Diploma Thesis

TECHNICAL UNIVERSITY OF CRETE

School of Electrical and Computer Engineering

Effective fake news detection using machine learning techniques

Author:

Michael Mersinias

Thesis Committee:

Associate Professor
Georgios Chalkiadakis
Technical University of Crete
Chania, Greece

Associate Professor
Stergos Afantenos
Universite Paul Sabatier
Toulouse, France

Associate Professor
Michail G. Lagoudakis
Technical University of Crete
Chania, Greece

A thesis submitted in partial fulfillment of the requirements for the degree of

Diploma in Electrical and Computer Engineering

September 17, 2019

Effective fake news detection using machine learning techniques

by Michael Mersinias

Supervisor: Associate Professor Georgios Chalkiadakis

Abstract

Fake news detection has in recent years emerged as an important new research area, necessitating the development of effective machine learning (ML) solutions in order to identify the authenticity of reported news, and classify them as fake or not.

Against this background, in this thesis we put forward a novel text vectorization approach, that generates feature vectors of numerical statistics to describe a document. Our so-called *class label frequency distance (CLFD)*, bears certain advantages when compared to ''classic" text vectorization methods. Moreover, we detail how to incorporate the approach within certain ML methods used for document classification. *CLFD* is shown experimentally to provide an effective way for boosting the performance of those ML methods.

Specifically, our experiments, carried out in the fake news detection domain, verify that *efficient* traditional ML methods that use our vectorization approach, consistently outperform deep learning methods that use word embeddings for small and medium sized datasets, while the results are comparable for large datasets. In addition, we demonstrate that a novel *hybrid* method that utilizes both a *CLFD*-boosted logistic regression classifier and a deep learning one, clearly outperforms deep learning methods even in large datasets.

Finally, for two datasets used in the literature, our *CLFD* vectorization approach allows both the *hybrid* method and certain traditional machine learning methods to provide significantly better results than those reported in recent published work within the fake news detection domain.

Αποτελεσματική ανίχνευση ''ψευδών ειδήσεων'' με χρήση τεχνικών μηχανικής μάθησης

του Μιχαήλ Μερσινιά

Επιβλέπων: Αναπληρωτής Καθηγητής Γεώργιος Χαλκιαδάκης

Περίληψη

Τα τελευταία χρόνια, η ανίχνευση ψευδών ειδήσεων είναι μια ανερχόμενη περιοχή έρευνας, που συνδέεται με την ανάπτυξη μεθόδων μηχανικής μάθησης (ΜΜ) για την ταυτοποίηση της αυθεντικότητας και την επιτυχή και αποδοτική ταξινόμηση κειμένων ειδήσεων ως ψευδών ή μη.

Στην παρούσα διπλωματική εργασία, προτείνουμε ένα νέο στατιστικό μοντέλο διανυσματοποίησης κειμένου, που αποσκοπεί στην δημιουργία διανυσμάτων χαρακτηριστικών για την αριθμητική αναπαράσταση ενός αρχείου κειμένου. Η μέθοδός μας, την οποία καλούμε *class label frequency distance (CLFD)*, έχει συγκεκριμένα προτερήματα σε σχέση με ''κλασσικές'' μεθόδους διανυσματικοποίσης κειμένου. Στην εργασία μας εξηγούμε το πώς μπορεί να ενσωματωθεί σε μεθόδους ΜΜ που χρησιμοποιούνται για ταξινόμηση κειμένων. Αποδεικνύουμε πειραματικά ότι η *CLFD* αποτελεί έναν αποτελεσματικό τρόπο για την βελτίωση της απόδοσης των μεθόδων αυτών.

Πιο συγκεκριμένα, κατά την πειραματική διαδικασία, η οποία διεξήχθη στο πεδίο της ανίχνευσης ψευδών ειδήσεων, επαληθεύεται ότι τα αποτελέσματα συγκεκριμένων παραδοσιακών μεθόδων ΜΜ με αποδοτική πολυπλοκότητα που χρησιμοποιούν την προτεινόμενη μέθοδο διανυσματοποίησης, είναι στατιστικά καλύτερα σε σχέση με αυτά των υψηλής πολυπλοκότητας μεθόδων βαθιάς μάθησης για μικρού και μεσαίου όγκου συλλογές αρχείων ειδήσεων - ενώ ταυτόχρονα, η απόδοση των μεθόδων είναι παρόμοια για μεγάλου όγκου συλλογές.

Ακολούθως, προτείνουμε μία νέα, *υβριδική* μέθοδο ΜΜ που συνδυάζει την παραδοσιακή και αποδοτική μέθοδο ''λογιστικής παλινδρόμησης'' (''logistic regression'') η οποία αξιοποιεί το στατιστικό μας μοντέλο και μία μη γραμμική μέθοδο βαθιάς μάθησης. Αποδεικνύουμε πειραματικά ότι η απόδοση της υβριδικής μεθόδου ξεπερνάει αυτήν των μη γραμμικών μεθόδων βαθιάς μάθησης ακόμα και για μεγάλου όγκου συλλογές αρχείων.

Τέλος, σε δύο συλλογές αρχείων ειδήσεων που χρησιμοποιούνται στη βιβλιογραφία, η εφαρμογή του στατιστικού μας μοντέλου είχε ως αποτέλεσμα η απόδοση τόσο της υβριδικής μεθόδου όσο και ορισμένων κλασικών μεθόδων μηχανικής μάθησης να είναι σημαντικά καλύτερη από αυτήν που αναφέρεται σε πρόσφατη σχετική ερευνητική δημοσίευση.

Acknowledgements

First and foremost, I would like to express my deep appreciation towards my supervisor, Associate Professor Georgios Chalkiadakis, for his support and trust in me as he gave me the opportunity to study and conduct research on a field which I had great interest but no prior exposure. I would also like to thank the other two members of the committee and especially Associate Professor Stergos Afantenos for his support and motivation regarding the field of natural language processing.

I cannot neglect to thank my family as it is them who have instilled in me the morals and the values that I always honor and uphold in life. Firstly, my mother, Maria, who, despite being deaf, has always been there to truly hear me and provide immense support. Secondly, my father, Dimitris, who has intrigued me to follow his steps into engineering and has helped me develop critical thinking and problem solving skills. Thirdly, my uncle, Michalis, who has helped me develop a drive and an ambition which in turn became a constant thirst for self-improvement.

Finally, I would like to thank my close friends Kostas, Dimitris and Michalis who have been my brothers-in-arms in both the academic and the social battlefields of life for 6 years. I would also like to thank Lexie for being by my side for 2 years. Despite being so far away, 10 time zones afar in the West Coast of the United States, she has been the closest one to me and has motivated me and supported me through both good and bad times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> ‘’The information of the people at large can alone make them the safe, as they are the sole, depository of our political and religious freedom.”
>
> Thomas Jefferson

Fake news is a term which describes news articles that are "intentionally and verifiably false and can mislead readers" [1]. This phenomenon has existed throughout history, from the turbulent times of the late Roman Republic when after the assassination of Julius Caesar, a disinformation war occurred between Octavian and Mark Anthony which included heavy use of propaganda in form of vitriolic speeches and defamatory slogans written upon coins in the style of archaic tweets. Later on, in the early 19th century, the New York newspaper *The Sun* published a series of articles about the supposed discovery of life and civilization on the Moon and described its inhabitants to consist of unicorns, bat-like winged humanoids and other fantastic animals. The *Great Moon Hoax*, as it was later named, managed to spread quickly and established *The Sun* as a successful newspaper.

However, the frequency and impact of fake news has substantially increased in the 21st century, due to the easy access to unfiltered information that is available to the public, mainly anonymously across the Web. In fact, the issue is so prevalent that Fake News has been named as the 2017 word of the year by the Collins dictionary [2], It has been shown that fake news spreads faster than credible news [3] while at the same time, fake news generation is greater than news fact checking [4]. According to a March 2018 Eurobarometer survey [5], fake news constitutes a threat to democracy for 83% of the EU population. Many governments have signed fake news laws which make it a crime to spread fake news online [6, 7]. This means that there is a growing demand for automatically detecting and stopping the spread of fake news, as its quick and effective detection can have a positive social impact.

In the following sections, we provide a clear definition about fake news as a term, we discuss the challenges and the objectives of fake news detection and finally we outline the contributions and the organization of this thesis.

1.1 Defining Fake News

Fake news is an umbrella term which encompasses various forms of unreliable information, although as a term, it may often have different interpretations. As mentioned above, at its core, it refers to news articles which are intentionally and verifiably false and can mislead readers. These news articles usually contain fabricated stories with no verifiable facts, sources or quotes. These stories may be propaganda which is intentionally designed to push a specific agenda by misleading the readers, or they may be clickbait written for economic incentives as the writer profits on the number of people who click on the story. However, it is not only false stories that may be defined as fake news. Certain news articles may contain the truth but may be written in language which is intentionally inflammatory, leaves out contextual details, or only presents one viewpoint. All of the above [8] can be summarized into two main categories of fake news:

- **Misinformation**: Misleading or inaccurate information that is mistakenly or inadvertently created and spread, but the intent is not to deceive.
- **Disinformation**: False information that is deliberately created and spread in order to influence public opinion or obscure the truth [9].

In Figure 1.1, a chart [10] is presented with seven types of misinformation and disinformation. They represent the basic forms of fake news as defined above.

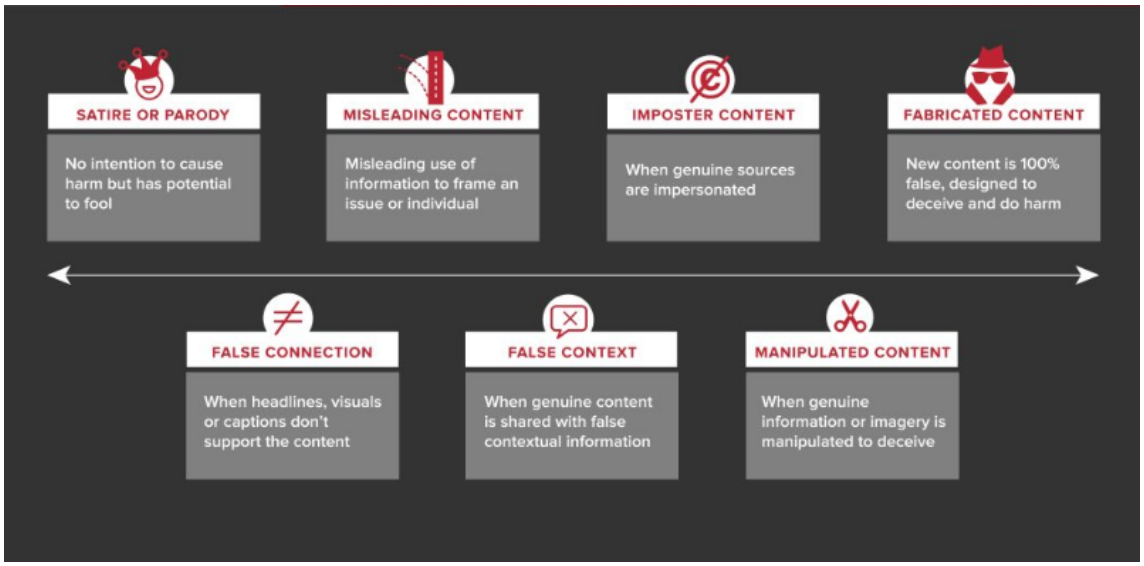


Figure 1.1: Seven types of mis/dis-information

1.2 Challenges & Objectives

In our work, we follow a natural language processing and machine learning approach and treat fake news detection as a binary text classification task which aims to classify news articles into credible or fake news. However, the complex nature of fake news, as described in Section 1.1, makes their detection a challenging task.

An important issue is that there is a lack of quantity and quality regarding training data. The lack of large quantities of training data is even more prevalent in the real world due to the fact that large datasets are difficult to build and annotate or they are proprietary. Furthermore, due to the need of early detection, classification time efficiency is another important factor. Therefore, our fake news detection system need to be time efficient and achieve a high performance regardless of dataset size.

Another issue is that since fake news are often created anonymously and spread by other sources, there is little to no information about them regarding non-textual metadata such as the source, the original author or the country of origin. Furthermore, information about the people who have read, liked or shared a news article might be either unavailable or raise concerns regarding their privacy. Therefore, the only consistently available information is the content of the news article.

After taking all of the above into account, our approach is to rely solely on the content, the title and the body text, of the news article and treat this information as the only available input data. This content-based fake news detection approach allows the classification to be done even on anonymous posts and the lack of non-textual features is not an obstacle, as the classification is performed on textual features that are always available. Finally, the fake news detection system needs to be time efficient and consistent in achieving high performance regardless of dataset size. Thus, the objective of this thesis is the creation of a robust, high performing and time efficient content-based fake news detection system.

1.3 Contributions

Our contributions can be summarized as follows:

- A definition of fake news and a description of different fake news detection system types.
- A novel numerical statistical approach for the construction of document feature vectors which boost the performance of certain machine learning methods. We also create and evaluate variations of this approach. Our vectorization approach leads to experimental results that show a high ranking among certain methods that use it. We demonstrate on particular datasets that our approach leads to performance that is superior to that of deep neural networks in small and medium sized datasets, while it is comparable in larger ones. Finally, in two datasets, our algorithms produce results that are significantly better than those reported in recent published work concerning those exact datasets.
- A novel hybrid method which combines the best performing machine learning methods through an average probability system for more effective classification. Furthermore, the hybrid method makes use of information retrieval techniques to search credible online news corpora in order to add a positive bias towards credible news during classification if there is a high similarity between the titles of the news articles.
- A systematic comparison of five vectorization techniques as well as five traditional machine learning and four deep learning methods for the binary text classification task of creating a fake news detection system that relies solely on the content of the news articles.

1.4 Thesis Organization

In this section we outline the organization of this thesis and present the content of each chapter.

In **Chapter 2**, we present related work regarding fake news detection and emphasize certain aspects of literature which have been taken into consideration for this thesis.

In **Chapter 3**, we define common natural language techniques for textual data preprocessing such as tokenization, stopword removal, stemming, lemmatizing and more. Furthermore, we define text vectorization and describe the most common text vectorization techniques. Finally, we present the chain of processes (pipeline) which outlines the functionality of our fake news detection system.

In **Chapter 4**, we develop a novel statistical algorithm for effective text vectorization. We provide the mathematical background and discuss the advantages it may provide compared to traditional text vectorization techniques. Furthermore, we develop and present variations of our algorithm. Finally, we describe its application in fake news detection by providing a detailed example.

In **Chapter 5**, we present and describe several machine learning methods which will be compared for the task of fake news detection. We describe their functionality, as well as their advantages and disadvantages, in detail. Furthermore, we outline the reasoning behind hyper-parameter selection for each of them. Finally, we propose a hybrid method which combines the prediction results of some of these methods, as well as information retrieval techniques, into an effective classifier and we discuss the advantages it may provide.

In **Chapter 6**, we present our experimental results. The evaluation is carried out on three datasets which differ in size, class balance and topic homogeneity. We systematically compare five vectorization techniques as well as five traditional machine learning and four deep learning methods. For the best performing methods, we further analyze the results along with error bars which correspond to 95% confidence intervals. Finally, we discuss the results in detail and point out the effectiveness of our vectorization approach as well as that of the hybrid method in terms of classification performance and time efficiency.

In **Chapter 7**, we conclude this thesis, and we provide directions for future work and possible extensions to our solutions.

Chapter 2

Related Work

There has been a considerable amount of work on fake news detection in the past few years. Existing research falls broadly within the realm of *linguistic analysis*, which uses natural language processing and machine learning methods; or that of *network analysis*, which utilizes knowledge network graphs [11]. We follow the first approach and treat fake news detection as a binary text classification task.

Considering a fake news detection system, a constant flow of news articles has to be classified as either credible or fake. More generally, a dataset is composed of a number of news articles, or documents, each of which contains a certain number of features. The features always included are the title and the body text of the news article. Other features such as the author may sometimes be included. Therefore, two types of fake news detection systems may be created depending on the assumption we make about the features included in the classification. We call the first type *content-based* and the second type *multi-criteria*. There is a need to define these two types and distinguish between them.

- In **content-based** fake news detection systems, the data provided only contains the title and the body text of each news article.

- In **multi-criteria** fake news detection systems, the data provided contains the title and the body text of each news article as well as other characteristics such as the author, the country of origin, the profiles of the readers who might have liked or shared it, the date, and more.

Most of the related work focuses on multi-criteria fake news detection systems. However, in practice, fake news are often created anonymously and spread by other sources, therefore there is little to no information about the source, the original author or the country of origin. As mentioned in Section 1.1, fake news may be in the form of information that is mistakenly or inadvertently created and spread and the intent might not be to deceive (misinformation). In this case, the profiles of the people who have liked or shared the news article might give little to no insight to the fake news detection system. Furthermore, making use of profile information contained in a social media platform might raise concerns over the privacy of its users or might be unavailable in the first place.

Therefore, while multi-criteria systems contain more information and may lead to higher performance, content-based systems utilize information that is consistently available. In the following subsections, we present certain approaches used in related work regarding fake news detection as well as their respective results.

2.1 Multi-criteria fake news detection systems

A high performing fake news detection system, CSI [12], is a multi-criteria hybrid system which combines deep learning methods for the text, the responses and the profile of the users who have promoted the news articles and achieves the best performance with an F1-score of 89% and 95% in two different datasets. Other similar systems have produced high results as well [13] [14]. However, these are multi-criteria fake news detection systems, as described above, and do not follow our content-based approach.

2.2 Content-based fake news detection systems

Related work regarding content-based fake news detection systems has been to date relatively limited. A comparison of several deep learning methods [15] has shown that recurrent neural networks and specifically LSTMs, Bidirectional LSTMs (Bi-lstm), and GRUs, which use pre-trained GloVE word embeddings [16] as input, are suitable for identifying fake news as they achieved the highest F1-score (81%). Another content-based fake news detection system found in the literature [17], combines the representation results from different methods into a single classification method. More specifically, a multilayer perceptron is used so the representation results of a convolutional (CNN) and a Bi-lstm neural network are combined into an effective classifier for fake news detection.

A recent content-based fake news detection system [18] utilizes traditional machine learning methods, such as *Gradient Boosting* [19], in order to achieve a compromise between classification time and high performance. In our fake news detection system, we make use of the two datasets used in [18], and demonstrate that we increase the performance of traditional machine learning methods through our novel vectorization approach. In this way, we retain the classification time advantage of traditional machine learning methods, while achieving very high performance: one that is on par or higher than that of deep learning methods, and significantly higher than the aforementioned state-of-the-art for those datasets.

Furthermore, it is worth mentioning the stance detection approach, as a subcategory of content-based fake news detection systems, which was inspired by the FNC-1 part of the "fake news challenge" [20]. It assumes that a great dissimilarity between the title and the body text of a news article is a solid sign of fake news. In literature, one of the best results of this approach, with an FNC-1 score of 81.72%, is achieved by a method which makes use of a shallow neural network and its input consists of the *tf-idf* vector of both the title and the body text of the news article as well as their cosine similarity [21]. Finally, a different subcategory of content-based fake news detection systems is the use of tensor embeddings which, in a semi-supervised approach, uses a fraction of the labelled data (2%) and achieves high performance with an accuracy score of 70.92% [22], as well as other content-based systems which are based on syntactic and semantic analysis [23, 24, 25] or hierarchical discourse-level structure [26] and show interesting results as well.

We conclude that the good results achieved in the scarce but important research done on content-based systems make this approach promising, as they show that relying solely on the content for classification does not significantly decrease the performance of a fake news detection system.

Chapter 3

Background

In this chapter, we provide some basic definitions related to natural language processing and text classification. We first present certain data preprocessing techniques which are commonly used as a precedent step to text classification. We then define the process of vectorization and describe the most common methods in detail. Finally, we outline the chain of processes (pipeline) and the functionality of a content-based fake news detection system.

