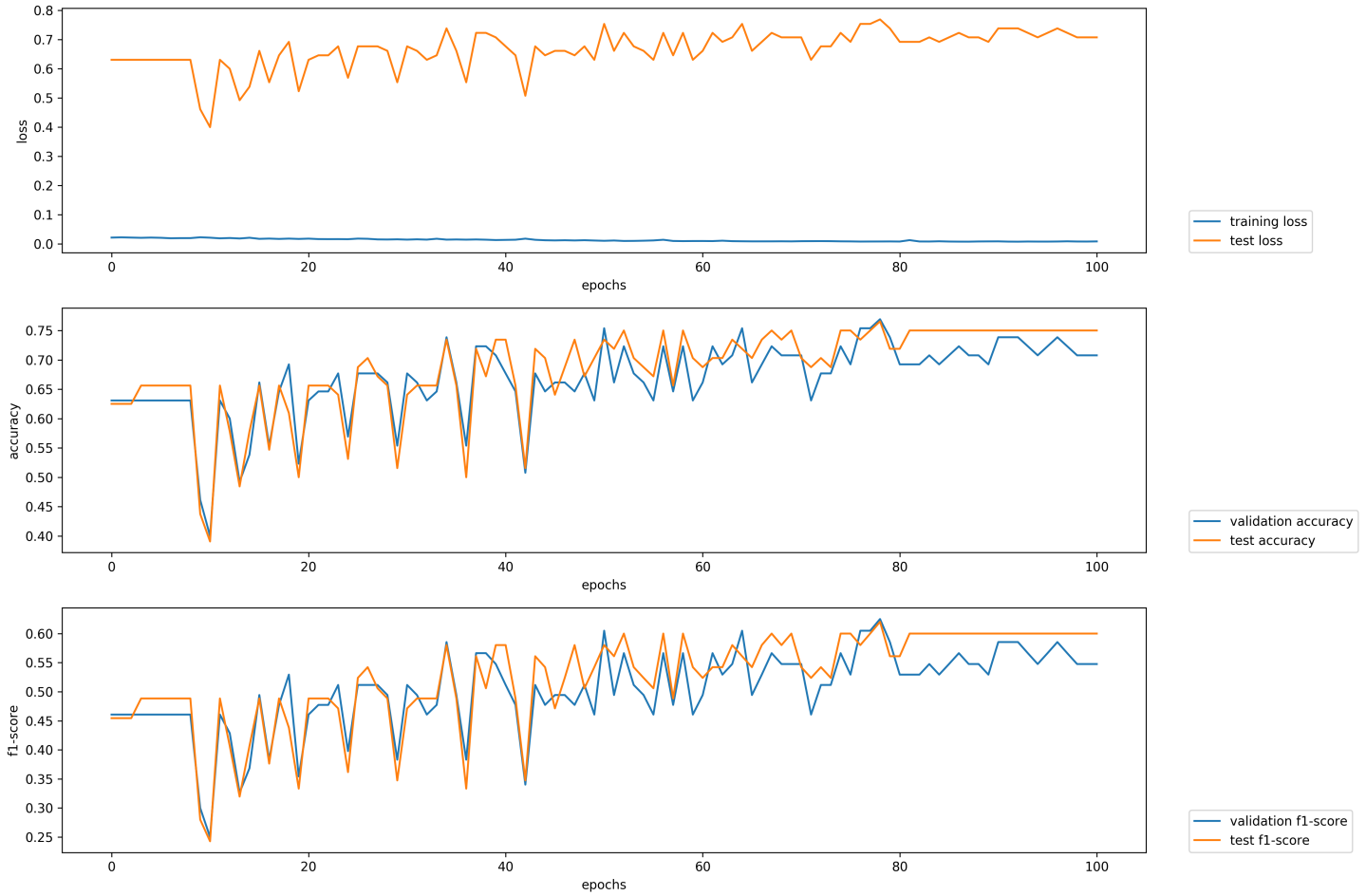


Figure 6: Three graphs showing the performance of the best classifier with neural networks over 120 epochs.

6.5 Summary

In this chapter, we showed that it is possible to build accurate classifiers for hyperpartisan news detection with SVMs. In particular, we were able to obtain promising results with bag-of-words, bag-of-clusters and word embeddings. This was the case for both the by-article training set and the by-publisher subset. We therefore selected the best bag-of-words, bag-of-clusters and word embeddings classifier of each dataset for further evaluation. We also managed to increase the performance of the best bag-of-words, bag-of-clusters and word embeddings classifier by combining each classifier with other features. Despite the discouraging results that we obtained with Recurrent Neural Networks, we decided to continue evaluating the best classifier of these experiments to gain more knowledge about their performance for hyperpartisan news detection.

7 In-domain Results

This chapter shows the results of the best classifiers for each classification approach in an in-domain setting. We present the results of the SVM classifiers in Section 7.1. We also describe the results of the Recurrent Neural Networks in Section 7.2. We finish this chapter by presenting the classifier with the highest result on the by-article test set, which was used to determine the rankings in the shared task on hyperpartisan news detection.

7.1 Results with Support Vector Machines

This section is divided into two parts. In the first part, we show the performance of the models that we trained on the by-article training set and evaluated on by-article test set. In the second part of this section, we present the results of the classifiers that we trained on the by-publisher subset and evaluated on the by-publisher test set.

7.1.1 Results on the By-article Test Set

We show two types of results in Table 31. The first are the results from the three classifiers that we trained on any of the three document representations: bag-of-words using prefixes, bag-of-clusters (`hyp-s300-w5-700`) or word embeddings (`FT-crawl`). The second type of results is from the classifiers that use the document representations and additional features. These results are from the best classifiers of Section 6.3, where we combined document representations with additional features via 5-fold cross-validation. We continued experimenting with the best classifiers from each of the following tables of Section 6.3: Tables 23, 25, and 27.

In general, the results in Table 31 show that we managed to obtain better results with the presented classifiers than the baseline, which scored an accuracy of 0.621 on the by-article test set (see Table 11). In particular, we already outperformed the accuracy of the baseline with the systems that only used one of the document representation techniques. Nonetheless, the best result was obtained by the classifier that combines several features, namely: bag-of-clusters (using `hyp-s300-w5-700`), bag-of-prefixes (1-to-3/4), POS unigrams and the negative sentiment. This classifier received an accuracy score of 0.7850 on the test set.

The results also show that combining features generally improved the performance of the classifiers on the test set. The three best accuracy scores are namely received by the classifiers that used a combination of features. This is similar to the results on 5-fold cross-validation, where adding features generally increased the performance of the SVMs (see Section 6.3). However, the improvements were only minor, since the accuracy scores came close to 0.80, but never exceeded it.