3.1 Data Preprocessing

In this section, we define certain text preprocessing techniques related to natural language processing and describe their functionality and usage in detail. These necessary steps of text preprocessing provide dimensionality reduction by reducing the vocabulary size the classification methods are exposed to, while at the same time, there is an increase in performance due to the explicit correlation of words with similar meanings and the removal of other words with little to no classification value.

3.1.1 Corpus, Tokenization & Vocabulary

We first provide some basic natural language processing definitions. The corpus is defined as the collection of documents, news articles in our case, which constitute our dataset. Furthermore, we define the vocabulary as the set of words which occur in the corpus. For example, consider a corpus which consists of the following two documents: "I love going out", "I love a nice ice cream". The process of tokenization generates a list of tokens which constitute the vocabulary. Therefore, after performing tokenization based on whitespace separation in this example, the resulting vocabulary is as follows: ["I", "love", "going", "out", "a", "nice", "ice", "cream"]. Each element of the vocabulary corresponds to a unique word contained in the corpus regardless of its number of occurrences and it may be notated as term, word or token.

3.1.2 Stopword Removal

It is obvious, even in the simple example presented above, that certain terms such as the term "a" have no value in classification. In fact, this is the case with very common terms which have a very high frequency of occurrence in the English language and may as a result be overvalued. This phenomenon creates noise in the input of the machine learning algorithms. In order to counter this, a list of stopwords is used in order to achieve the selective exclusion of certain terms from the vocabulary. An example of such a stopword list is presented below:

Stopword list: ["to", "a", ..., "the"]

However, not all such terms should be included in a stopword list as certain high frequency terms may be useful features in specific domains and tasks. For example, the term "not" has a high frequency in the English language but it is useful for identifying negation and may provide insight in certain cases.

3.1.3 Stemming & Lemmatizing

The goal of both processes [27] is to condense derived words into their basic forms.

- **Lemmatizing** is the process of grouping together the inflected forms of a term so they can be analysed as a single term, identified by the lemma of that term. It is often more accurate than stemming as it uses more informed analysis to create groups of terms with similar meaning based on the context around each term.

- **Stemming** is a similar process which is typically faster but less accurate as it simply chops off the end of a term using heuristics, without any understanding of the context in which a term is used.

In order to showcase the difference between lemmatizing and stemming, we present two examples. For the first example, consider the terms "meanness" and "meaning". Stemming transforms both of them to "mean", while lemmatizing does not make any changes. For the second example, consider the terms "goose" and "geese". Lemmatizing transforms both of them to "goose" while stemming transforms them into "goos" and "gees" respectively. Therefore, lemmatizing was the choice for our fake news detection system as it is preferable to stemming due to its increased accuracy.

3.2 Vectorization

As mentioned in Chapter 1, the goal of our fake news detection system is to classify news articles into credible or fake news by making use of machine learning classification methods. We can use a linear model, $y = w \cdot x$, where w is a vector of weights estimated by the machine learning method and x is a feature vector. Alternatively, we can use a non-linear model with a deep learning architecture. In our work, we experiment with both linear and non-linear methods.

However, most of these methods have been designed to learn from numerical data. In the case of news articles, the input is textual data which comes in the form of a sequence of terms, punctuation and whitespaces. All of the above are based on string data and thus, as a precedent step to classification, it is necessary to have a numerical representation which is equivalent to the string representation of every document. This process is called *text vectorization* [28]. Each term is regarded as a feature and it is assigned a numerical representation based on certain attributes such as its frequency or the context around it. The result is a feature vector for each document which corresponds to the set of features which will be used by the classification methods during the learning process. In this section, we present the most common vectorization techniques. We describe their functionality in detail, provide examples and discuss their respective advantages and disadvantages.

3.2.1 Term Frequency

Let us consider the following two documents:

"Mike likes to play football. Alex likes to play football too."
"Mike also likes to play basketball."

Performing tokenization based on whitespace separation and removing punctuation, leads to the generation of the following vocabulary:

["mike", "likes", "to", "play", "football", "alex", "too", "also", "basketball"]

Term frequency (tf) is a simple and useful numerical statistic used for term representation. It calculates the number of times each term of the vocabulary occurs in each document. This calculation results in a feature vector of length v for each document, where v is the length of the vocabulary. Each element represents the number of times the term in position i of the vocabulary appeared in the respective document. A variation of this method is *binary term frequency* ($b - tf$) vectorization where each term takes a value of 1 in case of occurrence in the document or a value of 0 otherwise. In our example, as we have two documents, the result would be the following two vectors:

	mike	likes	to	play	football	alex	too	also	basketball
Document 1	1	2	2	2	2	1	1	0	0
Document 2	1	1	1	1	0	0	0	1	1

Table 3.1: Term Frequency (TF) Vectors

	mike	likes	to	play	football	alex	too	also	basketball
Document 1	1	1	1	1	1	1	1	0	0
Document 2	1	1	1	1	0	0	0	1	1

Table 3.2: Binary Term Frequency (B-TF) Vectors

The advantages of *term frequency* (tf) and *binary term frequency* (b-tf) vectorization techniques are the following:

- Simple and fast vectorization techniques which provide a simple but effective term representation and often result in high accuracy.

However, the disadvantages of *term frequency* (tf) vectorization are the following:

- High sensitivity to terms such as "a", "the" and "is" which have a very high frequency of occurrence in the English language but do not provide any useful information. Therefore, in order not to overvalue these terms and allow noise in the input of the machine learning algorithms, a carefully created stopword list has to be generated for the selective exclusion of such terms.

3.2.2 Term Frequency - Inverse Document Frequency

An alternative approach is to emphasize terms which may be rare and interesting in a document but are overshadowed by other, more frequent terms whose information is not useful. This may happen either because the aforementioned terms were not identified as stopwords or because some domain-specific terms may be frequent in certain corpora but provide no classification value, e.g. the term "movie" in the imdb movie reviews dataset. In order to counter this issue, *inverse document frequency* is introduced. The *tf-idf* term weighting re-weights the *term frequency* (tf) features into floating point values that represent how rare, or important, each term is in the corpus. More specifically, *term frequency* (tf) is defined as the number of occurrences of a term in a document, divided by the total number of terms the document contains. The *inverse document frequency* (idf) depends on the whole document corpus and represents the total number of documents, divided by the number of documents which contain the term. Therefore, in order to calculate the *inverse document frequency*, we need to know the total number of documents (n_d) and the number of documents that contain the term, which is denoted as $df(d,t)$. The *tf-idf* values are calculated as follows:

$$idf(t) = \log \frac{1 + n_d}{1 + df(d,t)} + 1 \tag{3.1}$$

$$tfidf(t,d) = tf(t,d) \cdot idf(t) \tag{3.2}$$

In the previous example, the *tf-idf* document vectors would be the following:

	mike	likes	to	play	football	alex	too	also	basketball
Document 1	1	1	1	1	1.32	1.32	1.32	0	0
Document 2	1	1	1	1	0	0	0	1.32	1.32

Table 3.3: Term Frequency Inverse Document Frequency (TF-IDF) Vectors

The advantages of *tf-idf* vectorization are the following:

- Simple and fast technique which provides a more sophisticated term representation and often results in high accuracy.
- Low sensitivity to stopword selection due to the *idf* component. It significantly lowers the weight of terms that have a high frequency of occurrence in the English language as their occurrence in many different documents significantly decreases their *idf* value and as a result, their *tf-idf* weight.

However, the disadvantages of *tf-idf* vectorization are the following:

- A large corpus is required for an accurate impact of the *idf* component.
- The *tf-idf* term re-weighting isn't always beneficial as in certain cases it might reduce performance. For example, in a dataset such as imdb movie reviews, a common but important term such as the term "like" would have a much lower *tf-idf* weight compared to its respective term frequency weight. This could be a disadvantage in a natural language processing task, such as sentiment analysis, where this term should be considered important for classification.

3.2.3 Word embeddings

Word embeddings are used to improve the ability of classification methods, specifically deep learning methods, to learn from text data by representing terms as N-dimensional vectors. The vocabulary is represented in a vector space of several hundred dimensions, with each unique term in the corpus being assigned a corresponding vector in this space. They are based on the distributional hypothesis [29] stating that the meaning of a term can be understood from the contexts it appears in. Following this hypothesis, terms are represented by means of their neighbours and each term is associated with a vector that encodes information about its co-occurrence with other terms in the vocabulary. These vectors are called embeddings and provide a numerical representation, as described above, for every term.

One-hot encoding

The most common word embedding approach is to one-hot encode the textual data. In this case, we represent each word by an one-hot vector [0, ..., 0, 1, 0, ..., 0] of length v where v is the size of the vocabulary. The elements of this vector take a value of 1 at the index corresponding to the appropriate vocabulary term, and a value of 0 everywhere else. This approach requires substantial computational resources as computations with such one-hot encoded vectors are considered inefficient because all but one elements in each one-hot vector have a value of 0. Furthermore, the dimensionality of the vector space is very high and specifically at least $v \cdot h$ where v is the size of the vocabulary and h is the size of the neural network's hidden layer [30].

Pre-trained vectors

More advanced word embedding methods are pre-trained vectors which started as one-hot representations and dimensionality reduction methods were applied to them in the context of language models. They bring outside information and significantly reduce the dimensionality of the vector space which is R^d in this case, where d is the word embeddings size which is often assigned a value between 100 and 1000 [30]. Furthermore, the number of parameters that a method such as a neural network needs in order to learn from scratch is significantly reduced. *Word2vec* [31] and *GloVE* [16] word embeddings, developed by Google and Stanford University respectively, are the main pre-trained word embedding techniques used in natural language processing tasks. Their resulting representations showcase interesting linear substructures of the vector space as the vectors are positioned in the vector space in a way that allows terms that share common contexts in the corpus to be located in close proximity to one another. *Fasttext* [32], developed by Facebook, follows another approach and learns from subword information such as characters or character level n-grams.

3.3 Pipeline

As a problem approached by a natural language processing and supervised classification perspective, our content-based fake news detection system follows a specific chain of processes (pipeline). The pipeline includes steps such as data fetching and cleansing, data preprocessing, vectorization, method optimization and classification. More specifically:

- Data is being fetched and cleansed. All unwanted elements such as unrecognized characters or NaN elements are removed.
- The resulting raw text is tokenized into terms based on whitespace separation of text elements. Each token of text is preprocessed by performing techniques such as lemmatizating and stopword removal.
- Vectorization is performed and transforms each token of clean text to an equivalent numerical representation. This document feature vector generation step is being performed by making use of algorithms which are powerful term-weighting schemes that show the importance each term has for the classification process.
- Classification is performed through machine learning algorithms. Several combinations of vectorization techniques and machine learning methods are chosen. After fitting the training data, hyperparameters are tuned and evaluation follows.
- By comparing the results of the previous step, the best performing method, along with its respective feature vectors and hyperparameters, is chosen as the main method of the fake news detection system.

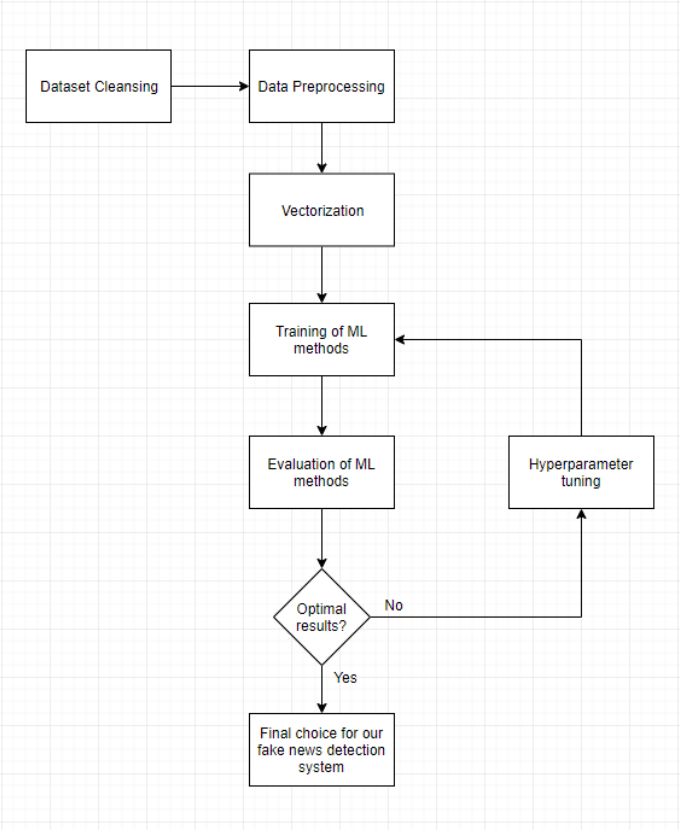


Figure 3.1: Fake news detection system pipeline

Chapter 4

Our approach

In this chapter, we propose a novel numerical statistical approach for vectorization which produces feature vectors that can be used for effective term numerical representation. As mentioned in Section 3.2, the process of transforming each textual element, or term, of a document into an equivalent numerical representation is called vectorization. While the traditional vectorization methods, such as *tf* or *tf-idf*, treat each term as part of a document, this vectorization technique assigns weights to each term based on the frequency it has within different class labels. We call this approach *class label frequency distance* or *clfd*. In what follows, we introduce our novel *clfd* technique and discuss its benefits; present its specification to fake news detection; and present machine learning algorithms that employ it or are used as baselines.

4.1 Mathematical Framework

We first provide a brief explanation of *clfd* before providing the details. Consider a corpus of documents D. Our approach works by determining the relative frequency of a term in a specific set of documents $D_i \in D$ which belong to a class label i within a set C of possible class labels, compared to the frequency of that term in the set of documents $\{D - D_i\}$ which belong to all other class labels. Intuitively, this calculation determines how relevant a given term is to a particular class label, relatively to its relevance to the rest of the class labels. We call this *class label frequency ratio* (*clfr*), and it is the main component of *clfd*. After calculating the *clfr* weight of a term t for each class label i, we calculate the maximum distance between those *clfr* weights by subtracting the smallest *clfr* value from the largest one. The result is the *clfd* weight for t, which signifies how likely it is that t belongs to a specific class label.

The formal procedure for determining *clfd* is the following. Given a corpus of documents D, a term t, and a set of documents $D_i \in D$ for each class label i, we first calculate three *class label term frequency* (*cltf*) vectors. The algorithm for their calculation is presented in the following page.

Algorithm 1: CLTF vectors calculation

```
Data: corpus, class_labels
Result: Generation of terms, occ, and total vectors
for document d in corpus do
    for class_label i in class_labels do
        if d belongs to i then
            for term t in d do
                terms(t,i) = t;
                occ(t,i) += tf(t,d);
                total(i) += tf(t,d);
            end
        end
    end
end
```

In the above algorithm, $tf(t, d)$ represents the number of occurrences of a term t in a document d. For each class label i, the vector *terms* contains the vocabulary and the vector *occ* the respective number of occurrences of each term t contained in *terms*. Finally, for each class label i, *total(i)* represents the total number of occurrences of all terms contained in the corresponding set of documents D_i.

Now that we have the three *cltf* vectors for each set of documents D_i that belong to class label i, we can calculate a *clfr* weight vector for each class label i, as follows:

$$clfr^i(t) = \log_e \frac{(1 + occ(t, i)) \cdot total(\hat{i})}{(1 + occ(t, \hat{i})) \cdot total(i)} + 1, \forall t \tag{4.1}$$

In Equation 4.1, *occ(t,i)* represents the number of occurrences of term t in D_i, while *occ(t,$\hat{i}$)* represents the number of occurrences of term t in $\{D - D_i\}$. Furthermore, *total(i)* contains the total number of occurrences of all terms which appear in D_i, while *total($\hat{i}$)* contains the total number of occurrences of all terms which appear in $\{D - D_i\}$. The $clfr^i$ vectors are used to calculate the *clfd* weights as follows:

$$clfd(t) = max_{i \in C}(clfr^i(t)) - min_{i \in C}(clfr^i(t)), \forall t \tag{4.2}$$

In Equation 4.2, the *clfd* weight for a term t is the maximum difference among its $clfr$ values. The algorithm for the generation of the (corpus-wide) *clfd* weight vector is presented in the following page.

Algorithm 2: CLFD calculation

Data: corpus, vocabulary, class_labels
Result: Generation of *clfd* vector

```
terms, occ, total = cltf(corpus, class_labels);
for term t in vocabulary do
    for class_label i in class_labels do
        clfr^i(t) computed as in Eq.4.1;
    end
    clfd(t) computed as in Eq.4.2;
end
```

In the above algorithm, we first calculate the *cltf* vectors *occ* and *total*, as in Algorithm 1, which represent, for each class label i, the number of occurrences of a term t in D_i and the total number of occurrences of all terms which appear in D_i respectively. The $clfr^i$ vectors contain the *clfr* weights of each term t for every class label, calculated according to Equation 4.1. Furthermore, *clfd* represents the maximum distance between those *clfr* weights, and is calculated according to Equation 4.2.

Then, the final *clfd* vector for a document d is the result of taking the Hadamard product between the computed *clfd* vector and one generated by a term frequency-based vectorizer such as *b-tf*, *tf* or *tf-idf*:

$$clfd^d(t) \leftarrow clfd(t) \circ \textit{tf-based\_vectorizer}(t, d), \forall t \tag{4.3}$$

This step is required in order to remove from a document's *clfd* vector the values that correspond to terms not appearing in that specific document, as a tf-based vectorizer calculates document-related numerical statistics, and would have assigned 0 values to such terms. All tf-based vectorizers create vectors of size equal to vocabulary size v, and Alg. 2 computes a *clfd* vector of size v, thus a simple element-wise multiplication is required in order to generate the final feature vectors for each document.

4.2 Discussion

The *clfd* technique provides a sophisticated weighting scheme with certain advantages compared to existing vectorization methods. This is because treating a term t as part of a set of documents D_i which belong to a class label i provides better insights for classification than simply treating t as part of a single document or the entire corpus.

The *clfd* weights represent the importance or relevance of every term for classification. If the *clfd* value of a term t is high, then that term is very likely to be related to documents associated with a specific class label. If the *clfd* value of a term t is low, then that term is not likely to occur in any documents associated with a specific class label. In case the *clfd* value is zero, then the term t is equally likely to occur in documents associated with any class label.

Compared to *term frequency* (*tf*) vectorization, *clfd* contains a natural filter to stopword noise due to the *class label frequency ratio* (*clfr*) component. Therefore, the performance of *clfd* is practically not affected by the quality or even the absence of data preprocessing: a term which has an equal number of occurrences between all class labels will have a *clfd* weight value of zero and it is practically removed. If a term has a significantly higher number of occurrences in documents which belong to a specific class label, and thus a high *clfd* weight, then it should not necessarily be considered a stopword for this domain in the first place, regardless of its frequency of occurrence in the English language.

Consider now *tf-idf* vectorization: in *tf-idf*, terms that are very common in certain domains, such as the term "like" in a sentiment analysis task, will have a low weight value due to the *idf* component. However, in the *clfd* approach, the term "like" will have a very high *clfr* value for the positive class label and a very low *clfr* value for the negative class label, thereby making the final *clfd* weight and relevance of the term "like", high.

Therefore, we can conclude that our vectorization approach provides important advantages over traditional vectorization methods. As such, it is conceivable that it can be used to boost the performance of machine learning algorithms, a fact verified experimentally in the fake news detection domain, as we detail later in this thesis.

4.3 Clfd variants

The vectorizer required for Equation 4.3 can be chosen from a list which includes a *b-tf* vectorizer (*b-clfd*), a *tf* vectorizer (*tf-clfd*) and a *tf-idf* vectorizer (*tfidf-clfd*). Consider an example of two documents: "Congress approved the controversial bill", "The congress is filled with lies, lies and deceit". The *clfd* variants are calculated as described in the following subsections.

Binary class label frequency distance

The first variant is to use the *clfd* vector as a multiplier to a *binary term frequency* (*b-tf*) vectorizer. We can multiply the final *clfd* weights with the respective *b-tf* weights for each term of a document.

In our example, the vocabulary is represented by this vector: [congress, approved, controversial, bill, full, lies, deceit]. After generating the *binary term frequency* (*b-tf*) feature vectors, as described in Section 3.2.1, we have [1, 1, 1, 1, 0, 0, 0] as the *b-tf* feature vector for document 1 and [1, 0, 0, 0, 1, 1, 1] as the *b-tf* feature vector for document 2. In the end, the resulting *b-clfd* feature vectors are as follows:

Vocabulary	clfd weights	b-tf vector (doc 1)	Document 1 vector
congress	0.00	1	0.00
approved	0.69	1	0.69
controversial	0.69	1	0.69
bill	0.69	1	0.69
filled	0.69	0	0.00
lies	1.10	0	0.00
deceit	0.69	0	0.00

Table 4.1: B-clfd term representation: Document 1 vector

Vocabulary	clfd weights	b-tf vector (doc 2)	Document 2 vector
congress	0.00	1	0.00
approved	0.69	0	0.00
controversial	0.69	0	0.00
bill	0.69	0	0.00
filled	0.69	1	0.69
lies	1.10	2	1.10
deceit	0.69	1	0.69

Table 4.2: B-clfd term representation: Document 2 vector

The advantages that *b-clfd* vectorization provides are the following:

- A sophisticated term weighting scheme based on the maximum term frequency distance between different class labels in order to calculate the relevance of a term for classification.
- There is little to no sensitivity to stopword selection as the usage of the class label frequency ratio (*clfr*) component acts as a natural filter. The quality of data preprocessing does not significantly affect performance.

Term frequency - class label frequency distance

The second variant is to use the *clfd* vector as a multiplier to a *term frequency* (*tf*) vectorizer. We can multiply the final *clfd* weights with the respective *tf* weights for each term of a document. Compared to *b-clfd* weights, the *tf-clfd* weights will be equal in the case terms do not occur more than once in each document, otherwise, they will be greater and specifically, multiples of the *b-clfd* weights.

In our example, the vocabulary is represented by this vector: [congress, approved, controversial, bill, full, lies, deceit]. After generating the *term frequency* (*tf*) feature vectors, as described in Section 3.2.1, we have [1, 1, 1, 1, 0, 0, 0] as the *tf* feature vector for document 1 and [1, 0, 0, 0, 1, 2, 1] as the *tf* feature vector for document 2. In the end, the resulting *tf-clfd* feature vectors are as follows:

Vocabulary	clfd weights	tf vector (doc 1)	Document 1 vector
congress	0.00	1	0.00
approved	0.69	1	0.69
controversial	0.69	1	0.69
bill	0.69	1	0.69
filled	0.69	0	0.00
lies	1.10	0	0.00
deceit	0.69	0	0.00

Table 4.3: Tf-clfd term representation: Document 1 vector

Vocabulary	clfd weights	tf vector (doc 2)	Document 2 vector
congress	0.00	1	0.00
approved	0.69	0	0.00
controversial	0.69	0	0.00
bill	0.69	0	0.00
filled	0.69	1	0.69
lies	1.10	2	2.20
deceit	0.69	1	0.69

Table 4.4: Tf-clfd term representation: Document 2 vector

The advantages that *tf-clfd* vectorization provides are the following:

- It retains the advantages of *binary class label frequency distance* (*b-clfd*) vectorization.
- It increases the maximum frequency distance between different classes in case a term occurs more than once in a document, thus giving it an even greater classification value.

Term frequency inverse document frequency - class label frequency distance

The third variant is to use the *clfd* vector as a multiplier to a *term frequency - inverse document frequency* (*tf-idf*) vectorizer. We can multiply the final *clfd* weights with the respective (*tf-idf*) weights for each term of a document.

In our example, the vocabulary is represented by this vector: [congress, approved, controversial, bill, full, lies, deceit]. After generating the *term frequency - inverse document frequency* (*tf-idf*) feature vectors, as described in Section 3.2.2, we have [0, 0.3, 0.3, 0.3, 0, 0, 0] as the *tf-idf* feature vector for document 1 and [0, 0, 0, 0, 0.3, 0.6, 0.3] as the *tf-idf* feature vector for document 2. In the end, the resulting *tfidf-clfd* feature vectors are as follows:

Vocabulary	clfd weights	tf-idf vector (doc 1)	Document 1 vector
congress	0.00	0	0.00
approved	0.69	0.3	0.207
controversial	0.69	0.3	0.207
bill	0.69	0.3	0.207
filled	0.69	0	0.00
lies	1.10	0	0.00
deceit	0.69	0	0.00

Table 4.5: Tfidf-clfd term representation: Document 1 vector

Vocabulary	clfd weights	tf-idf vector (doc 2)	Document 2 vector
congress	0.00	1	0.00
approved	0.69	0	0.00
controversial	0.69	0	0.00
bill	0.69	0	0.00
filled	0.69	0.3	0.207
lies	1.10	0.6	0.414
deceit	0.69	0.3	0.207

Table 4.6: Tfidf-clfd term representation: Document 2 vector

The advantages that *tfidf-clfd* vectorization provides are the following:

- It retains the advantages of *term frequency - class label frequency distance* (*tf-clfd*) vectorization.
- There is an even stronger filter against stopwords due to the inverse document frequency (*idf*) component.

4.4 Application to fake news detection

While our *clfd* approach can find application in several practical machine learning or natural language processing tasks, we use it here for fake news detection. As this is a binary classification problem, there are only two possible class labels: credible news (c) and fake news (f). Therefore, for the two classes i={c, f}, Equation 4.1 becomes:

$$clfr^{c}(t) = \log_e \frac{(1 + occ(t, c)) \cdot total(f)}{(1 + occ(t, f)) \cdot total(c)} + 1, \forall t \tag{4.4}$$

$$clfr^{f}(t) = \log_e \frac{(1 + occ(t, f)) \cdot total(c)}{(1 + occ(t, c)) \cdot total(f)} + 1, \forall t \tag{4.5}$$

Furthermore, the resulting *clfr* vectors will now correspond to class labels c and f only. Therefore, Equation 4.2 can be specified for these two classes i={c, f} by simply calculating the absolute value of the subtraction between the two aforementioned *clfr* weights as follows:

$$clfd(t) = |clfr^{c}(t) - clfr^{f}(t)|, \forall t \tag{4.6}$$

Now that we defined the equations for the generation of *clfd* weights for our binary text classification task, we provide an example to describe the process in detail. Consider four documents, the first and the third belonging in the class of credible news (c) and the second and the fourth belonging in the class of fake news (f).

	Documents
1	Congress approved the controversial bill.
2	The congress is filled with lies, lies and deceit.
3	The movie received plenty of controversial reviews.
4	This movie was made by aliens to brainwash us.

Table 4.7: Clfd example: Corpus

The first step is to generate the *cltf* vectors for each class label $i = \{c, f\}$ as described in Algorithm 1. Data preprocessing such as stopword removal and lemmatizing has been applied [27]. In this example, the generated *cltf* vectors, *terms*, *occ* and *total*, are as follows:

	Terms (c)	Occ (c)	Terms (f)	Occ (f)
1	congress	1	congress	1
2	approve	1	fill	1
3	controversial	2	lies	2
4	bill	1	deceit	1
5	movie	1	movie	1
6	receive	1	make	1
7	plenty	1	alien	1
8	review	1	brainwash	1
Total	-	9	-	9

Table 4.8: Clfd example: Class label term frequency (cltf) vectors

The next step is to calculate the *clfr* weights according to Equations 4.4 and 4.5. The resulting *clfr* weight vectors are as follows:

	Vocabulary	Credible news (c)	Fake news (f)
1	congress	0.69	0.69
2	approve	1.10	0.41
3	controversial	1.39	0.29
4	bill	1.10	0.41
5	movie	0.69	0.69
6	receive	1.10	0.41
7	plenty	1.10	0.41
8	review	1.10	0.41
9	fill	0.41	1.10
10	lies	0.29	1.39
11	deceit	0.41	1.10
12	make	0.41	1.10
13	alien	0.41	1.10
14	brainwash	0.41	1.10

Table 4.9: Clfd example: Class label frequency ratio (clfr) vectors

Finally, we calculate the *clfd* weights according to Equation 4.6. The resulting *clfd* weight vector is as follows:

	Vocabulary	clfd weight
1	congress	0.00
2	approve	0.69
3	controversial	1.10
4	bill	0.69
5	movie	0.00
6	receive	0.69
7	plenty	0.69
8	review	0.69
9	fill	0.69
10	lies	1.10
11	deceit	0.69
12	make	0.69
13	alien	0.69
14	brainwash	0.69

Table 4.10: Clfd example: Class label frequency distance (clfd) vector

Clfd Variants The final step is to multiply the *clfd* weight vector calculated above, with a vector generated by a vectorizer of our choice as in Equation 4.3, in order to apply the *clfd* weighting scheme to each document. Our choices in this work were *(i)* a *b-tf* vectorizer, *(ii)* a *tf* vectorizer, and *(iii)* a *tf-idf* vectorizer. This gives rise to three respective *clfd* variants: *b-clfd*, *tf-clfd*, and *tfidf-clfd*.

4.5 Analysis

In the *clfd* weight vector of Table 4.10, we observe that our *clfd* vectorization approach generates a vector of feature importance that is distinct to those that b-tf, tf or tf-idf vectorizers had produced. Specifically, we present the following example for the corpus of Table 4.7. Consider document 3:

	movie	receive	plenty	controversial	review
tf	1.00	1.00	1.00	1.00	1.00
tf-clfd	0.00	0.69	0.69	1.10	0.69

Table 4.11: Clfd example: Comparison of tf and tf-clfd (document 3)

In Table 4.11, we notice that a *tf* vectorizer considers all terms to have an equal importance in document 3. For instance, the term "movie" and the term "controversial" are considered equally important as they both have a weight of 1.00. However, a *tf-clfd* approach considers the term "controversial" very important (weight 1.10), the rest of the terms less important (weight 0.69) and the term "movie" non-existent (weight 0.00).

Therefore, we notice that the term "controversial" has been assigned by clfd a high *clfd* weight as it appears in one more "credible" document (document 1) in the collection. Indeed, "controversial" is a rather elaborate word, more likely to appear in credible news articles. On the other hand, a neutral term such as the term "movie" has a zero *clfd* weight. As such, *clfd* weights could potentially be used as hints to "trace back" and verify whether the terms of importance, according to *clfd*, are indeed likely to appear in their respective document classes.

We can conclude that *clfd* can be a viewed as a tool that could help provide *explainability* of machine learning outcomes, a very important concern in the further analysis of news articles. More specifically, its feature importance vector can help provide useful information which can be used as elements of a system in an explainable AI approach that will provide the reasons behind the classification decision. This comes in contrast to pure neural network approaches where explainability is a big challenge.

Chapter 5

Machine learning methods

The resulting numerical feature vector representations of vectorization techniques are used as input to certain machine learning methods whose task is to classify a news article into credible news or fake news. The methods which are used can be categorized into probabilistic, ensemble and deep learning methods. Furthermore, a *hybrid* method is developed and proposed for better performance. In the following sections, the functionality of these methods, as well as their advantages and disadvantages, is described in detail.

5.1 Probabilistic methods

Probabilistic machine learning methods are methods which rely on a probabilistic approach to classification. The goal of this approach is to capture the relationship between the input data and the output class label by comparing the probability of an event in the presence of another event. For example, the probability of having a fire (event A) if the weather is hot (event B). There may be several variables which event A depends on, such as the case of the weather being windy (event C). In the end, a probabilistic method is able to predict, given an observation of an input, a probability distribution over a set of class labels.

5.1.1 Multinomial Naive Bayes

The first probabilistic method that was used was a *Multinomial Naive Bayes* classifier. This method is based on the Bayes' theorem and thus adopts independence assumptions regarding the features. Because of this assumption, possible correlations between the input features are overlooked during classification. The multinomial based *Naive Bayes* classifier regards the input feature vectors as representations of the frequencies which certain events, terms of a document in our case, have been generated by a multinomial $(p_1, p_2, ..., p_n)$ where p_i is the probability that term i occurs in the document. Assuming that a feature vector $x = (x_1, x_2, ..., x_n)$ represents the number of occurrences of all terms, with x_i counting the number of times term i was observed in a particular document, the likelihood of a term i belonging to class k (p_{ki}), as well as the likelihood of the entire feature vector x belonging to class k ($p(x|C_k)$), are calculated as in Equations 5.1 and 5.2 that are presented in the following page.

$$p_{ki} = p(x_i|C_k) = \frac{N_{ki} + a}{\Sigma_{j=1}\, N_{kj} \;\; + a \cdot n} \tag{5.1}$$

$$p(x|C_k) = \frac{\Sigma_{i=1}\; x_i\,!}{\Pi_{i=1}\; x_i\,!} \cdot \Pi_{i=1}\; {p_{ki}}^{x_i} \tag{5.2}$$

In Equation 5.1, N_{ki} stands for the number of occurrences of term i in class k and n stands for the total amount of features which equals the vocabulary size. The hyperparameter alpha (a) is named smoothing priors and handles features not present in the learning samples while at the same time prevents the computation of zero probabilities. In Equation 5.2, x_i represents the number of occurrences of term i in a particular document and p_{ki} is the likelihood of that term i belonging to class k as calculated in Equation 5.1.

As an example, consider the following document: "We are going really really fast". Assuming that a *term frequency* (tf) vectorization is used and all p_{ki} have been calculated, a *Multinomial Naive Bayes* classifier would calculate the probability of this document belonging to class k, $p(x|C_k)$, according to Equation 5.2 as follows:

$$\begin{aligned} p(we, are, going, really, really, fast|C_k) = \\ p(we = 1, are = 1, going = 1, really = 2, fast = 1|C_k) = \\ \frac{6!}{2!} \cdot p(we|C_k) \cdot p(are|C_k) \cdot p(going|C_k) \cdot p(really|C_k)^2 \cdot p(fast|C_k) \end{aligned}$$

Even though the *Naive Bayes* method does not take possible correlations between the input features into account, it provides a good baseline method for text classification. The advantages it provides are the following:

- It is a simple but also fast and accurate method for prediction.
- It has a very low computational cost and can efficiently perform on a large dataset.
- It performs particularly well in the field of text analytics specifically.

However, the disadvantages are the following:

- The assumption of independent features, because in practice it is almost impossible that the input will be a set of features which are entirely independent. As a result, the accuracy of this method is affected negatively.
- If there is no training set of a particular class, this causes zero posterior probability and in this case, the classifier is unable to make predictions.

In our fake news detection system, we used the *Multinomial Naive Bayes* classification method and performed hyperparameter optimization by using grid search and comparing the results in order to empirically discover a good value for the alpha hyperparameter. The value of alpha which provided the best results was kept in order to maximize the performance of the *Naive Bayes* method for each dataset.

5.1.2 Logistic Regression

Another probabilistic method which was used was *Logistic Regression* [19]. This method attempts to optimize a function to fit the available data best and uses the maximum likelihood estimation (*MLE*) criterion. The *MLE* criterion determines the parameters that are most likely to produce the observed data by setting the mean and variance as parameters in determining the specific parametric values for a given model. The function maps any real prediction values to probability values which range between 0 and 1 and represent the probability of belonging to a certain class label. For binary classification, the threshold for the probability values is 0.5, therefore a higher probability ($p > 0.5$) indicates that the document is likely to belong to a certain class k, while a lower probability ($p < 0.5$) indicates that the document is likely to belong to a class different from class k. In order to calculate the probability values, the method follows the logistic sigmoid function which is as follows:

$$z = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + ... + e \tag{5.3}$$

$$S(z) = \frac{1}{1 + e^{-z}} \tag{5.4}$$

In Equation 5.3, β stands for the regression coefficients which are estimated through the maximum likelihood estimation (*MLE*) criterion and aims to provide a clear division between the probabilities of the class values. Furthermore, x_i are the respective features of our document feature vectors and e is an error bias. The resulting $sigmoid(z)$, or $S(z)$, value is a probability value between 0 and 1 and is calculated as in Equation 5.4. A figure [33] which clearly shows the logistic sigmoid function as well as the threshold for binary classification is presented below.

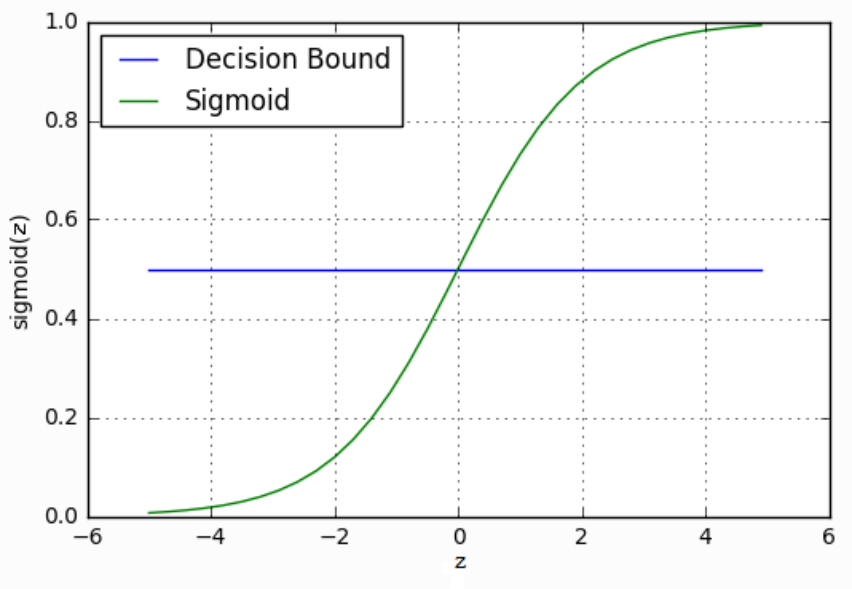


Figure 5.1: Logistic sigmoid function

In our fake news detection system, we used the *Logistic Regression* classification method and performed hyperparameter optimization by using grid search. After comparing the results, the default values were kept as they achieved a very high performance in all datasets.

5.2 Ensemble methods

Ensembles methods [19] are techniques which create multiple classifiers and then combine them to produce better results than any of the single classifiers individually. In our fake news detection system, we used the *Random Forest*, *Gradient Boosting* (*XGB*) and *Adaptive Boosting* ensemble methods. The main hyperparameters of ensemble methods include the number of decision trees, the depth of each decision tree and the learning rate.

Figure 5.2: Ensemble methods pipeline

The advantages of the ensemble methods are the following:

- High accuracy due to the combination of different classifiers.
- Acceptance of both categorical and numerical values in training data.
- Handling of missing data and no need for prior scaling of the training data.
- Generation of a reliable feature importance estimate.

5.2.1 Random Forest

The *Random Forest* method is based on *bootstrap aggregating*, also called *bagging*, which is a technique that first splits training data into numerous subsets. Afterwards, each subset is taken as input to a decision tree classifier and a prediction is made. Finally, the predictions of each decision tree classifier are aggregated to output the actual prediction. As an extra step to reduce overfitting, each decision tree only utilizes a randomly selected subset of features. An illustration [34] of the bootstrap aggregating technique used by the *Random Forest* method is presented below.