In addition, the results of the classifiers using either bag-of-prefixes, bag-of-clusters (using `hyp-s300-w5-700`) or word embeddings (`FT-crawl`) show that the highest accuracy score was received by the classifier using prefixes. We namely achieved an accuracy score of 0.7800. It also received the highest precision of all classifiers in Table

31. This is in parallel to the results that we obtained on 5-fold cross-validation. In particular, the sequence of the best to least accurate classifier is: prefixes, `FT-CRAWL` and `hyp-s300-w5-700`. The same pattern is visible in Table 31. Besides, there was only a slight difference between the accuracy on 5-fold cross-validation and the test set (see Section 6.1.4). For instance, the SVM classifier that used prefixes as features received an accuracy score of 0.7891 on 5-fold cross-validation and 0.7803 on the by-article test set.

In any case, it is disappointing that the denser representation techniques (bag-of-clusters and word embeddings) did not obtain better results when they were used in isolation. However, we surmised in the previous chapter that this pattern could appear in the in-domain evaluation, similar as in the results on cross-validation. We mentioned the few amount of out-of-vocabulary words as a possible reason.

System	**Accuracy**	**Precision**	**Recall**	**F1-score**
Prefixes (1-to-3/4)	0.7800	**0.8121**	0.7293	0.7685
Prefixes (1-to-3/4) + word uni-grams	0.7802	0.8014	0.7452	0.7722
Hyp-s300-w5-700	0.7500	**0.7716**	0.7102	0.7396
Hyp-s300-w5-700 + negative sentiment + prefixes (1-to-3/4) + POS uni-grams	**0.7850**	0.7993	**0.7611**	**0.7789**
FT-crawl	0.7548	0.7941	0.6879	0.7372
FT-crawl + prefixes (3-to-4) + negative sentiment	0.7818	0.8062	0.7420	0.7728

Table 31: The performance of the best SVM classifiers trained on the by-article training set and evaluated on the by-article test set.

7.1.2 Results on the By-publisher Test Set

We show the results on the by-publisher test set in Table 32. We present the results in the same way as in Table 31. Thus, we show the results of the classifiers using either one of the three document representations and the results of the classifiers that use a combination of features. The latter consists of the best classifier of each of the following tables: Tables 24, 26 and 28.

The results in Table 32 show that all classifiers outperformed the baseline, which scored an accuracy of 0.589 on the by-publisher test set (see Table 11). Moreover, the best score was achieved by the classifier that only relied on bag-of-word uni-grams. However, this score differs substantially with the result obtained via 5-fold cross-validation. In particular, the classifier received an accuracy of 0.6525 on the test set (see Table 32), but 0.9185 on cross-validation (see Table 14). There is a similar gap visible between the results on 5-fold cross-validation and the test set of the other classifiers in Table 32. For instance, the classifier that only used bag-of-clusters as features received an accuracy score of 0.8123 on 5-fold cross-validation, but 0.6218 on the test set. The only similarity between the scores in our preliminary experiments and the test set was the ranking of the classifiers'

performance. The best model was namely the classifier using bag-of-words, the second best bag-of-clusters and the worst word embeddings.

In addition, the results show that we were unable to improve the performance of the best classifier by adding linguistic features. In contrast, we were able to increase the performance of both the classifier using bag-of-clusters and the classifier using word embeddings. In particular, the accuracy of the classifier using `FT-crawl-excl-stopwords` was 0.6010, and it was increased to 0.6420 by adding bag-of-word uni-grams and sentiment features. However, none of the other classifiers outperformed the score of the classifier that only used word uni-grams.

Finally, comparing the results in Table 32 to the results on the by-article test set (see Table 31) reveals that we obtained higher results on the by-article test set than on the by-publisher test set. In particular, the scores on the by-article test set varied between 0.7803 and 0.7850, whereas the scores on the other test set varied between 0.6010 and 0.6525. Nonetheless, the performance of the systems that the participants submitted in the shared task on the by-publisher test set shows that we obtained reasonably fair results on this test set. In the competition, the highest accuracy score on the by-publisher test set was 0.706. Besides, if we would have evaluated the best system on the by-publisher test set during the competition, we would have obtained the 5th out of 28th position.

System	**Accuracy**	**Precision**	**Recall**	**F1-score**
Word uni-grams	**0.6525**	**0.6346**	0.7190	**0.6742**
Word uni-grams + POS bi-grams	0.6388	0.6225	0.7050	0.6612
Hyp-s300-w5-800	0.6218	0.6084	0.6835	0.6437
Hyp-s300-w5-800 + word uni-grams	0.6418	0.6280	0.6955	0.6600
FT-crawl-excl-stopwords	0.6010	0.5790	**0.7420**	0.6504
FT-crawl-excl-stopwords + word uni-grams + positive + negative sentiment	0.6420	0.6310	0.6840	0.6564

Table 32: The performance of the best bag-of-words, bag-of-clusters and word embedding model trained on by-publisher training and evaluated on the by-publisher test set.

7.2 Results with Recurrent Neural Networks

In Table 33, we describe the scores of the classifier using Recurrent Neural Networks and Flair embeddings. The results show that we received a generally low performance on the by-article test set. More specifically, we received an accuracy score of 0.5892, which did not outperform the performance of the baseline. Furthermore, the classifier seemed to randomly predict the documents. However, the results are approximately similar to the ones that we obtained in the previous evaluation set-up (see Table 29). More specifically, when we evaluated the system presented in Table 33 on the test set that we derived from the by-article training set, we received an accuracy score of 0.4600. The micro accuracy

score of this system was 0.6000. Thus, the results were not contradictory to the expected results that we had based on the previous results.

Metric	Score
Accuracy	0.5892
Precision	0.5848
Recall	0.6146
F1-score	0.5994

(a) Evaluation scores.

	True	False
True	193	121
False	137	177

(b) Matrix showing predictions (columns) and gold standard (rows).

Table 33: The results on the by-article test set of the classifier using Flair and GloVe embeddings using Recurrent Neural Networks.

7.3 Best Classifier

After evaluating the best SVM and neural network classifiers from Chapter 6 on the test sets, we took the classifier with the highest accuracy on the by-article test set for further improvement. This was the best SVM classifier presented in Table 31. It used the following features: the clusters `hyp-s300-w5-700`, POS uni-grams, prefixes (1-to-3/4) and the negative sentiment. Furthermore, this classifier received an accuracy score of 0.7954 on 5-fold cross-validation (see Table 25). Even though it did not obtain the best score on 5-fold cross-validation, it managed to achieve the best score on the by-article test set among all classifiers.

We conducted three additional experiments where we trained SVM classifiers on the by-article test set through 5-fold cross-validation while using the same features of the best classifier on the by-article test set. In these experiments, we tried to increase the performance of our best classifier by applying pre-processing techniques. In the first experiment, we removed the stop words of the texts. In the second experiment, we balanced the labels of the dataset[13]. Finally, we concatenated the title to the body of the text, similar as in the SVM classifiers from Singhania et al. (2017).

We show the results of these techniques on 5-fold cross-validation in Table 34. The best result was achieved by the classifier where we removed the stop words of the documents. In particular, we managed to increase the accuracy score on 5-fold cross-validation from 0.7954 to 0.8017. We also improved the results on the by-article test set with this classifier, which we show in Table 35. As shown in the table, the classifier scored an accuracy of 0.7866, which is close to the performance of the winning system in the shared task. In particular, if we would have submitted this system in the competition, then we would have obtained the 8th place out of 42 participants. In addition, this system substantially outperforms the baseline that we used in this project, which received an accuracy of 0.621.

[13]using stratified sampling from Sklearn, `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

System	Accuracy
Best classifier	0.7954
Best classifier + stop word removal	**0.8017**
Best classifier + balanced labels in training data	0.7905
Best classifier + concatenate title to body	0.7955

Table 34: Results the by-article training data on 5-fold cross-validation showing the effects of the pre-processing procedures of the best classifier on the by-article test set.

	Accuracy	Precision	Recall	F1-score
Winning system in competition	*0.8220*	*0.8710*	*0.7555*	*0.8090*
Baseline	0.6210	0.5820	0.860	0.6940
Best classifier	0.7850	0.7993	0.7611	0.7789
Best classifier without stopwords	**0.7866**	**0.8041**	**0.7580**	**0.7803**

Table 35: Results of the previous best classifier and the current best classifier on the by-article test set. We also show the results of the winning system in the competition.

7.3.1 Summary

Our best classifier on by-article test set received an accuracy score of 0.7866. This SVM classifier relied on bag-of-clusters (using the clusters `hyp-s300-w5-700`), prefixes, POS uni-grams and the negative sentiment of the text. Nonetheless, we were already able to obtain a good performance on the by-article test set by building classifiers that used either one of the three document representations: bag-of-prefixes, bag-of-clusters or word embeddings. We managed to increase the performance of these classifiers by combining features, but only slightly. The results on the by-publisher test set remained relatively low, despite the size of the by-publisher subset, which is much larger than the by-article training set. Finally, we were unable to receive similar good results with the Recurrent Neural Networks, which did not even exceed the scores of the baseline.

8 Across-datasets and Out-of-domain Evaluation

In this chapter, we present the results of two types of evaluation settings. First, across-datasets, namely, training on by-article and evaluating on by-publisher and vice versa. The second setting is out-of-domain, for which we use the best features from the classifier with the highest results on the by-article test set, without any specific tuning, to perform the out-of-domain evaluation on two fake news datasets as previously performed by Pérez-Rosas et al. (2018).

8.1 Across-datasets Evaluation on Hyperpartisan using SVM

In this section, we present the results of the best bag-of-words, bag-of-clusters and word embeddings classifiers across-datasets. We also show the results of the SVM classifiers using at least one of the three document representations and additional features. We present the results of the classifiers that we trained on the by-article training set in Section 8.1.1 and on the by-publisher subset in Section 8.1.2.

8.1.1 Training on By-article

We show the results of the classifiers that we trained on the by-article training set and evaluated on the by-publisher test set in Table 36. In the first row of the table, we present the scores of the classifier with the highest accuracy score on the by-article test set via in-domain evaluation. Thus, this was the best classifier that we trained on the by-article training data and evaluated on the by-article test set. This classifier received an accuracy score of 0.7866 on the by-article test set, but an accuracy of 0.5458 on the by-publisher test set (i.e., across-datasets). Hence, the classifier with the highest performance on the by-article test set did not obtain the best result on the by-publisher test set.

Instead, the highest accuracy (*accuracy=0.5820*) on the by-publisher test set was achieved by the classifier that only used word-embeddings (`FT-crawl`). The lowest accuracy score was received by the classifier using prefixes (1-to-3/4) and bag-of-word uni-grams (*accuracy=0.5330*). Furthermore, the classifier that only used bag-of-clusters received a higher accuracy (*accuracy=0.5648)* on the test set than the classifier using prefixes (*accuracy=0.5353*). These results confirmed our theory that denser representations would yield better performance across-datasets than in-domain.

Moreover, the results in Table 36 show that adding features did not increase performance of the classifiers. For instance, the classifier that only used prefixes scored an accuracy of 0.5353, whereas the classifier that used prefixes and bag-of-word uni-grams received an accuracy of 0.5330. A similar effect is visible for the the classifier using clusters and the word embeddings classifier. However, the results of the in-domain and cross-validation experiments show the opposite, since adding features generally improved the accuracy in that set-up.

In conclusion, we received the best accuracy score in this set-up when using the most dense document representations without additional features. Adding features only de-

creased the performance across-datasets when we trained systems on the by-article training set and evaluated them on the by-publisher test set. Furthermore, we generally received low results on the by-publisher test set with the systems that we trained on the by-article training set. However, these results are fair when we compare them to the results that we achieved in our in-domain evaluation set-up. As mentioned in the previous chapter, the best classifier that we trained on the by-publisher subset and evaluated on the by-publisher test set received an accuracy of about 0.65.

System	**Accuracy**	**Precision**	**Recall**	**F1-score**
Hyp-s300-w5-700 + POS bi-grams + prefixes + (1-to-3/4) + negative sentiment + stopword removal	*0.5458*	*0.5604*	*0.4245*	*0.4831*
Prefixes (1-to-3/4)	0.5353	0.5457	0.4210	0.4753
Prefixes (1-to-3/4) + bag-of-word uni-grams	0.5330	0.5402	0.4435	0.4871
Hyp-s300-w5-700	0.5648	0.5979	0.3965	0.4767
Hyp-s300-w5-700 + POS bi-grams + prefixes (1-to-3/4) + negative sentiment	0.5485	0.5637	0.4290	0.4872
FT-crawl	**0.5820**	**0.6137**	**0.4425**	**0.5142**
FT-crawl + prefixes (1-to-3/4) + negative sentiment	0.5560	0.5734	0.4375	0.4963

Table 36: The results of the best SVM classifiers trained on the by-article training set and evaluated on the by-publisher test set. The first row represents the scores of best system on the by-article test set from the in-domain evaluation experiments, of which we show the performance on the by-publisher test set in this table.

8.1.2 Training on By-publisher

We present the results of the classifiers that we trained on the by-publisher subset and evaluated on the by-article test set in Table 37. The best result was scored by the classifier that only used word embeddings (`FT-crawl-excl-stopwords`). On one hand, this was another confirmation that denser word representations perform better across-data. On the other hand, the classifier that only used bag-of-clusters had a lower performance (*accuracy=0.6369*) than the classifier that only used bag-of-words (*accuracy=0.6417*).

Similar to the results in Table 36, adding features did not increase the performance on the test set. For example, the lowest accuracy score was achieved by the classifier that used `FT-crawl-excl-stopwords` and additional features. However, one main difference between the results is that the classifiers in Table 37 received a high F1-score, which varied between 0.6953 and 0.7066. The F1-scores of the results across-datasets on the by-publisher test set however, varied between 0.4767 and 0.5142. The higher F1-score is related to the substantially higher recall of the classifiers in Table 37. Thus, despite

the large size of the by-publisher subset compared to the by-article training set, we were unable to obtain similar scores on the by-article test set as we did with by-article training set.