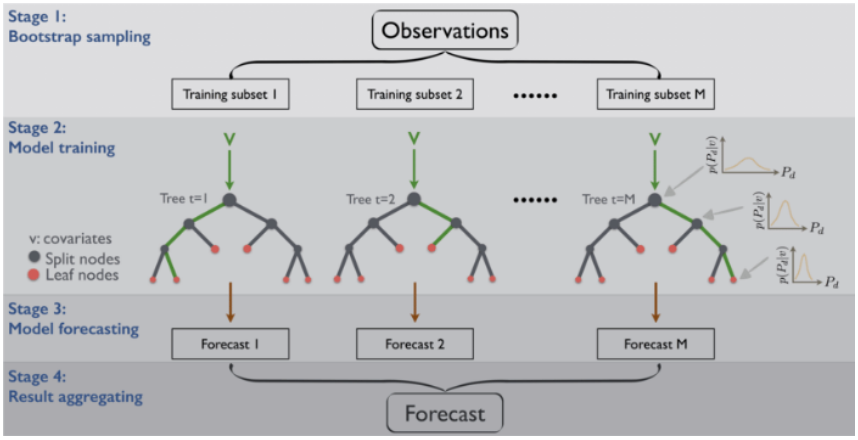


Figure 5.3: Bootstrap aggregating

The advantages the *Random Forest* method provides, in comparison to other ensemble methods, are the following:

- High accuracy and a significantly faster classification process due to parallelization during training.
- Reduction of variance and overfitting due to the bagging process and the random feature selection for each decision tree.
- Handling of outliers and noise which may appear in the distribution which significantly reduces their impact on classification.

The disadvantages of the *Random Forest* method, in comparison to other ensemble methods, are the following:

- Unweighted voting. All classifiers are assumed to be equally important for the final prediction.
- Independent classifiers which do not learn from each other's classification results.

In our fake news detection system, we used the *Random Forest* classification method and performed hyperparameter optimization by using grid search and comparing the results in order to empirically discover a good value for each hyperparameter. The number of decision trees was set to a value which doesn't affect training time significantly, the depth of each decision tree was assigned a very high value (no limit) and the learning rate was set to the default value. This way, we have a few strong decision tree classifiers who train in parallel and whose combination makes the *Random Forest* method perform very well in all datasets.

5.2.2 Gradient Boosting & Adaptive Boosting

The *Gradient Boosting* and the *Adaptive Boosting* classification methods are methods which also split training data into subsets and then combine several weak classifiers to create a strong classifier. This is done by focusing on their mistakes through *boosting*, which is an iterative technique that adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it increases the weight of this observation and vice versa. Compared to bootstrap aggregating, *boosting* is different in the following ways:

- The partitioning of the data into subsets is random in *bootstrap aggregating*, while *boosting* gives incorrectly classified samples a higher preference.
- Each classifier is independent in *bootstrap aggregating*, while *boosting* makes each classifier dependent on the previous ones.
- The combination of the weak classifiers into a strong one is done by calculating the average probability in *bootstrap aggregating*, while *boosting* utilizes a weighted majority vote.

An illustration [35] of the *boosting* technique used by the *Gradient Boosting* and *Adaptive Boosting* methods is presented below in a binary classification example which consists of a positive (+) and and negative (-) class label.

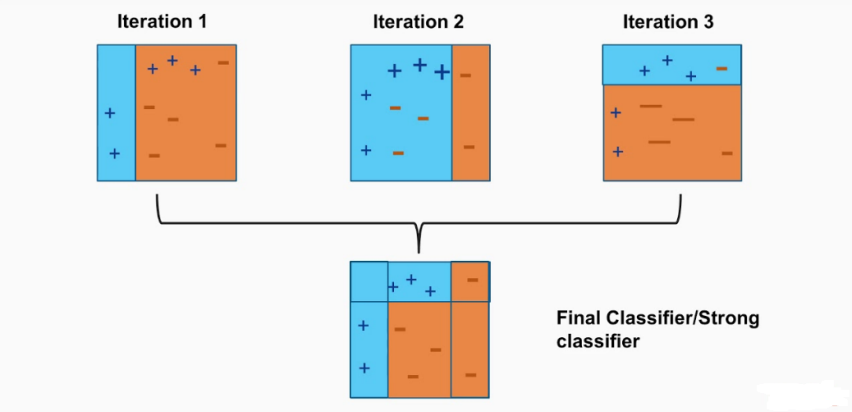


Figure 5.4: Boosting

Regarding the two boosting methods used, the main difference is that *Adaptive Boosting* adds the weak classifiers to the strong one according to their performance, calculated by an alpha-weight, while *Gradient Boosting* adds the weak classifiers to the strong one by using a gradient descent optimization process. The advantages the *Gradient Boosting* and *Adaptive Boosting* methods provide, in comparison to other ensemble methods, include:

- High accuracy due to the weighted contribution of each decision tree classifier to the final prediction as well as the fact that boosting adds new classifiers that perform well where previous classifiers have failed.

The disadvantages of the *Gradient Boosting* and *Adaptive Boosting* methods, in comparison to other ensemble methods, include:

- High time complexity. The process is sequential as each classifier is influenced by the previous ones, therefore training is iterative and cannot be done in parallel.
- High computational cost. A large number of decision trees is required and each decision tree is dependent on the previous ones.
- Prone to overfitting if cross-validation is not performed.

In our fake news detection system, we used the Gradient Boosting and Adaptive Boosting classification methods and performed hyperparameter optimization by using grid search and comparing the results in order to empirically discover a good value for each hyperparameter. The conclusion was that the number of decision trees was assigned a very high value and the learning rate was set to the default value. This way, we have a plethora of relatively weak decision tree classifiers who train sequentially and learn from each other's mistakes. Their large number makes the *Gradient Boosting* and the *Adaptive Boosting* methods achieve a high performance in all datasets.

5.3 Deep learning methods

Deep learning is a machine learning technique which contains algorithms inspired by the structure and function of the human brain called artificial neural networks (ANNs). As a term, it describes an algorithm that employs ANNs with more than 3 layers. An ANN is based on a collection of connected units called artificial neurons. These neurons are aggregated into layers and each layer performs a different kind of transformation on its input. Each connection can transmit a signal from an artificial neuron of one layer to that of another. Signals travel from the first layer (the input layer), to other (hidden) layers and finally to the last layer (the output layer). An ANN learns by adjusting the weight, a real number, of each connection which results in the increase or decrease of the strength of the signal at that particular connection. Finally, the output of each artificial neuron is computed by a non-linear function of the sum of its inputs.

A recurrent neural network (RNN) is a type of artificial neural network which contains an internal state (memory) to process sequences of inputs. In our case, this means that an RNN takes into account the order of the terms contained in the news article and processes them sequentially. Thus, unlike the previous bag of word methods, the order matters as the news article text is considered a sequence and each term affects the terms which come after it. The main characteristic of the RNNs, which is their memory component, makes them suitable for various sequential natural language processing tasks such as text classification.

In our fake news detection system, we attempted several RNN based architectures [36]. We will outline the architecture of each method and then describe its functionality in detail in the following subsections.

We first provide an overview of the architectures of the deep learning methods that were developed for our fake news detection system before explaining them in detail. Word embeddings were used as input into four deep learning methods. We used pre-padding of zeroes and did not limit the maximum input length or the maximum size of the vocabulary in order to achieve maximum performance despite the cost in computational resources. The four deep learning methods, along with their respective architectures, are the following:

1. Long short-term memory neural network (lstm)

   The architecture of this deep learning method begins with an embedding layer followed by a dropout layer in order to reduce overfitting. Afterwards, an lstm layer follows and its output is passed into a dense layer with a rectified linear unit (ReLU) activation function. Finally, the classification is done on a dense layer with a sigmoid activation function.

2. Bidirectional long short-term memory neural network (Bi-lstm)

   The architecture of this deep learning method begins with an embedding layer followed by a dropout layer in order to reduce overfitting. Afterwards, a Bi-lstm layer follows and its output is passed into a dense layer with a rectified linear unit (ReLU) activation function. Finally, the classification is done on a dense layer with a sigmoid activation function.

3. Combined convolutional and long short-term memory neural network (cnn+lstm)

   The architecture of this deep learning method begins with an embedding layer followed by a dropout layer in order to reduce overfitting. Afterwards, there are three repetitions of the combination of an 1-dimensional convolutional layer and an 1-dimensional max pooling layer. The resulting output is passed into an lstm layer followed by a dense layer with a rectified linear unit (ReLU) activation function. Finally, the classification is done on a dense layer with a sigmoid activation function.

4. Multiple long short-term memory layers (mult.lstm)

   The architecture of this deep learning method begins with an embedding layer followed by a dropout layer in order to reduce overfitting. Afterwards, the first lstm layer follows and its output is passed, through a repeat vector layer, into a second lstm layer. The output of the second lstm layer is passed into a dense layer with a rectified linear unit (ReLU) activation function. Finally, the classification is done on a dense layer with a sigmoid activation function.

5.3.1 Long short-term memory neural network

A long short-term memory neural network (*LSTM*) is a type of deep learning method which is widely popular in natural language processing and has also been very successful in previous work in the field of fake news detection research [15]. The core components of an *LSTM* neural network are the cell state and the various gates. It is also composed of two states C_t and h_t which represent the cell state and the hidden state accordingly. The architecture is presented in Figure 5.5 [37] and all of the components are described below in detail.

- **Forget Gate** (f_t). This gate decides what information should be kept or thrown away. Information from the previous hidden state h_{t-1} and information from the current input x_t are passed through a sigmoid function. The resulting values are in a range from 0 to 1. The values which are closer to 0 are forgotten, and the values which are closer to 1 are kept.

- **Input Gate** (i_t). This gate is used to update the cell state. Firstly, we pass the previous hidden state h_{t-1} and the current input x_t into a sigmoid function. The result of this function i_t decides which values will be updated by transforming the values to be in a range from 0 to 1. The values which are closer to 0 are not important, while the values which are closer to 1 are important. The hidden state $\hat{C}_t$ and current input x_t is also passed into a tanh function in order to transform the values in a range from -1 to 1 which help regulate the network. Then there is a multiplication of the tanh output with the sigmoid output and the resulting value shows the importance of the information.

- **Cell State** (C_t). This is the cell state which represents the memory of the neural network. Firstly, pointwise multiplication is performed between the previous cell state C_{t-1} and the forget vector. This multiplication has a possibility of dropping values in the cell state if it gets multiplied by a forget vector which contains values close to 0. Afterwards, a pointwise addition is performed between the newly calculated cell state and the input state. This addition updates the cell state to new values that the neural network finds relevant. The result is the new cell state C_t.

- **Output Gate** (o_t). This gate decides what the next hidden state should be. The hidden state contains information on previous inputs and is also used for predictions. Firstly, the previous hidden state h_{t-1} and the current input x_t are passed into a sigmoid function. Then the newly modified cell state $\hat{C}_t$ is passed to the tanh function. Finally, there is a multiplication of the tanh output with the sigmoid output to decide what information the hidden state should contain. The output is the hidden state h_t, while the new cell state C_t and the new hidden state h_t are carried over to the next time step.

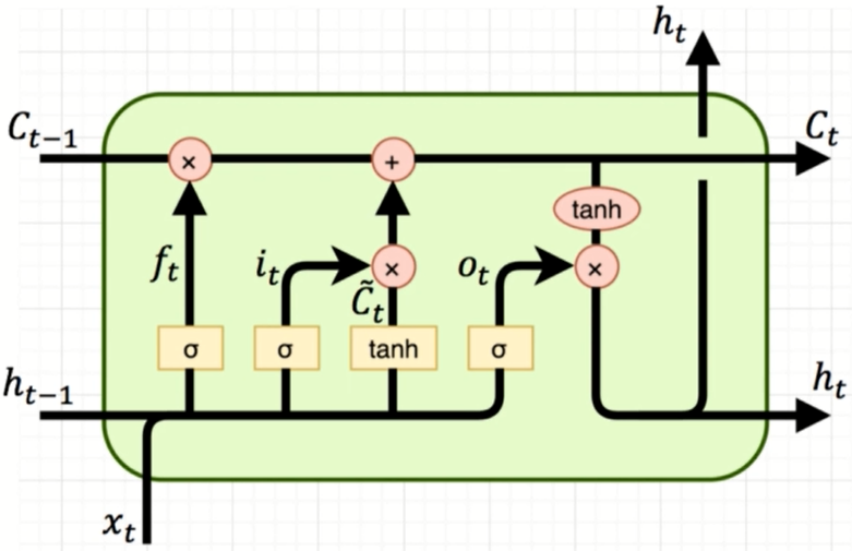


Figure 5.5: Long short-term memory neural network

$$f_t = \sigma({W_f}^T \cdot [x_t, h_{t-1}] + b_f) \tag{5.5}$$

$$i_t = \sigma({W_i}^T \cdot [x_t, h_{t-1}] + b_i) \tag{5.6}$$

$$o_t = \sigma({W_o}^T \cdot [x_t, h_{t-1}] + b_o) \tag{5.7}$$

$$\hat{C}_t = \tanh({W_c}^T \cdot [x_t, h_{t-1}] + b_c) \tag{5.8}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \tag{5.9}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{5.10}$$

In the above equations, f_t stands for the forget gate (Equation 5.5), i_t stands for the input gate (Equation 5.6), o_t stands for the output gate (Equation 5.7), $\hat{C}_t$ stands for the candidate cell (Equation 5.8), C_t stands for the cell state (Equation 5.9) and h_t stands for the hidden gate (Equation 5.10).

In conclusion, the cell state C_t acts as a transport highway that transfers relevant information throughout the processing of the sequence and represents the memory of the neural network. Regarding the gates of an *LSTM*, the forget gate f_t decides what is relevant to keep from prior steps, the input gate i_t decides what information is relevant to add from the current step, and the output gate o_t determines what the next hidden state should be.

In our fake news detection system, an *LSTM* neural network was used as the baseline deep learning method as its utility of preserving information throughout a sequence has made it achieve the most promising results in previous related work in the field of fake news. Furthermore, we attempted a neural network architecture of multiple *LSTM* layers, as deeper RNN architectures empirically show better results than shallow ones [38].

5.3.2 Bidirectional long short-term memory neural network

As described above, *LSTM* preserves information from inputs which have already passed through it by making use of the hidden state. However, the preserved information consists of terms which precede the current term as the text is processed sequentially. A bidirectional long short-term memory neural network (*Bi-LSTM*) will process the text sequentially in two ways, resulting in preserving both information which precedes and information which follows the current term. Consider the following example:

"The students went to the pool and enjoyed swimming."

For the term "pool", an *LSTM* would utilize all preceding information which corresponds to the following terms: ["The", "students", "went", "to"]. On the other hand, a *Bi-LSTM* would utilize information both before and after the term "pool" by performing a concatenation of the two. That information corresponds to the following terms: ["The", "students", "went", "to", "the", "and", "enjoyed", "swimming"]. By being able to see both the past and the future context of a term, a *Bi-LSTM* has the advantage of providing more information to the neural network and often achieves higher accuracy. The downside is the increased cost in computational time and memory consumption.

In our fake news detection system, we used a *Bi-LSTM* neural network as a natural improvement over the baseline deep learning method of an *LSTM* neural network.

5.3.3 Convolutional long short term memory neural network

Convolutional neural networks (*CNNs*) [36], also called convnets, are widely used in image classification and computer vision because they able to extract features from images and use them in neural networks. They are specialized neural networks that are able to detect specific, complex patterns. In fact, repeated hidden (convolutional) layers increase the complexity of the patterns the neural network can detect. The core of the technique used by CNNs is the mathematical process of convolution or more specifically, a slight variant of convolution called cross correlation. This process compares the input signal with a filter and outputs a high value (spike) in case of a high correlation between the filter and the input signal. An example of cross correlation between the input signal and a filter is presented below in Figures 5.6 and 5.7 [37].

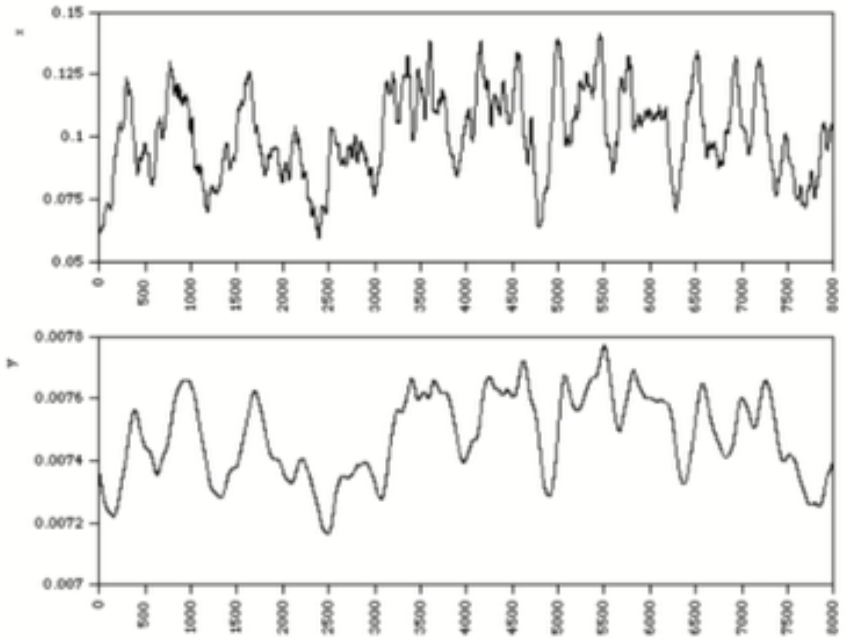


Figure 5.6: Input signal (up) and Filter (down)

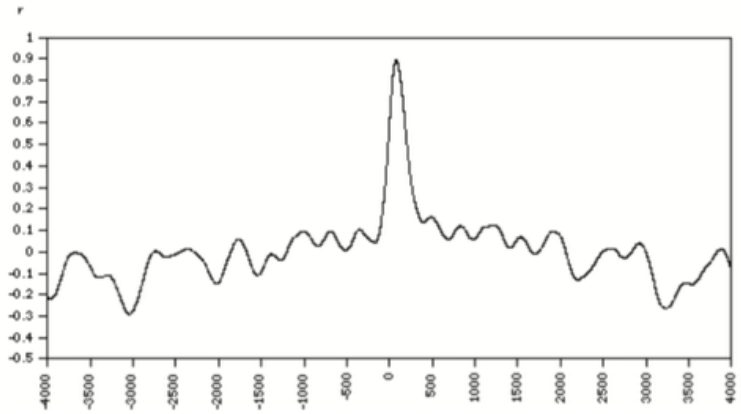


Figure 5.7: Output of cross correlation

In Figure 5.7, we notice a spike around the value of 0. This spike around zero means that the input signal and the filter match when they are both aligned exactly in the center, as shown in Figure 5.6.

Since fake news detection is a text classification task, the text of a news article, which is represented by a sequence of vectors (word embeddings), is treated as an one dimensional vector signal. Therefore, the neural network utilizes one dimensional convolutions which are capable of capturing complex patterns in a sequence of text.

The feature extraction process of an one dimensional convolution is simple. Firstly, a patch of input features of size equal that of the filter kernel is taken. The output feature is the dot product of this patch and the multiplied weights of the filter. An illustration of the feature extraction process is presented in Figure 5.8 [37].