System	Accuracy	Precision	Recall	F1-score
Bag-of-word uni-grams	0.6417	0.5982	0.8631	0.7066
Bag-of-word uni-grams + POS bi-grams	0.6369	0.5931	0.8726	0.7062
Hyp-s300-w5-800	0.6354	0.5934	0.8599	0.7022
Hyp-s300-w5-800 + bag-of-word uni-grams	0.6354	0.5938	0.8567	0.7014
FT-crawl-excl-stopwords	**0.6465**	**0.5991**	**0.8854**	**0.7147**
FT-crawl-excl-stopwords + bag-of-word uni-grams + negative + positive sentiment	0.6274	0.5881	0.8503	0.6953

Table 37: The results of the best SVM classifiers trained on the by-publisher subset and evaluated on the by-article test set.

8.2 Across-datasets Evaluation on Hyperpartisan using Neural Networks

We show the results of the classifier using Recurrent Neural Networks in Table 38. Recall that this classifier used the Flair embeddings (both forward and backward) and GloVe embeddings (`840B.300D`). The scores that the classifier obtained on the by-publisher test set were slightly lower than the scores that it received in the in-domain evaluation procedure (see Table 33). In particular, the classifier scored an accuracy of 0.5458 on the by-publisher test set. The results also show that the classifier randomly classified the documents, similar as in the in-domain evaluation. This is particularly visible in the confusion matrix. Because of these results, we concluded that the Flair embeddings and the Recurrent Neural Networks were unable to perform well in general, since the baseline outperformed the results of the neural networks in both evaluations. We will pinpoint a few of the possible reasons in Chapter 9.

Metric	Scores
Accuracy	0.5425
Precision	0.5478
Recall	0.4870
F1-score	0.5156

(a) Evaluation scores.

	True	False
True	974	804
False	1026	1196

(b) matrix showing predictions (columns) and gold standard (rows).

Table 38: The across-datasets performance of the best neural network classifier using the Flair and GloVe embeddings. The classifier was trained on the by-article training set and evaluated on the by-publisher test set.

8.3 Out-of-domain Evaluation on Fake News

In this section, we present the results of three classifiers that we evaluated through two different set-ups using the two datasets of Pérez-Rosas et al. (2018): the FakeNewsAMT and CelebrityNews dataset. In the first set-up, we trained the classifiers via 5-fold cross-validation on the FakeNewsAMT dataset. In the second set-up, we trained these classifiers on the FakeNewsAMT dataset and evaluated them on the CelebrityNews dataset. Thus, we tried to reproduce the experimental set-up described in Pérez-Rosas et al. (2018) (see Chapter 2). In this experiment, the authors performed an across-domain evaluation by training their classifiers on the FakeNewsAMT dataset and evaluating them on the CelebrityNews dataset, and vice versa. However, in our experiment, we used our best classifiers without performing specific feature tuning for fake news detection, this set-up allowed us to investigate the robustness of our features.

In particular, we worked with the features that were used by the best bag-of-words (prefixes), bag-of-clusters (`hyp-s300-w5-700`) and word embeddings (`FT-crawl`) classifiers which we selected based on the results on the by-article test set (see Table 31). We decided use these classifiers instead using their alternative versions with additional features. The reason was that the results of the experiments across-datasets in Section 8.1 (see Table 36 and 36) showed that adding features generally decreases the performance of the classifiers. It should also be noted that we decided to work with the systems and features that we trained on the by-article training data because of the substantially lower performance of the systems that we trained on the by-publisher subset on other settings than 5-fold cross-validation.

8.3.1 Results

We show the results of the classifiers evaluated on 5-fold cross-validation in Table 39. The table shows that we obtained the best result with the classifier using word embeddings (`FT-crawl`) (*accuracy=0.638*). Moreover, the accuracy score of the classifier using prefixes was 0.5120, which was slightly higher than the classifier that relies on clusters. This

classifier scored an accuracy of 0.5104.

System	Accuracy
Prefixes	0.5104
Hyp-s300-w5-700	0.5120
FT-Crawl	**0.6438**

Table 39: The results on 5-fold cross-validation of the classifiers trained on the FakeNewsAMT dataset of Pérez-Rosas et al. (2018). Each classifier used either one of the best features from the best bag-of-words, bag-of-clusters or word embeddings classifier that we developed for fake news detection.

We show the accuracy scores of the classifiers that we trained on FakeNewsAMT and evaluated on CelebrityNews in Table 40. We additionally show the results of the classifiers from Pérez-Rosas et al. (2018) using the same experimental set-up. We show a more detailed overview of the performance of our systems on the FakeNews dataset in Table 41.

System	Training	Testing	Features	Accuracy
Pérez-Rosas et al. (2018)	FakeNewsAMT	CelebrityNews	LIWC	0.480
			Readability	**0.520**
			All features	0.500
Our work	FakeNewsAMT	CelebrityNews	Prefixes (1-to-3/4)	0.608
			hyp-s300-w5-700	**0.632**
			FT-Crawl	0.588

Table 40: Out-of-domain evaluation on the CelebrityNews dataset of Pérez-Rosas et al. (2018).

Features	Class	Precision	Recall	F1-score	Accuracy
Prefixes (1-3/4)	Fake	0.60	0.67	0.63	0.608
	Real	0.62	**0.54**	**0.58**	
Hyp-s300-w5-700	Fake	**0.61**	**0.75**	**0.67**	**0.632**
	Real	**0.67**	0.52	0.58	
FT-crawl	Fake	0.57	0.73	0.64	0.588
	Real	0.62	0.44	0.52	

Table 41: Results of the SVM classifiers using the best document representations of our systems on the by-article test set. The systems are trained on the FakeNewsAMT data and evaluated on the CelebrityNews dataset from Pérez-Rosas et al. (2018).

The results show that all three classifiers outperformed the results of Pérez-Rosas et al. (2018). Their best system, which used readability features, received an accuracy score of 0.520. Our best classifier scored an accuracy score of 0.632 on the CelebrityNews dataset

and the classifier with the lowest performance received an accuracy of 0.588. The classifier with the highest performance relied on clusters (`hyp-s300-w5-700`), whereas the classifier with the lowest results used word embeddings `FT-crawl`. The latter is surprising, since we achieved the best results with this system when we evaluated it via 5-fold cross-validation (see Table 39). However, the classifier using clusters (*accuracy=0.608*) performed better than the prefixes (*accuracy=0.632*), which suggests that denser representations perform better out-of-domain.

Based on these results, we concluded that our features yielded more robust systems than local features, which were mainly used in the systems of Pérez-Rosas et al. (2018). All our SVM systems, especially those that did not use local features, outperformed the SVM classifiers of Pérez-Rosas et al. (2018). Besides, the results of the experiments across-datasets showed that local features decreased the performance. Thus, local features decrease the performance vastly when applied across-datasets and in out-of-domain settings.