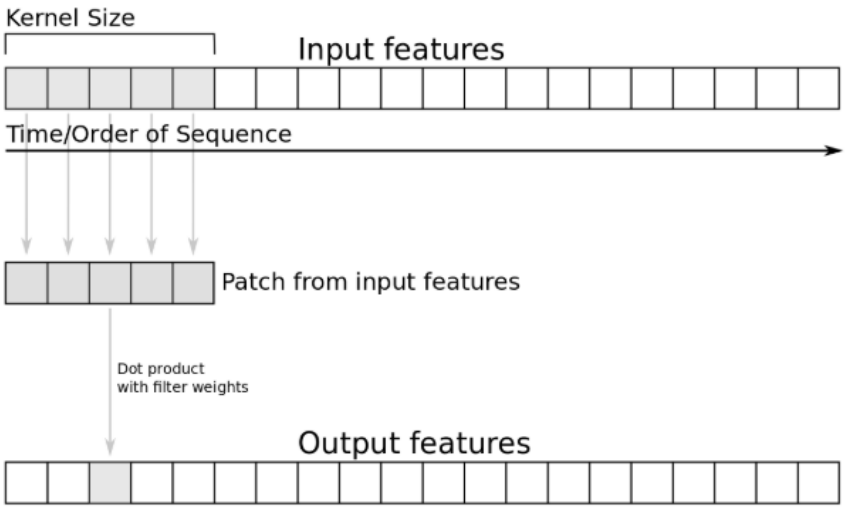


Figure 5.8: Convolutional neural networks - Feature extraction

Convolutional neural networks (CNN), and the one dimensional CNN in particular, also have the characteristic of translation (spatial) invariance, which allows the neural network to generalize and recognize certain sub-sequences regardless of the position of each of their components, terms in our case. This is an important advantage and it happens due to the max pooling layers that follow the convolutional layers in the neural network architecture (see Figure 5.9 [37]) and perform a downsampling operation. As a result, the neural network is capable of generalizing and detecting complex features. Therefore, repeated combinations of convolutional and max pooling layers often increase performance. Finally, due to the max pooling layer, the degree of overfitting as well as the computational time and the memory consumption of the neural network operations are reduced.

Since we want to predict if a sequence of text (news article) is part of a specific label (credible news or fake news), the architecture of our convolutional neural network also includes a dense layer in the end with a sigmoid activation function.

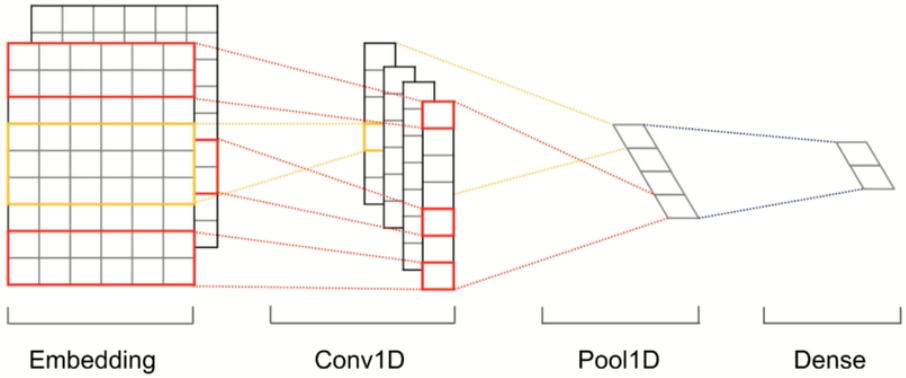


Figure 5.9: Convolutional neural network for classification

In our fake news detection system, we combined a *CNN* and a *LSTM* in order to retain the advantages of both neural networks. After a repeated combination of convolutional and max pooling layers, an LSTM layer was added, followed by two dense layers. This architecture has the following advantages:

- Low computational cost in both training time and memory consumption.
- Preservation of relevant information throughout the sequence.
- Spatial invariance and detection of complex patterns.
- Reduced overfitting.
- Very high and consistent accuracy.

5.4 Hybrid method

In addition to the previous machine learning methods, we developed a method, named *hybrid* method, which combines machine learning and information retrieval techniques in order to classify a news article into credible or fake news. For the first part, we created a stacking method which utilizes the prediction results of two machine learning classification methods through an average probability system in order to increase classification performance. For the second part, the title of the news article is compared to a plethora of titles from news articles which are collected from certain credible online sources. In case of a high title similarity, a positive bias is added towards credible news during classification. In the following subsections, we will describe the two components of the *hybrid* method in detail.

5.4.1 Stacking

As described in Section 5.2, the idea behind ensemble methods is the combination of several classifiers into a more effective one. This reasoning has been present in the field of fake news detection research, as described in chapter 2.1, in the form of combining two different deep learning methods through a multilayer perceptron [17].

Therefore, prompted by the observed performance of the methods above during our experimental evaluation, we developed an algorithm which combines the representation results of two classifiers: *(i)* the traditional machine learning method of *Logistic Regression* utilizing *b-clfd* feature vectors (l), and *(ii)* the deep learning method of *cnn+lstm* mentioned above (c). We first calculate the classification probabilities of these two machine learning methods for each class label i. The *hybrid* method combines these probability vectors by calculating the *average probability* h_i of each news article d to belong to class label i, as follows:

$$h_i = (l_i + c_i)/2 \tag{5.11}$$

In Equation 5.11, l_i and c_i denote the probability assigned to d belonging to i by the l and c methods, respectively. In our binary text classification task of fake news detection, we only have two h values for each d document, h_c and h_f, which represent the probability of d belonging to the class label of credible or fake news respectively. Since $h_c + h_f = 1$, we classify d into credible news if $h_c > 0.5$ or into fake news if $h_f > 0.5$. Notice that the hybrid method's extension to non-binary classification problems is entirely straightforward.

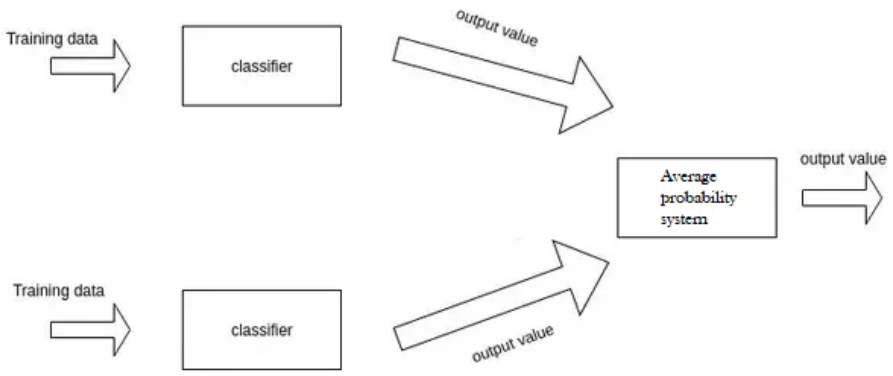


Figure 5.10: Hybrid method - Stacking

5.4.2 Title similarity

The second component of the *hybrid* method depends on information retrieval techniques that fetch the title of news articles taken from the online *New York Times* [39] and *The Guardian* [40] newspapers. After retrieving all relevant titles through a query search, a title similarity score is calculated in order to identify whether the title of a news article also exists in the online news repositories of the aforementioned credible newspapers.

In order to calculate the title similarity score, we use certain techniques such as cosine similarity and word mover's distance [41], both of which are based on a word2vec model trained on a 3.5 GB Google News corpus [42]. Finally, a next sentence prediction BERT model [43] was deployed for the same task. Only if all three metrics show a high similarity between two news article titles, a positive bias towards credible news is added.

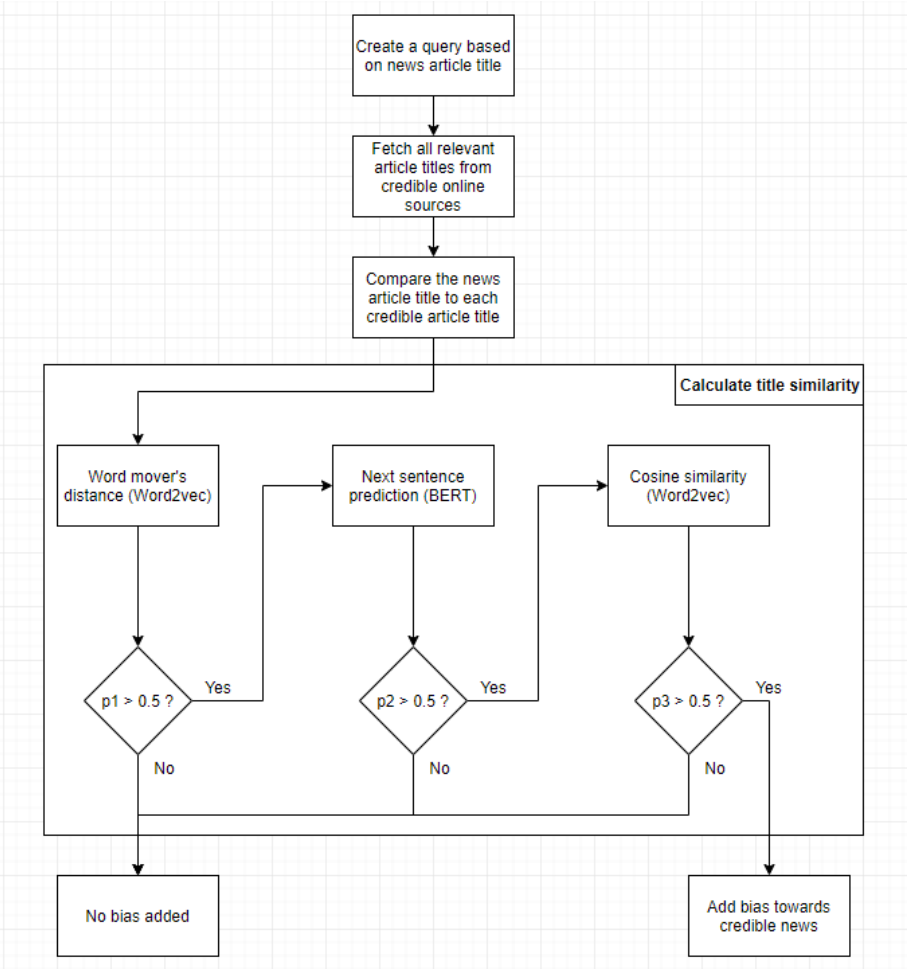


Figure 5.11: Hybrid method - Title similarity

However, an important drawback of this technique is the time that is required for the retrieval of the online corpus that is consisted of the titles of news articles from the aforementioned credible online sources. Even with a short date range, several minutes are required for the classification of a single news article. Thus, despite the promising results that this technique showed during testing, it could not be used for the classification of our datasets which contain thousands of news articles. Therefore, in the experimental evaluation, we only used the stacking component, as described in Section 5.4.1, for the *hybrid* method.

Chapter 6

Evaluation

In this chapter, we experimentally evaluate both our vectorization and machine learning methods, on three datasets which differ in size, class balance and data homogeneity. For the experiments, we have built a fake news detection system in the Python programming language, utilizing libraries such as NumPy, pandas, regex, nltk [44], scikit-learn [45] and keras [46]. The experiments were conducted on a virtual environment, Google Colab, which is a free online Jupyter Notebook environment. It provides 12.72 GB of RAM, 48.97 GB of disk space, and most importantly, a Tesla K80 GPU which increases computational effectiveness. The metrics used for the evaluation of our experiments were accuracy, precision, recall and F-1 score [47].

6.1 Corpora

Label	Dataset 1	Dataset 2	Dataset 3
Fake	3164	10369	24396
Credible	3171	10349	13614
Total	6335	20718	38010

Table 6.1: Structure of the datasets

Our first dataset (Dataset 1) was George Mcintyre's dataset [48] which contains 6335 news articles, evenly distributed between the two classes. The main attribute of this dataset is its homogeneity, as the 2016 US election news is the common topic of all articles.

Our second dataset (Dataset 2) was a Kaggle dataset [49] which contains 20718 news articles, evenly distributed between the two classes. This open sourced dataset has been reviewed by the Kaggle community and contains credible and fake news articles taken from the Web. Compared to the first dataset, apart from the difference in size, this dataset provides news articles about various topics which will show how our algorithms and methods work in non-homogeneous data.

Our third dataset (Dataset 3) contains 38010 news articles, and is the union of the previous two datasets and another Kaggle dataset of approximately 13000 fake news articles [50]. Adding such a number of fake news articles had the effect of making this dataset large and imbalanced, as it now contains a greater quantity of fake than credible news. Dataset 3 is also even more heterogeneous, as it is composed of three datasets which not only have news articles about various topics, but they are also taken from various different sources.

6.2 Evaluation Metrics

The metrics used for the evaluation of our experiments were accuracy, precision, recall and F-1 score. These evaluation metrics are defined as follows:

Accuracy is defined as the number of correct predictions divided by the total number of predictions. Its value is calculated as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of predictions made}}$$

For the other evaluation metrics, we need to define four important terms.

True Positives: The cases in which we predicted FAKE and the actual output was also FAKE.

True Negatives: The cases in which we predicted CREDIBLE and the actual output was CREDIBLE.

False Positives: The cases in which we predicted FAKE and the actual output was CREDIBLE.

False Negatives: The cases in which we predicted CREDIBLE and the actual output was FAKE.

	Predicted: CREDIBLE	Predicted: FAKE
Actual: CREDIBLE	True Negatives	False Positives
Actual: FAKE	False Negatives	True Positives

Table 6.2: Confusion matrix

The metrics based on the confusion matrix (Table 6.2) are precision, recall and F-1 score. We provide a brief definition for each of them:

- Precision is defined as the number of correct positive results divided by the number of positive results predicted by the classifier. Its value is calculated as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Recall is defined as the number of correct positive results divided by the number of all samples that should have been identified as positive. Its value is calculated as follows:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- F-1 score is defined as the Harmonic Mean between precision and recall. It shows how precise the classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). Its value can be calculated by using one of the following two ways:

$$\text{F-1 score} = \frac{\text{Precision x Recall}}{\text{Precision} + \text{Recall}} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Our goal is to maximize all of the aforementioned metrics (accuracy, precision, recall, F-1 score) and thus achieve a consistently high performance for our binary text classification task of fake news detection.

6.3 Comparison of vectorization techniques

6.3.1 Evaluation on the first dataset

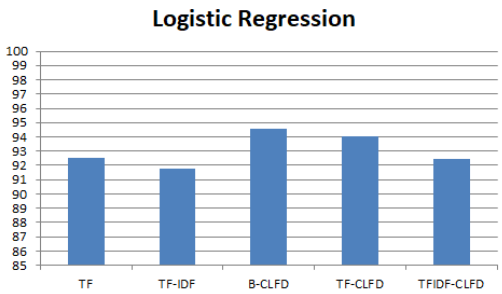
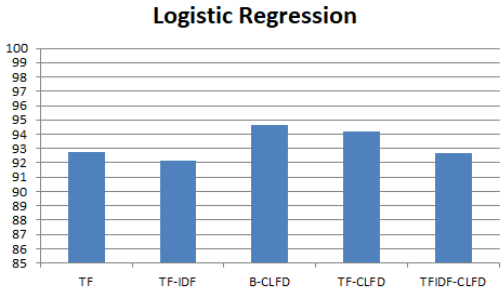
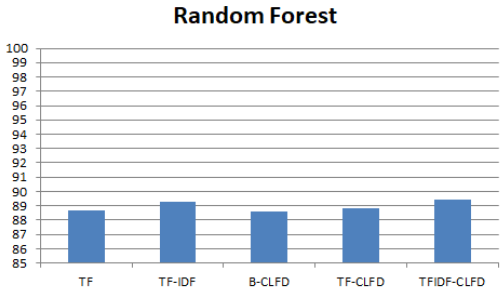
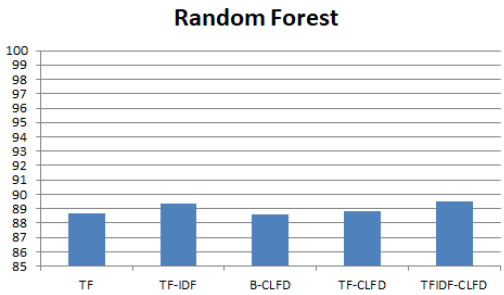
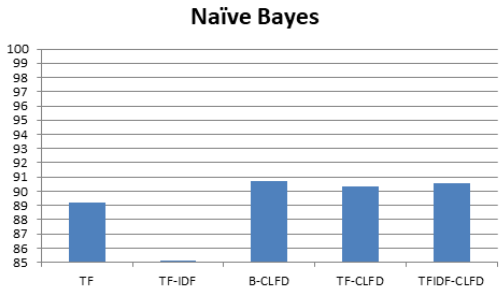
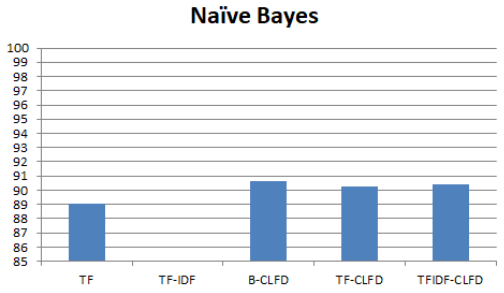
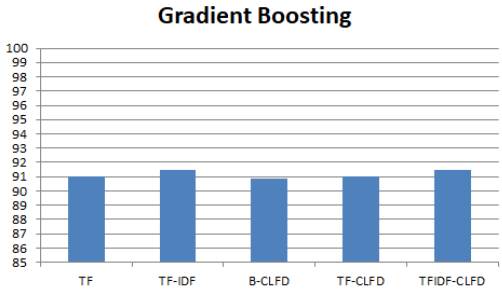
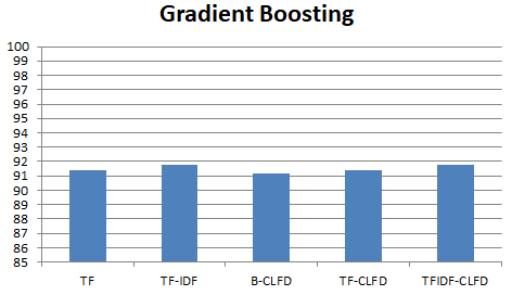
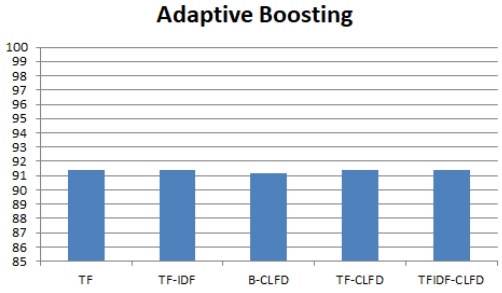
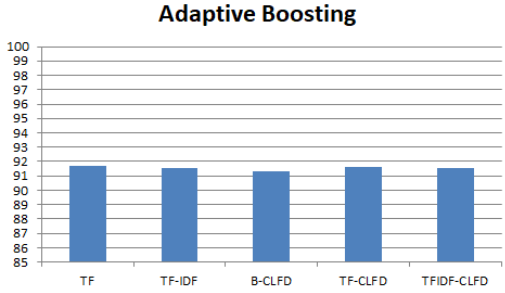
Accuracy	F-1 score
	
	
	
	
	

Table 6.3: Dataset 1: Accuracy and F-1 score

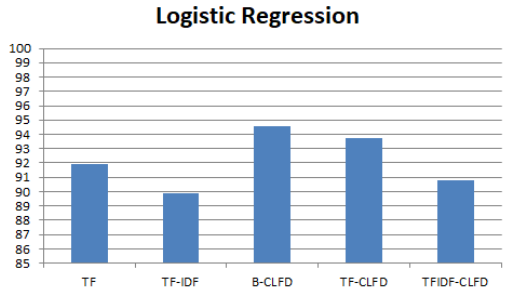
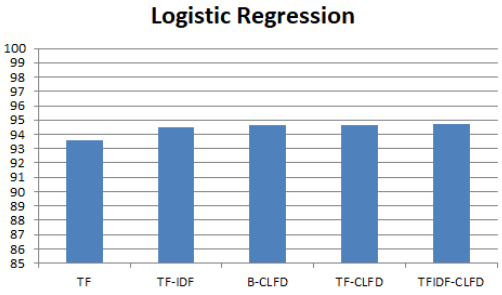
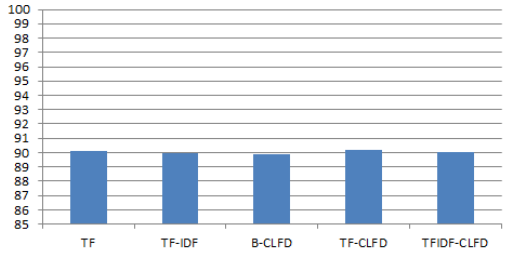
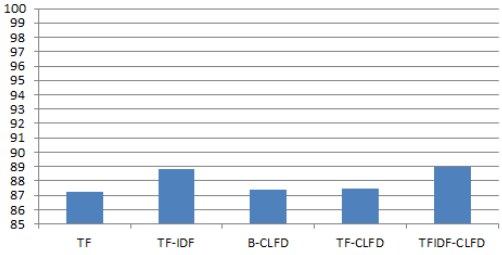
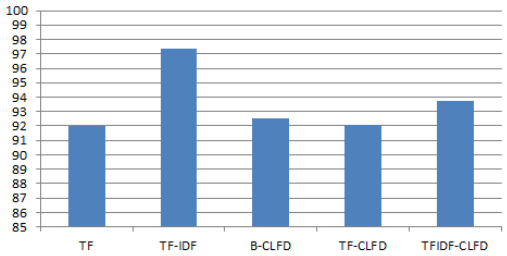
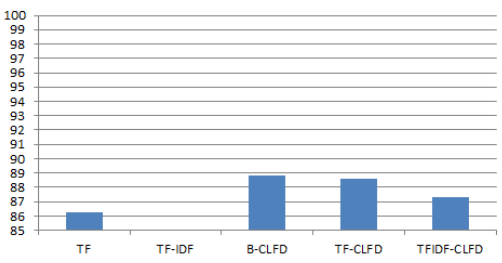
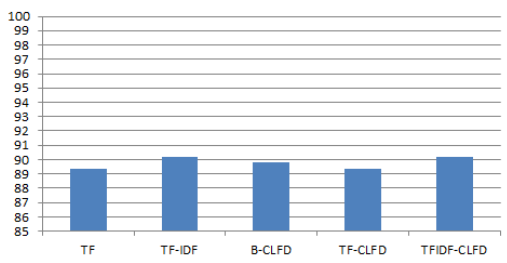
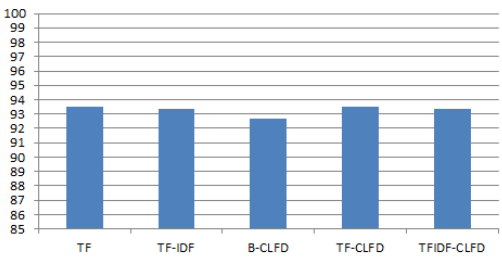
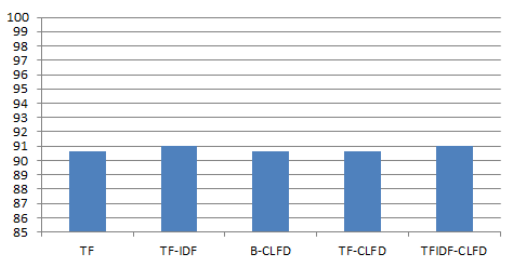
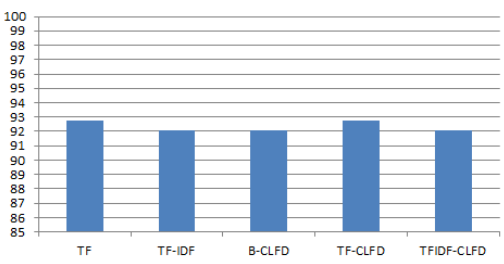
Precision	Recall
	
	
	
	
	

Table 6.4: Dataset 1: Precision and Recall

Accuracy

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	92.55	88.65	89.21	91.02
tf-idf	91.79	89.27	84.33	**91.46**
b-clfd	**94.53**	88.56	**90.69**	90.89
tf-clfd	94.05	88.79	90.31	91.02
tfidf-clfd	92.42	**89.4**	90.57	**91.46**

Precision

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	91.92	90.12	92.01	89.35
tf-idf	89.9	89.96	**97.38**	**90.22**
b-clfd	**94.6**	89.85	92.56	89.78
tf-clfd	93.74	**90.21**	92.05	89.35
tfidf-clfd	90.77	90.07	93.7	**90.22**

Recall

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	93.57	87.24	86.24	**93.51**
tf-idf	94.48	88.8	71.05	93.35
b-clfd	94.64	87.37	**88.8**	92.66
tf-clfd	94.61	87.47	88.57	**93.51**
tfidf-clfd	**94.71**	**88.97**	87.31	93.35

F-1 score

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	92.73	88.64	89.02	91.37
tf-idf	92.13	89.37	82.15	**91.74**
b-clfd	**94.61**	88.59	**90.63**	91.18
tf-clfd	94.16	88.8	90.27	91.37
tfidf-clfd	92.69	**89.5**	90.38	**91.74**

Table 6.5: Dataset 1: Performance of vectorization methods

In the experiments conducted on the first dataset, as presented in Tables 6.3, 6.4 and 6.5, we can make the following observations:

- The ensemble machine learning methods, *Random Forest*, *Gradient Boosting* and *Adaptive Boosting*, do not show any significant difference in the performance with different vectorization techniques.

- The utilization of *clfd*-based feature vectors provides better results for the probabilistic machine learning methods, *Logistic Regression* and *Naive Bayes*, compared to the results provided by traditional vectorization techniques for those methods. More specifically, comparing *clfd*-based vectorization to the next best performing traditional vectorization technique, there is an approximately 1.5% to 2% increase in performance across all metrics.

- There is a significant boost in the performance of the *Naive Bayes* method when we compare the results of *tf-idf* with those of its *clfd* variant, *tfidf-clfd*. More specifically, there is a 6% to 16% increase in performance if *tfidf-clfd* feature vectors are used instead of *tf-idf* ones.

- The only exception is the precision score of the *Naive Bayes* method which is higher with *tf-idf* feature vectors and the recall score of *Logistic Regression* which shows no significant difference with *clfd*-based and *tf-idf* feature vectors.

- The best performing method is *Logistic Regression* with *b-clfd* feature vectors as it provides the highest accuracy and F-1 score.

6.3.2 Evaluation on the second dataset

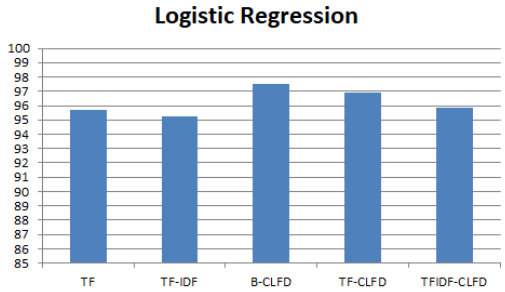
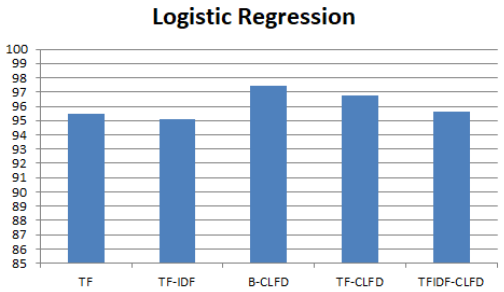
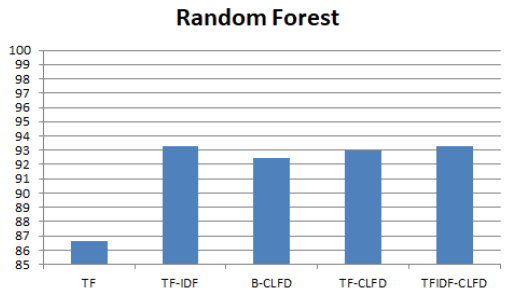
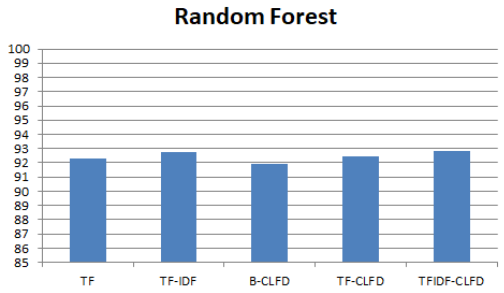
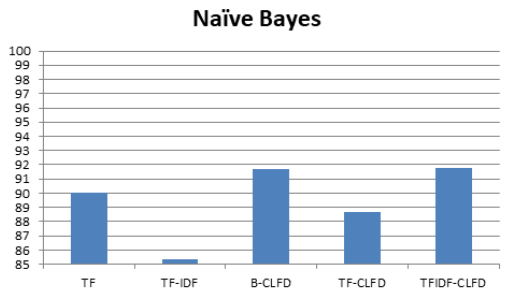
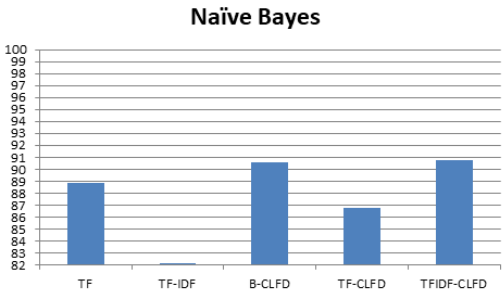
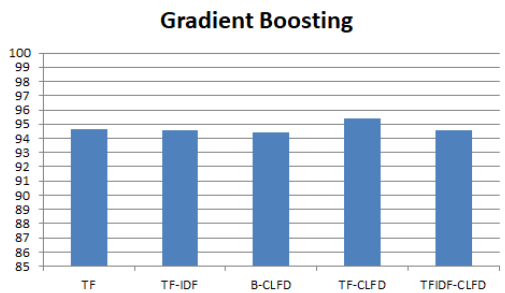
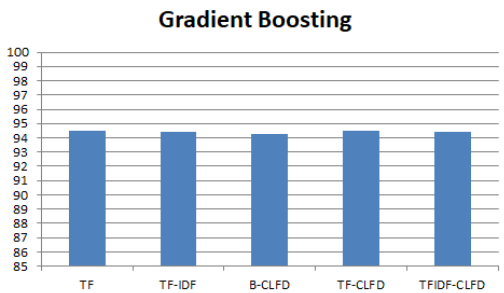
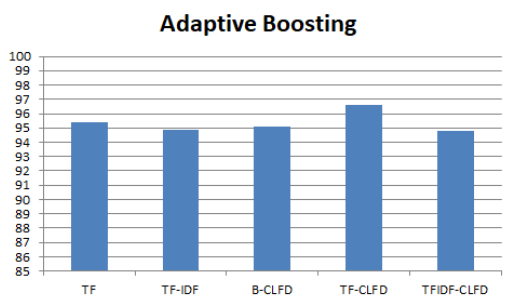
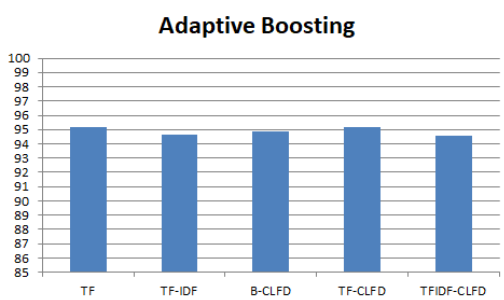
Accuracy	F-1 score
	
	
	
	
	

Table 6.6: Dataset 2: Accuracy and F-1 score

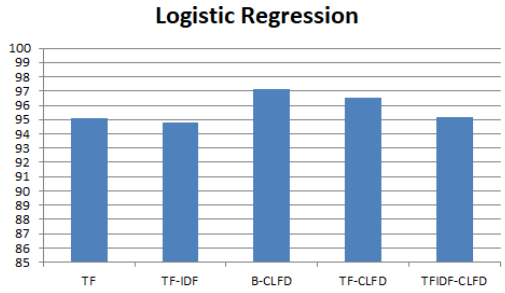
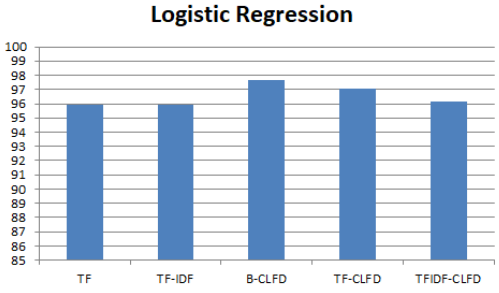
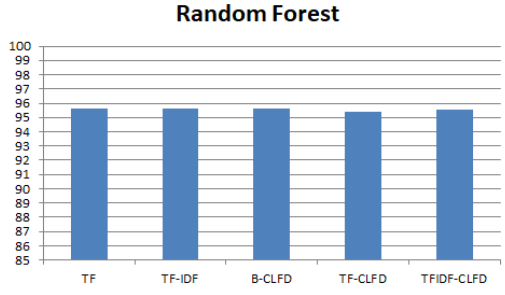
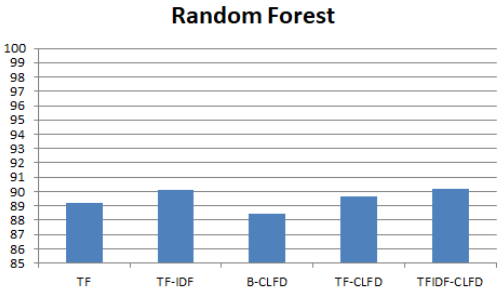
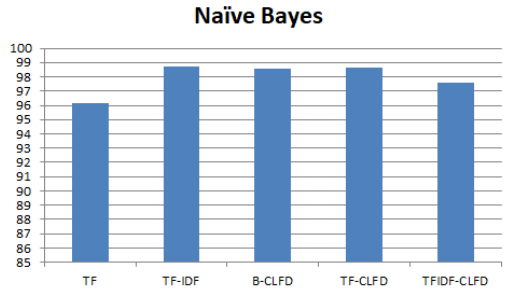
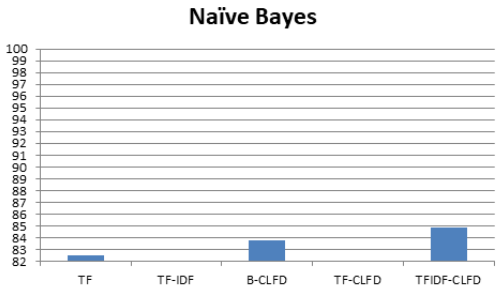
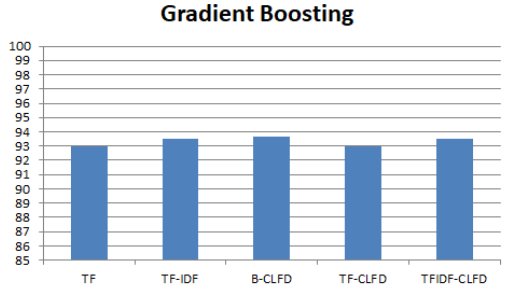
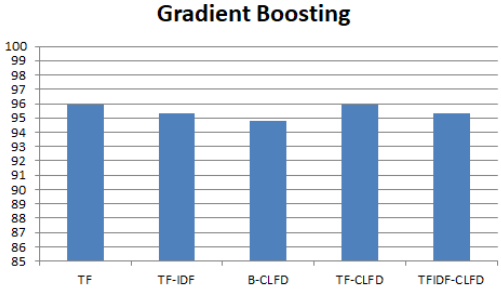
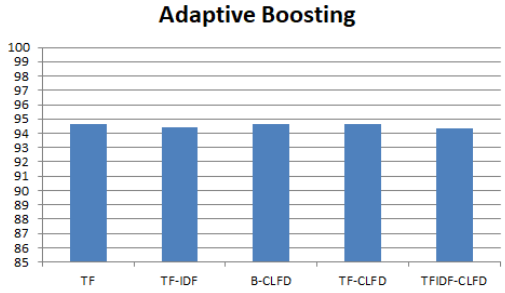
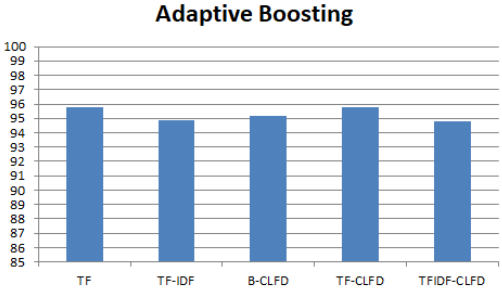
Precision	Recall
	
	
	
	
	

Table 6.7: Dataset 2: Precision and Recall

Accuracy

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	95.69	86.66	90.06	94.61
tf-idf	95.27	93.29	85.11	**94.59**
b-clfd	**97.52**	92.45	91.66	94.44
tf-clfd	96.92	92.99	88.68	95.37
tfidf-clfd	95.82	**93.31**	**91.77**	**94.59**

Precision

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	95.08	95.6	96.17	93
tf-idf	94.77	95.65	**98.72**	93.49
b-clfd	**97.17**	**95.66**	98.58	**93.69**
tf-clfd	96.51	95.42	98.61	93
tfidf-clfd	95.16	95.58	97.57	93.49

Recall

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	95.96	89.21	82.51	**95.96**
tf-idf	95.93	90.08	69.81	95.35
b-clfd	**97.65**	88.42	83.79	94.77
tf-clfd	97.06	89.66	77.44	**95.96**
tfidf-clfd	96.17	**90.21**	**84.91**	95.35

F-1 score

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	95.51	92.29	88.82	**94.46**
tf-idf	95.08	92.78	81.78	94.41
b-clfd	**97.41**	91.9	90.58	94.23
tf-clfd	96.79	92.45	86.75	**94.46**
tfidf-clfd	95.66	**92.81**	**90.8**	94.41

Table 6.8: Dataset 2: Performance of vectorization methods

In the experiments conducted on the second dataset, as presented in Tables 6.6, 6.7 and 6.8, we can make the following observations:

- The ensemble machine learning methods, *Random Forest*, *Gradient Boosting* and *Adaptive Boosting*, do not show any significant difference in the performance with different vectorization techniques. The only exception is the accuracy score of the *Adaptive Boosting* method which is 1.23% higher with *tf-clfd* feature vectors.

- The utilization of *clfd*-based feature vectors provides better results for the probabilistic machine learning methods, *Logistic Regression* and *Naive Bayes*, compared to the results provided by traditional vectorization techniques for those methods. More specifically, comparing *clfd*-based vectorization to the next best performing traditional vectorization technique, there is an approximately 2% increase in performance across all metrics.

- There is a significant boost in the performance of the *Naive Bayes* method when we compare the results of *tf-idf* with those of its *clfd* variant, *tfidf-clfd*. More specifically, there is a 6.5% to 15% increase in performance if *tfidf-clfd* feature vectors are used instead of *tf-idf* ones.

- The only exception is the precision score of the *Naive Bayes* method which is approximately the same with *clfd*-based and *tf-idf* feature vectors.

- The best performing method is *Logistic Regression* with *b-clfd* feature vectors as it provides the highest accuracy, recall and F-1 score.