8.4 Summary

In this chapter, we presented two types of results. We started by presenting the performance of our systems across-datasets. We evaluated the classifiers that we trained on the by-article training set on the by-publisher test set, and the classifiers that we trained on the by-publisher subset on the by-article test set. In both evaluations, we obtained the best results with the classifiers that only used word embeddings. In particular, these were the word embeddings from FastText, trained on Common Crawl (`FT-crawl` and `FT-crawl-excl-stopwords`). The results also showed that adding local features decreases the performance of the classifiers when they are evaluated across-datasets. This was apparent on both evaluations.

We also presented the results of the experiments in which we used the settings of the best bag-of-words, bag-of-clusters and word embeddings classifiers without additional features for fake news detection. In one experiment, we evaluated these systems on 5-fold cross-validation using only one of the datasets from Pérez-Rosas et al. (2018). We achieved the best results with the classifier that used the most dense vector document representations (i.e., word embeddings). In the other experiment, we trained the same systems from the first experiment on one dataset of Pérez-Rosas et al. (2018) and evaluated it on the other. With our systems, we were able to outperform the results of Pérez-Rosas et al. (2018), whose systems only relied local features. This led us conclude that local features are detrimental to the performance of classifiers in out-of-domain evaluations. Finally, since the prefixes received the lowest performance, we concluded that denser document representations perform better out-of-domain than sparser document representations.

9 Discussion

In this chapter, we provide an analysis of the results that we obtained in our experiments. First, we discuss the results and features of our best classifier in Section 9.1. Then, we analyze the results of the in-domain and across-datasets evaluations, in Sections 9.2 and 9.3 respectively. We finish this chapter by discussing the out-of-domain results on fake news in Section 9.4.

9.1 Best Classifier

With our best classifier, were able to reach an accuracy of 0.7866 on the official test set of the shared task on hyperpartisan news detection. This system performed close to the winning system of the competition, even though it was simple to implement. If we would have submitted this model during the evaluation procedure of the competition, then we would have obtained the 8th out of 42th position in the competition. The classifier used SVM and relied on a combination of features: bag-of-clusters using the clusters `hyp-s300-w5-700`, bag-of-prefixes (1-to-3/4), POS uni-grams and the negative sentiment. The classifier was trained on a training set derived from the same corpus as the official test set, which consisted of manually annotated data.

In the rest of this section, we provide more information about our best classifier. We divide the section into two parts. In the first, we provide more information about our best classifier by discussing the confusion matrix. In the second part, we discuss the most important features that the classifier used.

9.1.1 Common Confusions

We show the confusion matrix in Table 42. The results show that 238 instances were correctly labeled as being hyperpartisan ("true"), but that 58 instances were incorrectly labeled as being hyperpartisan news. In addition, 256 documents were correctly labeled as non-hyperpartisan news. However, 76 documents were labeled as not being hyperpartisan, whereas they were hyperpartisan according to the gold standard. This suggests that the system is likely to incorrectly label a document as being mainstream/non-hyperpartisan news. This could be related to the substantially higher amount of mainstream news documents in the training set.

9.1.2 Important Features

We attempted to expose the most important features that our classifier selected by observing the weight coefficients that the SVM classifier assigned to the features. We were unable to do this for our best classifier on the by-article test set, because it was not possible to inspect the weights of the sentiment features. Therefore, we only show the most important features for a classifier that is almost the same as our classifier. The only difference is that this classifier did not use sentiment features. Thus, this classifier relied on clusters

	Precision	Recall	F1-score	Support
True	0.8041	0.7580	0.7803	314
False	0.7711	0.8153	0.7926	314

(a) precision, recall and f1-score.

	True	False
True	238	76
False	58	256

(b) matrix showing predictions (columns) and gold standard (rows).

Table 42: The performance of our best classifier on hyperpartisan news detection, evaluated on the by-article test set. The classifier scored an accuracy of 0.7866.

`hyp-s300-w5-700`, bag-of-prefixes (1-to-3/4) and POS uni-grams. We also removed the stop words of the documents, following the same procedure of the best classifier.

We show the Top 30 most important features for predicting hyperpartisan (the positive coordinates) and mainstream news (the negative coordinates) in Figure 7. The longer the bars, the higher the importance of the features. The results show that clusters are important features to predict hyperpartisan news. More specifically, 14 out of 30 of the most contributory features for hyperpartisan news detection are clusters, as well as the three most important features.

We show a proportion of the words of the clusters that are among the top three most important features for predicting hyperpartisan news detection in Table 43. It is worthwhile to note that the words in the clusters from Table 43 are related to topics and words that are associated with hyperpartisan news. Some examples of such words are *hate*, *anti-immigrant*, and *dogwhistle*. This observation reveals that the classifier correctly models the task of hyperpartisan news detection by making efficiently use of the occurrence of themes and words that are related to hyperpartisan news. It also explains why it is important to work with clusters that are related to the task.

In addition to clusters, prefixes are also important features for hyperpartisan news detection, according to Figure 7. Even though the prefixes were able to capture important sequences of words, such as *Don* and *Dona*, they mainly extracted features related to punctuation marks. Following this, it is possible that a more effective pre-processing to remove punctuation marks might have helped to improve performance.

Finally, Figure 7 also shows that POS uni-grams are helpful features for hyperpartisan news detection. The POS uni-grams seem to share characteristics with the prefix features. There are namely some words in the POS uni-grams which are also captured by the prefixes, such as the words *said* and *men*. However, the effectiveness of the POS uni-grams is also partly related to the same issue as we mentioned about the prefixes, as there appear several punctuation marks in the POS uni-grams as well.