6.3.3 Evaluation on the third dataset

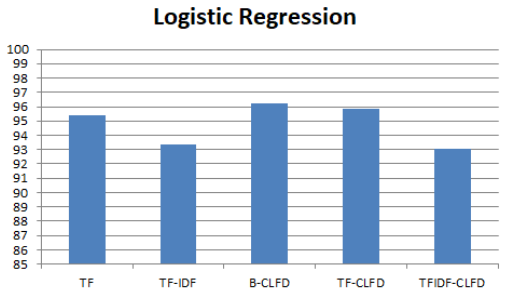
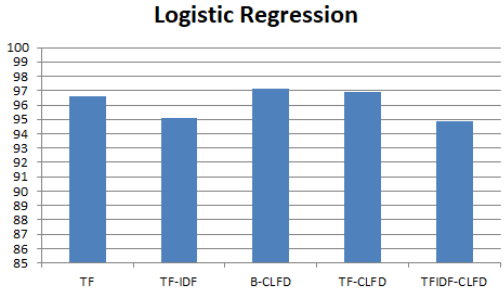
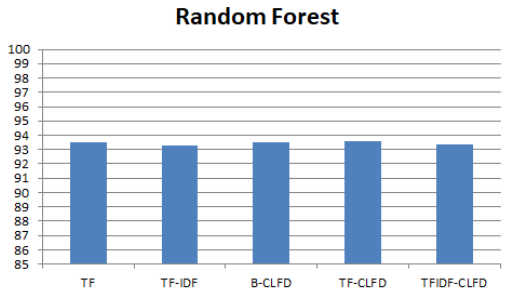
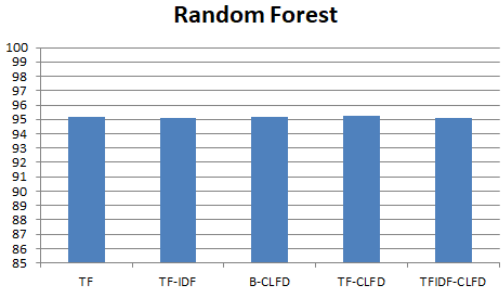
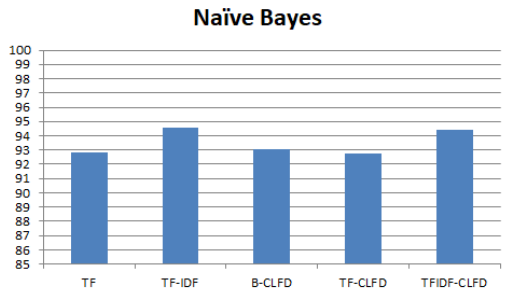
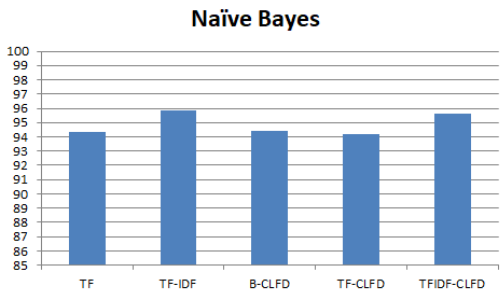
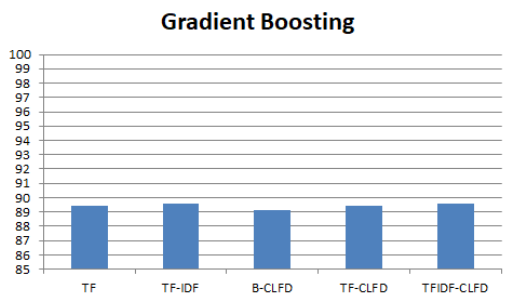
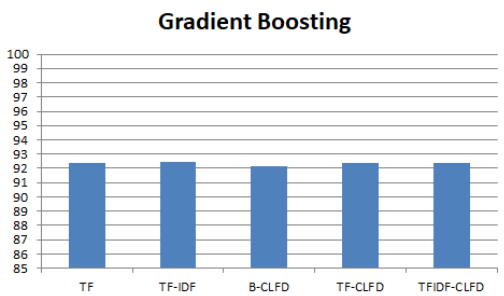
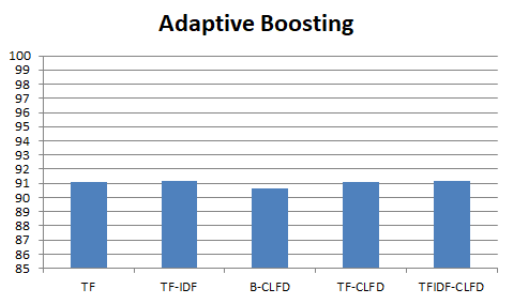
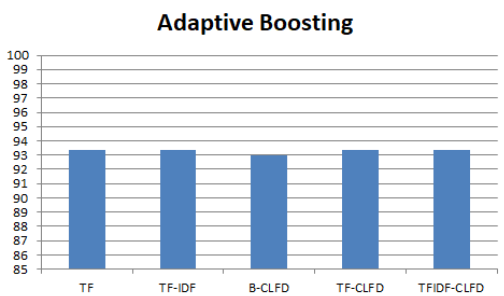
Accuracy	F-1 score
	
	
	
	
	

Table 6.9: Dataset 3: Accuracy and F-1 score

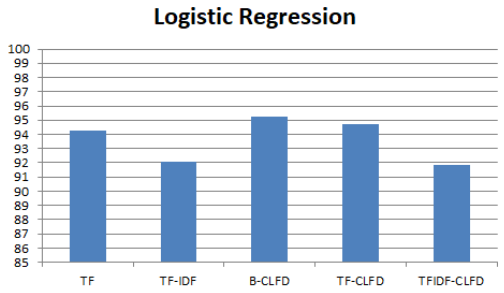
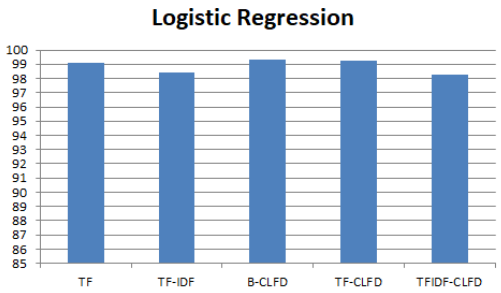
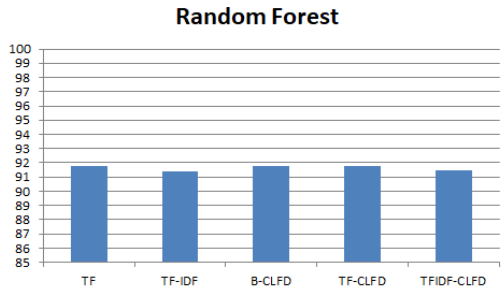
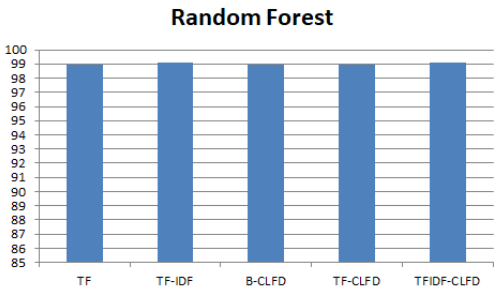
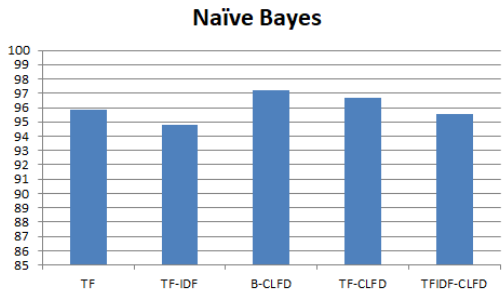
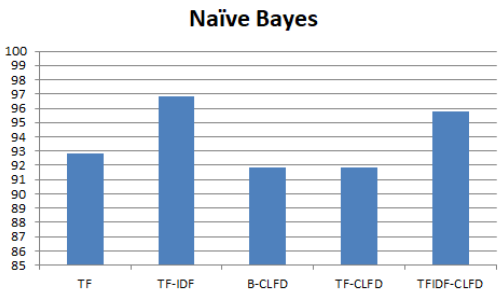
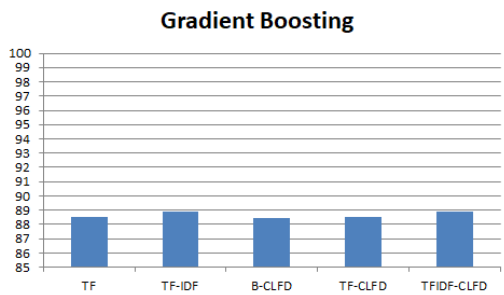
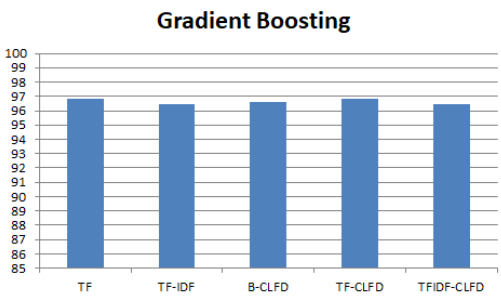
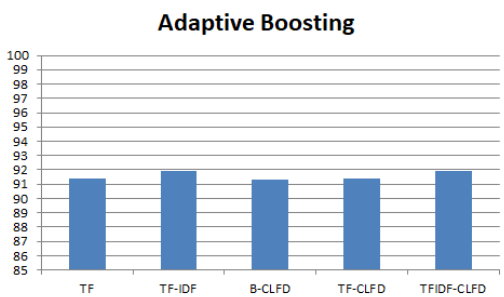
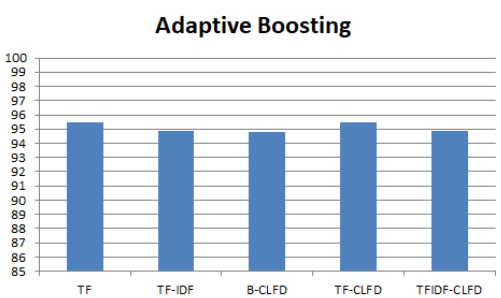
Precision	Recall
	
	
	
	
	

Table 6.10: Dataset 3: Precision and Recall

Accuracy

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	95.43	93.51	92.83	89.45
tf-idf	93.34	93.31	**94.55**	**89.6**
b-clfd	**96.21**	93.53	93.05	89.13
tf-clfd	95.84	**93.55**	92.72	89.45
tfidf-clfd	93.03	93.38	94.39	**89.6**

Precision

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	94.3	91.75	95.88	88.5
tf-idf	92.09	91.43	94.83	**88.92**
b-clfd	**95.22**	91.78	**97.22**	88.41
tf-clfd	94.73	**91.79**	96.66	88.5
tfidf-clfd	91.85	91.48	95.54	**88.92**

Recall

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	99.09	98.95	92.86	**96.87**
tf-idf	98.39	99.08	**96.85**	96.48
b-clfd	**99.29**	98.95	91.82	96.64
tf-clfd	99.26	98.98	91.87	**96.87**
tfidf-clfd	98.29	**99.09**	95.78	96.48

F-1 score

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	96.59	95.18	94.34	92.36
tf-idf	95.07	95.06	**95.82**	**92.42**
b-clfd	**97.17**	95.2	94.44	92.17
tf-clfd	96.89	**95.21**	94.2	92.35
tfidf-clfd	94.87	95.1	95.65	**92.42**

Table 6.11: Dataset 3: Performance of vectorization methods

In the experiments conducted on the third dataset, as presented in Tables 6.9, 6.10 and 6.11, we can make the following observations:

- The ensemble machine learning methods, *Random Forest*, *Gradient Boosting* and *Adaptive Boosting*, do not show any significant difference in the performance with different vectorization techniques.
- The *Naive Bayes* method does not show any significant difference in the performance with different vectorization techniques either. However, it provides a higher precision score with *b-clfd* feature vectors which is the highest among traditional machine learning methods.
- The utilization of *clfd*-based feature vectors provides better results for the *Logistic Regression* method compared to the results provided by traditional vectorization techniques for that method. More specifically, comparing *clfd*-based vectorization to the next best performing traditional vectorization technique, there is an approximately 1% increase in performance across all metrics.
- The best performing method is *Logistic Regression* with *b-clfd* feature vectors as it provides the highest accuracy, recall and F-1 score.

6.3.4 Discussion of results

Regarding the comparison of vectorization techniques for traditional machine learning methods, we observe certain patterns in the results. We notice that the ensemble machine learning methods, *Random Forest*, *Gradient Boosting* and *Adaptive Boosting*, do not show any significant difference in their performance with varying vectorization approaches. However, the probabilistic machine learning methods, *Logistic Regression* and *Naive Bayes*, achieve a higher performance by using *clfd*-based feature vectors. In fact, our *b-clfd* vectorization approach consistently increases the performance of *Logistic Regression* in Dataset 1, Dataset 2 and Dataset 3 by at least 1.88%, 1.9% and 0.58% respectively. The only exception is the performance of *Naive Bayes* in Dataset 3 which is slightly higher (0.17%) with *tf-idf* feature vectors. Therefore, there is a consistent high ranking for machine learning methods that utilize our *clfd* vectorization approach, as it allows them to achieve comparable or higher results in all datasets. The same pattern of results also appears in other metrics such as accuracy, precision and recall. The best performing traditional machine learning method is *Logistic Regression* with *b-clfd* feature vectors which consistently provides the highest results, outperforming other traditional machine learning methods in F-1 score in Datasets 1, 2 and 3 by at least 2.87%, 2.95% and 1.35% respectively.

	With preprocessing	Without preprocessing
Tf	92.73	91.73
Tfidf	92.13	91.75
B-clfd	**94.61**	**95.71**

Table 6.12: Logistic Regression - Dataset 1

Finally, we report that the performance of *clfd*-based methods is not affected by the quality or absence of data preprocessing. As presented in the example of Table 6.12, without preprocessing, *Logistic Regression (LR)* with *b-clfd* provides results that are actually better than the ones achieved after preprocessing (Table 6.5). On the other hand, the performance of *LR with tf* and *LR with tf-idf* is reduced in case of no data preprocessing.

6.4 Comparison of machine learning methods

In this section, we present the classification results of deep learning methods and make a comparison with the best performing traditional machine learning method, *Logistic Regression* with *b-clfd* feature vectors. Furthermore, we present the results of the *hybrid model* and discuss its advantages. Finally, apart from a performance comparison, we also compare the best performing methods in terms of classification time.

6.4.1 Performance comparison

Dataset 1				
	Accuracy	Precision	Recall	F-1 score
Bali 2019	87.3	89	87	89
lstm	90.9	92.4	88.96	90.65
Bi-lstm	92.09	92.25	91.76	92
cnn+lstm	92.22	90.33	94.41	92.33
Mult.lstm	92.62	90.51	**95.08**	92.74
LR + clfd	**94.53**	**94.6**	94.64	**94.61**
Dataset 2				
	Accuracy	Precision	Recall	F-1 score
Bali 2019	91.05	93	94	94
lstm	94.54	96.43	91.74	94.02
Bi-lstm	95	95.45	93.8	94.62
cnn+lstm	96.55	97.11	95.48	96.29
Mult.lstm	94.68	95.86	92.64	94.22
LR + clfd	**97.52**	**97.17**	**97.65**	**97.41**
Dataset 3				
	Accuracy	Precision	Recall	F-1 score
lstm	96.24	96.61	97.63	97.12
Bi-lstm	95.64	96.69	96.36	96.63
cnn+lstm	**96.78**	**97.26**	97.78	**97.52**
Mult.lstm	95.96	97.2	96.54	96.87
LR + clfd	96.21	95.22	**99.29**	97.17

Table 6.13: Comparison of machine learning methods

From the comparison of the machine learning methods, in the experimental evaluation conducted on the three datasets, we observe certain patterns in the results. Regarding deep learning methods, we notice that *cnn+lstm* outperforms the other deep learning methods for Datasets 2 and 3, while its performance is comparable to that of *mult.lstm* for Dataset 1. In Dataset 1, *lstm* and *Bi-lstm* have the highest precision score among deep learning methods, but are otherwise performing at least slightly worse than *cnn+lstm* in all other cases (regardless of evaluation metric or dataset). Therefore, we consider *cnn+lstm* as the best performing deep learning method.

However, we notice that deep learning methods are outperformed in many occasions. In Dataset 1, *Logistic Regression* with *b-clfd* vectorization consistently achieves the highest performance. It outperforms all deep learning methods in accuracy, precision and F-1 score by at least 1.91%, 2.2% and 1.87% respectively. The *mult.lstm* achieves a slightly (0.44%) higher recall score. In Dataset 2, *Logistic Regression* with *b-clfd* consistently achieves the highest results across all metrics. It outperforms deep learning methods in accuracy, precision, recall and F-1 scores by at least 0.97%, 0.06%, 2.17% and 1.12% respectively. In Dataset 3, the results of deep learning methods and those of *Logistic Regression* with *b-clfd* are comparable. The best performing deep learning method, *cnn+lstm*, provides higher accuracy, precision and F-1 scores by 0.57%, 2.04% and 0.35% respectively. However, *Logistic Regression* with *b-clfd* feature vectors outperforms all other deep learning methods in those metrics. Furthermore, it achieves the highest overall recall score (at least 1.51% higher than others). Therefore, *Logistic Regression with b-clfd vectorization* achieves clearly higher results than deep learning methods in Datasets 1 and 2, while the results are comparable for Dataset 3.

Furthermore, for Datasets 1 and 2, our *clfd* vectorization approach allows *Logistic Regression* to outperform existing state-of-the-art work [18], which also utilizes traditional machine learning methods, by a significant margin. More specifically, for Dataset 1, it outperforms state-of-the-art results by 7.23% in accuracy, 5.6% in precision, 7.64% in recall and 5.61% in F-1 score. For Dataset 2, it outperforms state-of-the-art results by 6.47% in accuracy, 4.17% in precision, 3.65% in recall and 3.41% in F-1 score.

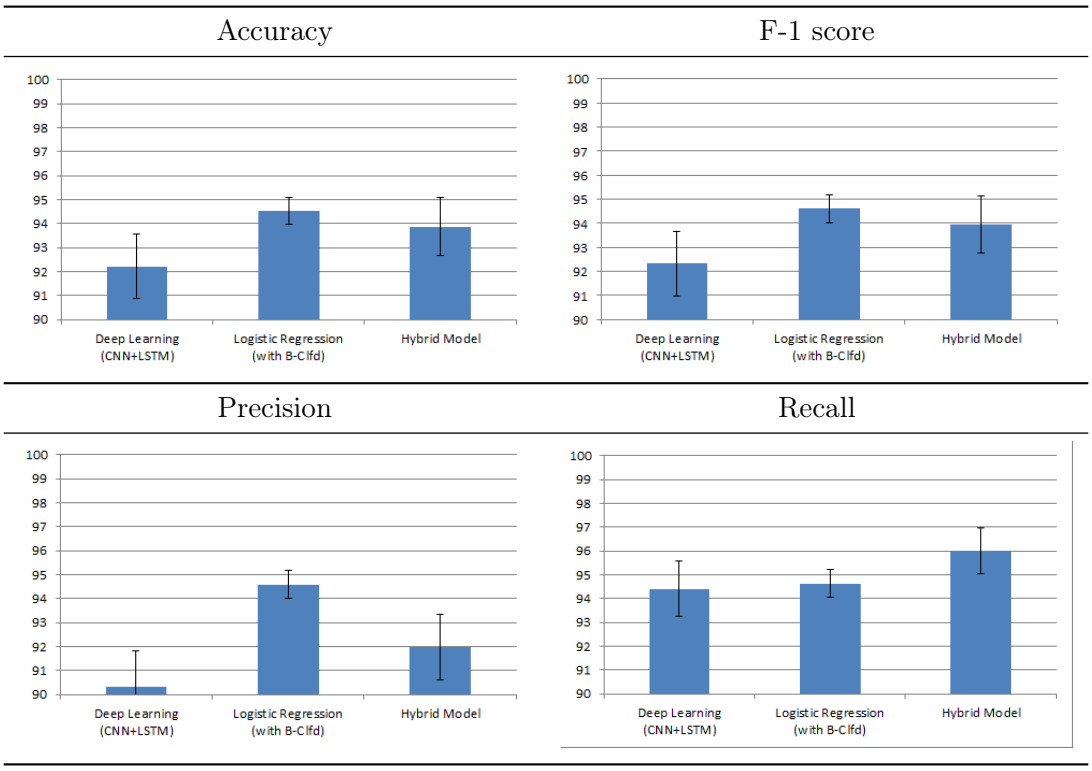


Table 6.14: Dataset 1: Performance of machine learning methods (95% confidence)

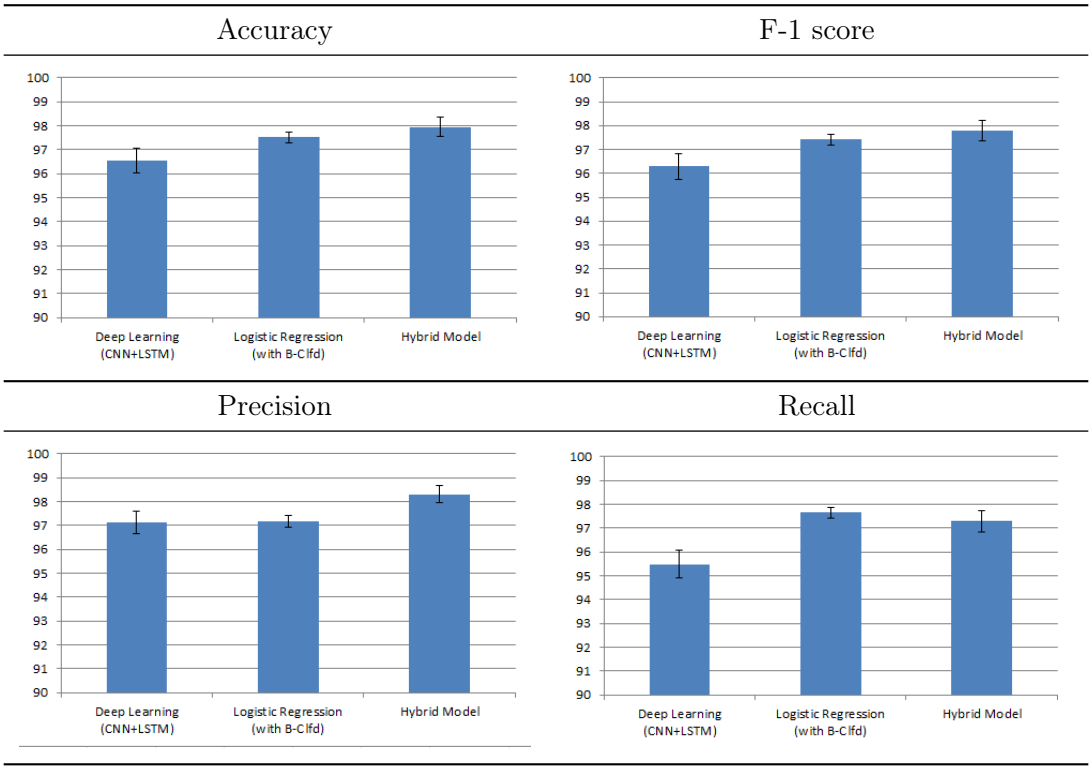


Table 6.15: Dataset 2: Performance of machine learning methods (95% confidence)

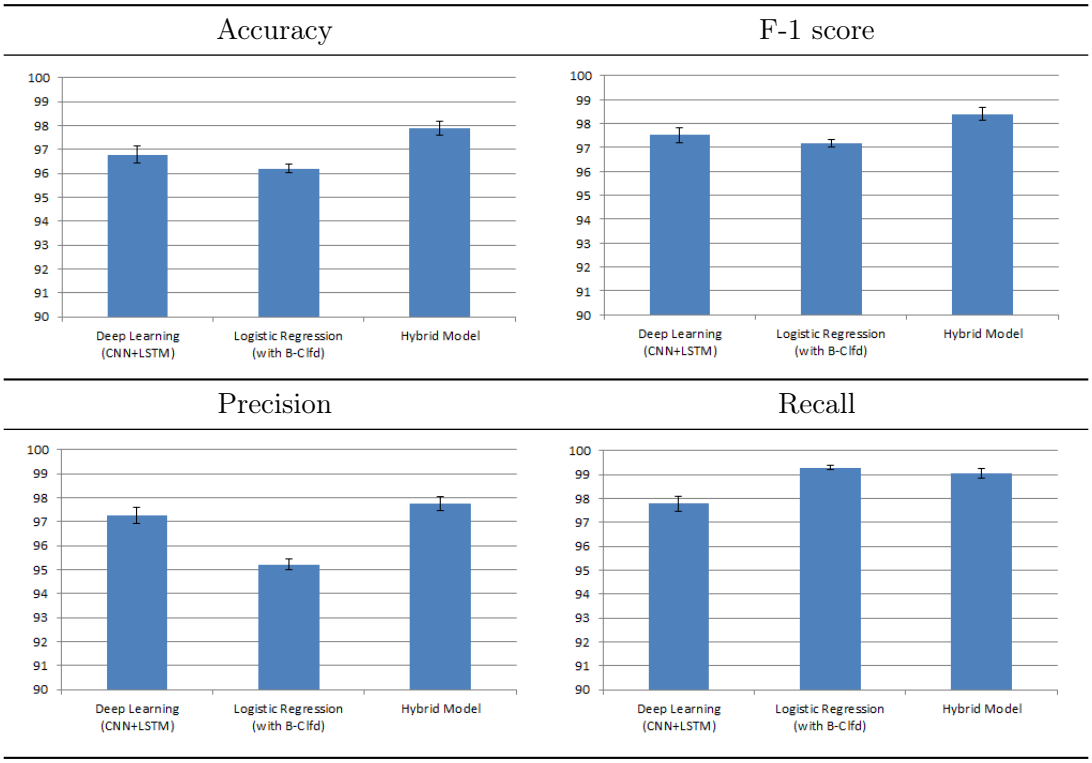


Table 6.16: Dataset 3: Performance of machine learning methods (95% confidence)

In Table 6.14, Table 6.15 and Table 6.16 we present the classification results of the best performing traditional machine learning method (*Logistic Regression* with *b-clfd* feature vectors), the best performing deep learning method (*cnn+lstm*) and the *hybrid* method. For the *statistical validity* of our results, we present their mean performance for each of the three datasets, along with error bars which correspond to 95% confidence intervals. These confidence intervals were calculated as follows:

$$[\hat{p} - \varepsilon, \hat{p} + \varepsilon], \text{ where } \varepsilon = z_{1-\frac{a}{2}} \cdot \sqrt{\frac{\hat{p} \cdot (1 - \hat{p})}{n}} \tag{6.1}$$

In the above equation, $\hat{p}$ is the probability that represents the mean performance and ε is the confidence interval. Furthermore, n is the number of testing samples and z, or z-score, is a numerical measurement used in statistics of a value's relationship to the mean (average) of a group of values, measured in terms of standard deviations from the mean [51]. Since we want to calculate an interval which corresponds to 95% confidence:

$$1 - a = 0.95 \iff a = 0.05 \iff 1 - \frac{a}{2} = 0.975 \tag{6.2}$$

From the z-score lookup table [52], we know that $z_{0.975} = 1.96$. Therefore, we conclude to the following equation for the calculation of the 95% confidence intervals:

$$[\hat{p} - \varepsilon, \hat{p} + \varepsilon], \text{ where } \varepsilon = 1.96 \cdot \sqrt{\frac{\hat{p} \cdot (1 - \hat{p})}{n}} \tag{6.3}$$

Therefore, in Tables 6.14, 6.15 and 6.16, we can make certain statistically valid observations. We can see that *Logistic Regression* with *b-clfd* feature vectors has the lowest variance. In Datasets 1 and 2, *Logistic Regression* with *b-clfd* feature vectors provides results that are *significantly higher* than those of the best deep learning method, while they are comparable to those of *hybrid*. In Dataset 3, *Logistic Regression* with *b-clfd* feature vectors and deep learning methods provide results which are comparable to each other. However, in Dataset 3, *hybrid* provides results that are clearly superior to those of other methods, across all metrics. Specifically, we report that *hybrid* consistently outperforms deep learning methods in accuracy, precision, recall and F-1 score by at least 1.13%, 0.49%, 1.27% and 0.88% respectively.

6.4.2 Time comparison

	Dataset 1	Dataset 2	Dataset 3
LR + clfd	6.91	28.53	54.67
cnn+lstm	110.17	273.63	1295.32
Hybrid	112.22	292.88	1330.92

Table 6.17: Time comparison

In Table 6.17, we make a comparison of classification times among the best performing machine learning methods in terms of classification time required after data preprocessing. Note that this does not include the time required for the training of deep learning methods. We observe that *Logistic Regression* with *b-clfd* feature vectors is approximately 16, 10 and 24 times more time efficient than the other methods for Datasets 1, 2 and 3 respectively. The other two methods are comparable, with *hybrid* requiring approximately 5% more time than *cnn+lstm*. Time scalability is also shown in Figure 4, where the significant classification time advantage of *Logistic Regression* with *b-clfd* is clearly demonstrated.

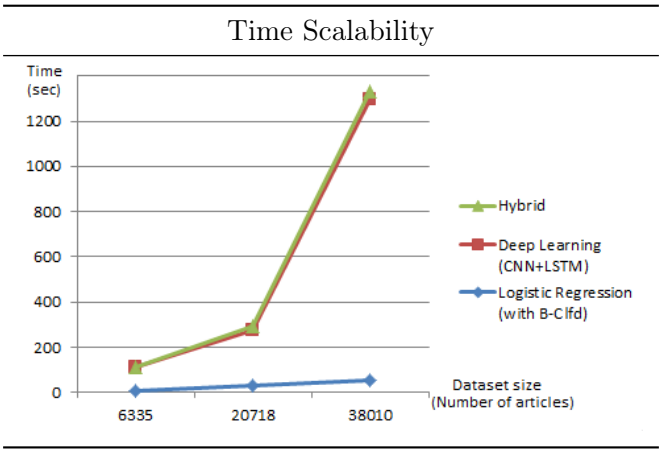


Table 6.18: Time scalability

Therefore, we conclude that *Logistic Regression* with *b-clfd* feature vectors appears to have a linear time scaling and a training and classification time of under a minute even for large datasets. On the other hand, we observe that there is no significant difference between the classification time of the *hybrid model* and that of the *cnn+lstm* neural network for different dataset sizes. This means that using *Logistic Regression* with *b-clfd* feature vectors as a component in the creation of the *hybrid model* can increase the performance of a deep learning method without significantly affecting its time efficiency. Both the *hybrid model* and the *cnn+lstm* neural network show an exponential scaling of classification time with different dataset sizes, which makes *Logistic Regression* with *b-clfd* feature vectors gain an important advantage regarding time efficiency.

6.5 Discussion of classification results

The systematic comparison of different vectorization techniques as well as machine learning methods for the binary text classification task of creating an effective fake news detection system leads to the following conclusions:

- We demonstrated that our novel vectorization approach, *clfd*, is a simple and effective way for boosting the performance of certain machine learning methods. The experimental results show that a *clfd*-based vectorization approach consistently provides comparable or better results than other vectorization techniques, such as *tf* or *tf-idf*, for traditional machine learning methods.

- In addition, the experimental results also show that the *Logistic Regression* method provides significantly higher results than those reported in recent published work [18] for Dataset 1 and Dataset 2.

- In fact, it can also outperform deep learning methods if *b-clfd* feature vectors are used. More specifically, it provides consistently higher results than all deep learning methods for small and medium sized datasets (Datasets 1 and 2), while there is no significant difference in performance for large datasets (Dataset 3).

- Moreover, for large datasets (Dataset 3), the *hybrid* method successfully combines a *cnn+lstm* neural network with *Logistic Regression* which utilizes *b-clfd* feature vectors, to achieve the highest performance and significantly outperform deep learning methods across all metrics.

Chapter 7

Conclusions & Future Work

7.1 Conclusion

In this thesis, we systematically compared several vectorization techniques as well as machine learning methods for the binary text classification task of creating an effective content-based fake news detection system.

We proposed a novel text vectorization technique, *clfd*; discussed its advantages with respect to "classic" vectorization approaches; and showed that its employment by machine learning methods can lead to the development of a content-based fake news detection system that achieves high performance and requires a minimal amount of classification time. In particular, we showed that *clfd* can turn *Logistic Regression* into a method that is a winner for small and medium-sized datasets; a method the classification performance of which is comparable to that of deep neural networks in large datasets, while it retains the advantage of minimal classification time. Moreover, when used as a component of our novel *hybrid* method, its performance clearly surpasses that of "pure" deep learning methods across all metrics, even for large datasets.

The results of this work may have certain implications in the real world. Regarding academia, *clfd* is a statistical approach for vectorization which leads to more effective supervised text classification. Furthermore, the *hybrid* method leads to higher results than deep neural networks across all metrics and dataset sizes. These contributions can be of particular interest within the natural language processing and machine learning research communities. Regarding industry, a *clfd*-boosted Logistic Regression classifier achieves a very high performance while the cost regarding training and classification time is minimal. This can be of particular interest to online news agencies, as it can be used for the quick and effective classification of news articles in an online fashion. We encourage the utilization of our work in the aforementioned ways as well as its further improvement.

7.2 Future Work

We encourage the utilization of our work in the aforementioned ways as well as its further improvement. As a natural next step, we intend to evaluate our approach in other application domains of interest. Future work also includes improving the *hybrid* method further, via equipping it with a more sophisticated weighting scheme. Moreover, additional machine learning methods can be added to its existing pool of methods, to increase its diversity and potentially its power. More advanced deep learning architectures, such as hierarchical attention networks [53], can also be considered. Finally, more research has to be done on how to increase the effectiveness and time efficiency of the *title similarity* component of our *hybrid* method, by making use of heuristics and more advanced information retrieval techniques.

Bibliography

[1] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–236, 2017.

[2] Julia Hunt. 'Fake news' named Collins Dictionary's official Word of the Year for 2017. Available at: https://www.independent.co.uk/news/uk/home-news/fake-news-word-of-the-year-2017-collins-dictionary-donald-trump-kellyanne-conway-antifa-corbynmania-a8032751.html, 2017. [Online; accessed 26-February-2019].

[3] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.

[4] Sara Rimer-Boston. Is fact-checking 'fake news' a waste of time? Available at: https://www.futurity.org/fact-checking-fake-news-1475152/, 2017. [Online; accessed 26-February-2019].

[5] Eurobarometer. Final results of the Eurobarometer on fake news and online disinformation. Available at: https://ec.europa.eu/digital-single-market/en/news/final-results-eurobarometer-fake-news-and-online-disinformation, 2018. [Online; accessed 25-April-2019].

[6] Ikagai Law. The "fake news" problem: What are governments doing about it? Available at: https://www.ikigailaw.com/the_fake_news_problem, 2018. [Online; accessed 26-February-2019].

[7] The Moscow Times. Putin signs fake-news internet insults bills into law. Available at: https://www.themoscowtimes.com/2019/03/18/putin-signs-fake-news-internet-insults-bills-into-law-a64850, 2019. [Online; accessed 25-April-2019].

[8] University of Michigan. "Fake News," Lies and Propaganda: How to Sort Fact from Fiction. Available at: https://guides.lib.umich.edu/fakenews, 2018. [Online; accessed 26-February-2019].

[9] Merriam-Webster Dictionary. Disinformation: Noun. Available at: https://www.merriam-webster.com/dictionary/disinformation, 2019. [Online; accessed 26-February-2019].

[10] Claire Wardle. Fake news. It's complicated. Available at: https://firstdraftnews.org/fake-news-complicated/, 2017. [Online; accessed 26-February-2019].

[11] Niall J Conroy, Victoria L Rubin, and Yimin Chen. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4, 2015.

[12] Natali Ruchansky, Sungyong Seo, and Yan Liu. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806. ACM, 2017.

[13] Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 252–256, 2017.

[14] Kai Shu, Suhang Wang, and Huan Liu. Understanding user profiles on social media for fake news detection. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 430–435. IEEE, 2018.

[15] Samir Bajaj. The pope has a new baby! *Fake News Detection Using Deep Learning*, 2017.

[16] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[17] Arjun Roy, Kingshuk Basak, Asif Ekbal, and Pushpak Bhattacharyya. A deep ensemble framework for fake news detection and classification. *arXiv preprint arXiv:1811.04670*, 2018.

[18] Arvinder Pal Singh Bali, Mexson Fernandes, Sourabh Choubey, and Mahima Goel. Comparative performance of machine learning algorithms for fake news detection. In *International Conference on Advances in Computing and Data Sciences*, pages 420–430. Springer, 2019.

[19] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[20] Fake News Challenge. Fake News Challenge Stage 1 (FNC-1): Stance Detection. Available at: http://www.fakenewschallenge.org, 2017. [Online; accessed 26-February-2019].

[21] Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*, 2017.

[22] Gisel Bastidas Guacho, Sara Abdali, Neil Shah, and Evangelos E Papalexakis. Semi-supervised content-based detection of misinformation via tensor embeddings. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 322–325. IEEE, 2018.

[23] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937, 2017.

[24] Sameer Badaskar, Sachin Agarwal, and Shilpa Arora. Identifying real or fake articles: Towards better language modeling. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[25] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*, 2017.

[26] Hamid Karimi and Jiliang Tang. Learning hierarchical discourse-level structure for fake news detection. *arXiv preprint arXiv:1903.07389*, 2019.

[27] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[28] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach.* " O'Reilly Media, Inc.", 2017.

[29] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[30] GluonNLP. Pre-trained Word Embeddings. Available at: https://gluon-nlp.mxnet.io/examples/word_embedding/word_embedding.html, 2019. [Online; accessed 26-February-2019].

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[32] 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.

[33] Deep Learning Course Wiki. Logistic Regression. Available at: http://wiki.fast.ai/index.php/Logistic_Regression, 2019. [Online; accessed 06-June-2019].

[34] Juhi. Simple guide for ensemble learning methods. Available at: https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2, 2019. [Online; accessed 06-June-2019].

[35] Pentaho. Increasing the Performance of ML Algorithms using Boosting Techniques. Available at: https://blog.tenthplanet.in/boosting-techniques, 2019. [Online; accessed 06-June-2019].

[36] Charu C Aggarwal. Neural networks and deep learning. *Cham: Springer International Publishing*, 2018.

[37] Lazy Programmer Inc. (Udemy). Deep Learning: Advanced NLP and RNNs. Available at: https://gluon-nlp.mxnet.io/examples/word_embedding/word_embedding.html, 2019. [Online; accessed 06-June-2019].

[38] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[39] New York Times. Breaking News, World News & Multimedia. Available at: https://www.nytimes.com, 2019. [Online; accessed 30-March-2019].

[40] The Guardian. News, sport and opinion. Available at: https://www.theguardian.com/international, 2019. [Online; accessed 30-March-2019].

[41] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966, 2015.

[42] Google. Word2Vec: Tool for computing continuous distributed representations of words. Available at: https://code.google.com/archive/p/word2vec/, 2013. [Online; accessed 30-March-2019].

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[44] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[45] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[46] François Chollet. others. 2015. *Keras: Deep learning library for theano and tensorflow. URL: https://keras. io/k*, 2015.

[47] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[48] George McIntire. Fake news dataset. Available at: https://github.com/docketrun/Detecting-Fake-News-with-Scikit-Learn/tree/master/data, 2017. [Online; accessed 17-November-2018].

[49] Kaggle. Fake News. Available at: https://www.kaggle.com/c/fake-news/data, 2017. [Online; accessed 20-January-2019].

[50] Megan Risdal. Getting Real about Fake News. Available at: https://www.kaggle.com/mrisdal/fake-news, 2016. [Online; accessed 20-January-2019].

[51] Investopedia. Z-Score Definition. Available at: https://www.investopedia.com/terms/z/zscore.asp, 2019. [Online; accessed 15-August-2019].

[52] Data Science Central. Z-table (Right of Curve or Left). Available at: https://www.statisticshowto.datasciencecentral.com/tables/z-table, 2019. [Online; accessed 15-August-2019].

[53] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.