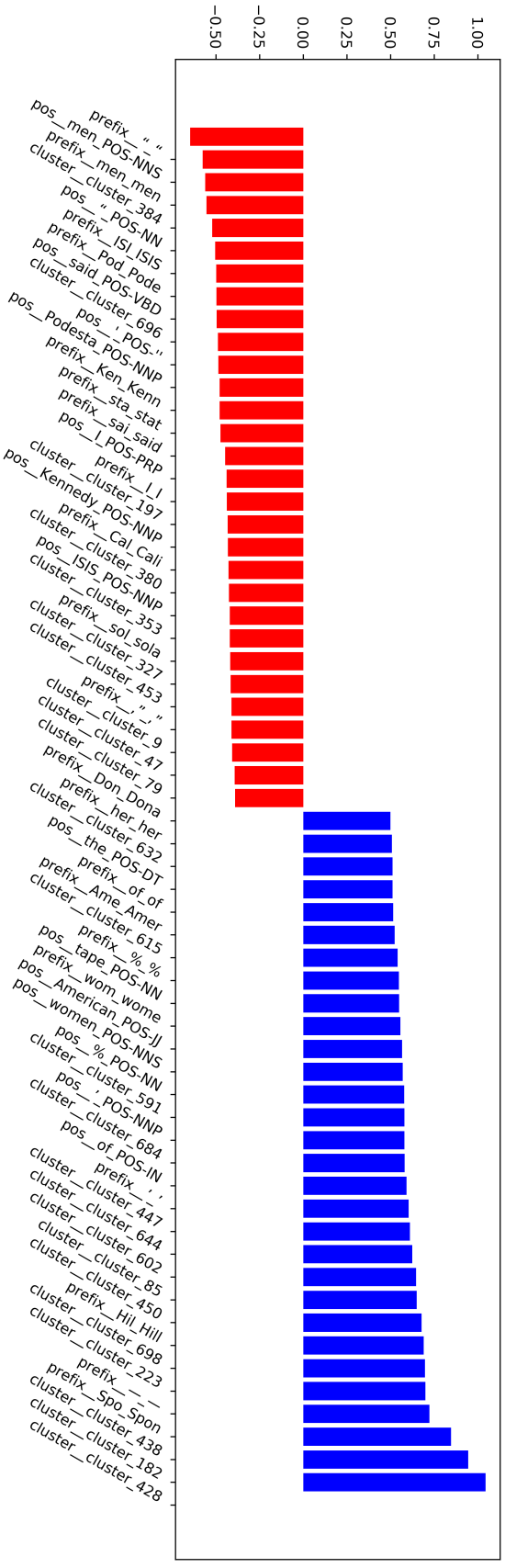


Figure 7: The top 30 most important features for predicting hyperpartisan (positive values) and mainstream news (negative values).

Cluster 428	**Cluster 182**	**Cluster 438**
threats	alwaysa	bi-partisans
herinsensitive	ddresspolicies	repressively
anti-indigenous	nicer	rebelling
dogwhistle	about.republicans	electoralism
anti-defamation	because	jacksonians
anti-hispanic	blink-182	neofascists
silencing	maybe	outmaneuvered
klanner	obamacare.they	revolutionaries
hateful	somewhere	subordinating
white-supremacist	crummy	non-ideological
slights	anything	anticolonial
k.k.k.	advertisementyes	potentates
decry	luckily	superstate
fascistic	it.donald	anti-revolutionary
anti-blackness	commiserate	overthrowing
hate	seeing	plutocratic
blasphemous	feels	counter-revolutionary
anti-immigrant	happier	theocracies
divisiveness	americaplan	repressiveness
racists	obama-	plebeians

Table 43: The three most important clusters for predicting hyperpartisan news (left: most important, right: less important). The three clusters were also the most important features to detect hyperpartisan news.

9.2 In-domain Results

We scored an accuracy of 0.7866 on the by-article test set, which would yield the 8th out of 42th position in the competition. Our best system on the by-publisher test set scored an accuracy of 0.6525, which would have resulted to the 5th out of 28th position in the ranking. The most probable reason why we did not manage to obtain a similar score on the by-publisher test set might be related to the lesser quality of the annotations in the by-publisher data in general. The documents from the by-publisher corpora were namely semi-automatically annotated, whereas the documents from the by-article data were manually annotated. Besides, we have seen that there were documents in the data which were not about political news, even though they were labeled as being hyperpartisan news (see Chapter 3).

Overall, we obtained good results with the SVM classifiers in the in-domain evaluation procedure. This supports the previous studies that we discussed, which proved that SVMs are effective for related tasks to hyperpartisan news detection, including fake news. Furthermore, the results revealed that, when tested on their own, sparse document representations yield better results than dense document representations. We explained that it could be the case that there were a low amount of out-of-vocabulary words in the test set. Besides, we have seen in Section 9.1.2 that prefixes, for instance, tended to extract patterns that are not directly associated with hyperpartisan news. Instead, they captured elements that occurred specifically in the corpus of consideration, such as punctuation sequences. It could be the case that these punctuation sequences occurred frequently in the test set as well.

In addition, the results of the SVMs showed that local features and/or combining several document representations increased the performance of classifiers that only relied on either one of the three document representation techniques that we used in SVM. Examples of local features were POS uni-grams, POS bi-grams and the sentiment of the text. We mentioned in Chapter 2 that these features proved to be effective in studies that focused on bias detection in Wikipedia, such as Recasens et al. (2013). Thus, our results confirm the previous findings on local features for related tasks. However, our results also showed that in addition to combining local features, combining document representations also helps to increase the results. This aspect was not discussed in the works of the theoretical framework that we outlined, since these studies focused on combining a large amount of local features, with minimal usage or experimentation on document representations.

Lastly, we were unable to obtain good results with Recurrent Neural Networks, despite that they were the most sophisticated method of document representation. One reason could be the size of the by-article training set, which might have been too small to train the neural networks. Another issue could be the length of the documents. Perhaps the documents were too long for the Recurrent Neural Networks, despite using the GRU variant. It could also be the case that the Flair Embeddings were not suitable for this task. Nonetheless, we believe that more in-depth work would be required to understand and improve the results for this task using Recurrent Neural Networks and Flair. A good point of departure would for instance be to study the obtained representations with Flair. It would

also be worthwhile to experiment in more detail with the parameters of the Recurrent Neural Networks, since we only used the default settings.

9.3 Results Across-datasets

We obtained remarkable results when we evaluated our SVM systems across-datasets, since we received the best results on both the by-article and by-publisher test set with the classifiers that only used word embeddings. More specifically, these systems used the word embeddings from FastText (trained on Common Crawl). The classifier using bag-of-clusters obtained better results than the prefixes on the by-publisher test set when we trained the classifiers on the by-article training set. However, the clusters scored lower than the bag-of-words on the by-article test set, when we trained our systems on the by-publisher subset. We suspect that this is related to the corpus of the clusters, since the clusters were trained on the training and validation set of the by-publisher corpus. Consequently, this could have limited the amount of out-of-vocabulary words (i.e., clusters) on the by-publisher test set.

The results also showed that, in contrary to the in-domain results, adding features generally decreases the performance. This was apparent in both the results of the by-publisher and by-article test set. One explanation could be that the local features, such as sentiment, tend to capture elements that are highly associated with characteristics from the corpus, rather than hyperpartisan news. Moreover, the reason that neither combining document representations did not increase the results could be that this resulted to over-fitting on the training data. It could also be the case that the document representations fail to effectively complement each other. It would therefore be interesting to perform a similar study of the most important features as we did in Section 9.1.2, specifically when evaluating a system across-datasets.

Moreover, we were unable to obtain better results on the by-article test set when we trained our classifiers on the by-publisher subset, despite the substantially larger size of the by-publisher subset. On one hand, this could be explained by the difference in the quality of the annotations of the by-publisher subset. As mentioned before, the quality of the annotations is lower than the quality of the annotations in the by-article training set. On the other hand, we did not manage to obtain good results on the by-publisher test set when we trained the classifiers on the by-article training set. The concern with this set-up could be the small size of the by-article training set. However, the low results on the by-publisher test set can also be related to the noise in the by-publisher corpus in general, especially since the competition results on the by-publisher test set were generally lower than on the by-article test set.

9.4 Out-of-domain Results

Even though our features were tuned for hyperpartisan news detection, our systems outperformed the classifiers from Pérez-Rosas et al. (2018) for fake news detection across-domains.

We achieved this with our our best bag-of-prefixes, bag-of-clusters (using hyperpartisan/-mainstream news clusters) and word embeddings classifiers. Since the systems of Pérez-Rosas et al. (2018) fully relied on local features, such as the readability of the texts, we concluded that document representation features work better across fake news domains. Thus, it seems that classifiers which are trained for a downstream-task on either bag-of-prefixes, bag-of-clusters and word embeddings are capable of working well across tasks and domains.

In addition, we scored the best results with the denser document representations. We obtained the best results on 5-fold cross-validation with word embeddings. This could be related to the same reasons as we mentioned about the good performance of word embeddings across-datasets. The word embeddings are trained on very large corpora (i.e., Common Crawl) and are therefore less sensitive to the negative effects of out-of-vocabulary words, which is more problematic for sparse representations such as prefixes. Something similar occurred with the clusters-based representations, which obtained the best results in this evaluation setting. Our findings are therefore particularly promising for experimental set-ups where it is difficult to train task and/or domain specific clusters due to scarcity or the unavailability of large corpora. A simple solution would then be to train clusters on a related task and to use these clusters for the current task.

10 Conclusion

In this final chapter, we answer the questions that we posed at the start of this thesis. We also discuss what we would have done if we would have had more time to continue working on this project.

10.1 Summary

This thesis described our work to develop competitive classification systems on the shared task on hyperpartisan news detection (SemEval-2019 Task-4). Our best system obtained an accuracy of 0.7866 on the official test set of the competition and used the following features: bag-of-clusters (using closely domain-related clusters), bag-of-prefixes, POS unigrams and the negative sentiment of the text. However, the models which used only dense word representations based on word clusters or word embeddings behaved more robustly when considering out-of-domain evaluation settings and evaluation settings across-datasets.

If we look back at our first research question, *What is the most effective method to vectorize documents for hyperpartisan news?*, the results indicated that clusters and word embeddings work best, although there were differences. For in-domain evaluations, combining them with local and sentiment features generally improved the results. However, as the findings of our experiments across-datasets (by-publisher and by-article) and across tasks (fake news) showed, the most robust models are those that use only clusters or word embeddings.

With respect to the second question, we were wondering whether we could improve the results of our classifiers by combining document representations and/or by adding local features. Our findings showed that this is indeed the case, but only in in-domain evaluation. Nonetheless, the improvements were only minor, and the extent to which the features effectively complemented each other remained relatively uncertain. In addition, when we evaluated our systems across-datasets, we observed that combining document representations and adding features were detrimental to the results. Instead, we received better results with the systems that only relied on either one of the three document representations that we used in the SVM classifiers: bag-of-words/characters, bag-of-clusters or word embeddings.

The third research question was: *Is it possible to build a classifier for hyperpartisan news detection that is robust enough to perform well across different datasets of hyperpartisan news?*. In our experiments, it was challenging to build such a system with the datasets that we used. On one hand, we worked with a large data set of hyperpartisan/mainstream news articles, but the quality of the annotations was relatively lower than the other data set. On the other hand, we also worked with a dataset with a good quality of annotations, but the dataset was relatively small. However, the results obtained when evaluating on the by-publisher test set showed that using clusters trained on the by-publisher training set helped to improve results. This means that using in-domain data to obtain dense word representations is the way to go in this setting.

In addition, our findings showed that it is possible to train classifiers for hyperpartisan

news detection and to use the most effective features of this experiment for fake news detection. Since we only conducted a few experiments, this finding can be seen as an interesting starting point for further research on transfer learning. Besides, there are currently more datasets available for fake news detection than for hyperpartisan news detection. These studies may go hand-in-hand with future research in journalism, to shed more light on the differences and similarities between fake news and hyperpartisan news.

However, we did not manage to obtain good results with the Recurrent Neural Networks and contextual-character based embeddings. If we would have had the opportunity to continue this work, we would have studied the obtained contextual character-based word embeddings from Flair. Therefore, further research will have to be carried out to shed more light on the performance of recurrent neural networks and contextual-character based embeddings for hyperpartisan news detection.

Summarizing, when the aim is to build a robust classifier that is able to perform well across-datasets on hyperpartisan news detection, we suggest to use dense document representations. However, to optimize the performance of a system on a specific data set, it is better to use a combination of dense word representations with task-specific features such as sentiment.

10.2 Closing Remarks

We presented a systematic comparative analysis to study the performance of different document representation techniques for hyperpartisan news detection. We obtained very competitive results for in-domain evaluation on a recent, official shared task on hyperpartisan news detection. We also described a robust set of features that can be used for hyperpartisan news detection that also work well on fake news detection, without using specific tuning for fake news.

Hyperpartisan news detection is an interesting problem that can be tackled with simple text classifiers. We hope that the body of research on hyperpartisan news detection will continue to grow in the future. In particular, we would like to see the task extended in two ways. First, by asking the systems to justify its decisions, namely, a type of "interpretable hyperpartisan news detection". Second, in terms of the granularity of analysis in order to be able to detect who said what, when and to whom. We believe that this would allow to better understand the reasons why systems (and humans) identify a given document as hyperpartisan.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research at the University of the Basque Country.

References

Rodrigo Agerri and German Rigau. Language independent sequence labelling for opinion target extraction. *Artificial Intelligence*, 268:85–95, 12 2018. doi: 10.1016/j.artint.2018.12.002.

Hadeer Ahmed, Issa Traore, and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. In *Proceedings of the International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pages 127–138, 10 2017. ISBN 978-3-319-69154-1. doi: 10.1007/978-3-319-69155-8_9.

Oluwaseun Ajao, Deepayan Bhowmik, and Shahrzad Zargari. Fake news identification on twitter with hybrid cnn and rnn models. In *Proceedings of the 9th International Conference on Social Media and Society*, SMSociety '18, pages 226–230, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6334-1. doi: 10.1145/3217804.3217917. URL http://doi.acm.org/10.1145/3217804.3217917.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

Samir Bajaj. " the pope has a new baby ! " fake news detection using deep learning, 2017. URL https://web.stanford.edu/class/cs224n/reports/2710385.pdf. CS 224N, Winter 2017.

Shweta Bhatt, Sagar Joglekar, Shehar Bano, and Nishanth Sastry. Illuminating an ecosystem of partisan websites. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 545–554, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5640-4. doi: 10.1145/3184558.3188725. URL https://doi.org/10.1145/3184558.3188725.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL http://arxiv.org/abs/1409.1259.

Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. Authorship attribution with support vector machines. *Applied Intelligence*, 19, 07 2003. doi: 10.1023/A:1023824908771.

Asif Ekbal and Sivaji Bandyopadhyay. Named entity recognition using support vector machine: A language independent approach. *International Journal of Electrical, Computer, and Systems Engineering*, 4:155–170, 2010.

S. Jerril Gilda. Evaluating machine learning algorithms for fake news detection. *2017 IEEE 15th Student Conference on Research and Development (SCOReD)*, pages 110–115, 2017.

Stephan Greene and Philip Resnik. More than words: Syntactic packaging and implicit sentiment. In *NAACL*, pages 503–511, 2009.

R. Gunning. *The technique of clear writing*. McGraw-Hill International Book Co, New York: NY, 1952.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Joan B. Hooper. On assertive predicates. In J. Kimball, editor, *Syntax and Semantics Volume 4*, volume 4, pages 91 – 124. Academic Press, New York, 1975.

Benjamin D. Horne and Sibel Adali. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *CoRR*, abs/1703.09398, 2017.

Christoph Hube and Besnik Fetahu. Detecting biased statements in wikipedia. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1779–1786, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5640-4. doi: 10.1145/3184558.3191640. URL `https://doi.org/10.1145/3184558.3191640`.

C.J. Hutto, Scott Appling, and Dennis Folds. Computationally detecting and quantifying the degree of bias in sentence-level text of news stories. *HUSO 2015: The first international conference on HUman and Social Analytics*, pages 30–34, 2015.

Clayton J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eytan Adar, Paul Resnick, Munmun De Choudhury, Bernie Hogan, and Alice H. Oh, editors, *ICWSM*. The AAAI Press, 2014. URL `http://dblp.uni-trier.de/db/conf/icwsm/icwsm2014.html#HuttoG14`.

Bhumika Jahdav and Vimalkumar Vaghela. Sentiment analysis using support vector machine based on feature selection and semantic analysis. *International Journal of Computer Applications*, 146:26–30, 07 2016. doi: 10.5120/ijca2016910921.

Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016. URL `http://arxiv.org/abs/1607.01759`.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minnesota, United States, June 2019. Association for Computational Linguistics.

Han Kyul Kim, Hyunjoong Kim, and Sungzoon Cho. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing*, 266:336 – 352, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.05.046. URL http://www.sciencedirect.com/science/article/pii/S0925231217308962.

J.P. Kincaid, R.P Fishburne, R.L. Rogers, and B.S. Chissom. Derivation of new readability formulas (automated readability index, fog count and flesh reading ease formula. *Research Brand Report*, pages 8–75, 1975. Chief of Naval Technical Training: Naval Air Station Memphis.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257.

Dilek Küçük and Fazli Can. Stance detection on tweets: An svm-based approach. *CoRR*, abs/1803.08910, 2018. URL http://arxiv.org/abs/1803.08910.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360, 2016. URL http://arxiv.org/abs/1603.01360.

T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25:259–284, 1998.

Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *In Proceedings of WWW 2005*, pages 342–351, 2005.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, CA, 1967. University of California Press.

Alice Marwick and Rebecca Lewis. *Media Manipulation and Disinformation Online*. Data and Society, 2017.

G. McLaughlin. Smog grading — a new readability formula. *Journal of Reading*, 12: 639–646, 1969.

Thomas Mikolov. Distributed representations of words and phrases and their compositionality. *Proc. of ICLR*, pages 1–9, 01 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL `http://arxiv.org/abs/1301.3781`.

Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2004.

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

James Pennebaker, Ryan Boyd, Kayla Jordan, and Kate Blackburn. The development and psychometric properties of liwc2015. Technical report, University of Texas at Austin, 09 2015.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/C18-1287`.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL `http://arxiv.org/abs/1802.05365`.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. *CoRR*, abs/1702.05638, 2017. URL `http://arxiv.org/abs/1702.05638`.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. TIRA Integrate Research Environment. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer, 2019.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. Linguistic models for analyzing and detecting biased language. In *ACL (1)*, pages 1650–1659. The Association for Computer Linguistics, 2013.

Filipe Ribeiro, Lucas Henrique, Fabrício Benevenuto, Abhijnan Chakraborty, Juhi Kulshrestha, Mahmoudreza Babaei, and Krishna P Gummadi. Media bias monitor: Quantifying biases of social media news outlets at large-scale. In *In Proc. AAAI Intl. Conference on Web and Social Media (ICWSM), Stanford, USA, June 2018*, 09 2018.

Mir Shahriar Sabuj, Zakia Afrin, and K. M. Hasan. Opinion mining using support vector machine with web based diverse data. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 673–678, 11 2017. ISBN 978-3-319-69899-1. doi: 10.1007/978-3-319-69900-4_85.

Sneha Singhania, Nigel Fernandez, and Shrisha Rao. 3han: A deep neural network for fake news detection. In *Proceedings of the International Conference on Neural Information Processing*, 11 2017. doi: 10.1007/978-3-319-70096-0_59.

E.A. Smith and R. Senter. The automated readability index. *Aerospace Medical Research Laboratories (6570th)*, 1, 1967.

Philip J. Stone, Robert F. Bales, J. Zvi Namenwirth, and Daniel M. Ogilvie. The general inquirer: A computer system for content analysis and retrieval based on the sentence as a unit of information. *Behavioral Science*, 7:484 – 498, 10 2007. doi: 10.1002/bs.3830070412.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. Sentiment strength detection for the social web. *J. Am. Soc. Inf. Sci. Technol.*, 63(1):163–173, January 2012. ISSN 1532-2882. doi: 10.1002/asi.21662. URL `http://dx.doi.org/10.1002/asi.21662`.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P10-1040`.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg, 1995. ISBN 0-387-94559-8.

Emmanuel Vincent and Maria Mestre. Crowdsourced measure of news articles bias: Assessing contributors' reliability. In *CEUR Workshop Proceedings*, volume 2276, 2019.

B. Wyse. Factive / non-factive predicate recognition within question generation systems. Technical Report 2009/09, Department of Computing Faculty of Mathematics, Computing and Technology The Open University, Walton Hall, Milton Keynes, MK7 6AA, September 2009. URL `http://computing-reports.open.ac.uk/2009/TR2009-09.pdf`.

Chao Xing, Dong Wang, Xuewei Zhang, and Chao Liu. Document classification with distributions of word vectors. *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–5, 2014.

Tae Yano, Philip Resnik, and Noah A. Smith. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 152–158, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1866696.1866719.

Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 895–906. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7368-on-the-dimensionality-of-word-embedding.pdf.

Nurulhuda Zainuddin and Ali Selamat. Sentiment analysis using support vector machine. In *I4CT 2014 - 1st International Conference on Computer, Communications, and Control Technology, Proceedings*, pages 333–337, 09 2014. doi: 10.1109/I4CT.2014.6914200.