Disinformation Detection with Model Explanations

Konstantinidis Ioannis

SID: 3308170009

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2018

THESSALONIKI – GREECE

Disinformation Detection with Model Explanations

Konstantinidis Ioannis

SID: 3308170009

Supervisor:	Assist. Prof. Vassilios Peristeras
Supervising Committee Members:	Assist. Prof. Vassilios Peristeras
	Dr. Christos Berberidis
	Assist. Prof. Christos Tjortjis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2018

THESSALONIKI – GREECE

Abstract

This dissertation was written as a part of the MSc in Data Science at the International Hellenic University.

Disinformation on the web has become an important problem to our society, and generally refers to inaccurate information that is intended to harm the public. It includes fake news, imposter and fabricated content, hoaxes and other types of false information. Currently, most approaches to identify such content are based on fact-checking agencies manually searching for evidence that supports or contradicts the news statement. These approaches have natural limitations because they require a great amount of manual labour. The need for automatic disinformation detection tools, that can quickly and massively detect disinformation by its source has therefore been widely acknowledged. Recent research studies deal with this problem by utilizing a variety of Machine Learning and Natural Language Processing techniques that in certain domains appear to achieve high prediction accuracy. However, none of them involves explanations for their classification; for a model to be trustworthy, it has to provide the user with information on *why* it classified a content item as true or false. Machine learning model interpretability is an open research challenge that aims to enhance model transparency and reduce bias and prejudice. Opening the "black box" of algorithmic tools also allows for a deeper understanding of the inherent characteristics of a domain, which can eventually lead to better approaches for the mitigation of its negative aspects.

This thesis investigates the problem of disinformation detection by implementing various fake news classification models using only their textual content and subsequently evaluates state-of-the-art algorithms for explaining classifier decisions.

Acknowledgements

This thesis has received funding from the European Union's Horizon 2020 research and innovation program under the Grant Agreement No. 770302[1].

I would like to thank my thesis advisors Professor Vassilios Peristeras and Dr Christos Berberidis for their valuable and constructive feedback during the planning and development of this research work. Their willingness to afford their time so generously, has been very much appreciated.

Finally, I must express my very profound gratitude to my parents and my brother Konstantinos as well as Elena for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Konstantinidis Ioannis

07/12/2018

[1] The Co-Inform project is co-financed by Horizon 2020 – the Framework Programme for Research and Innovation (2014-2020), H2020-SC6-CO-CREATION-2016-2017 (CO-CREATION FOR GROWTH AND INCLUSION), with a contribution of EUR4,110,760

Contents

1 Introduction

With the emergence of social media on the Web, everybody can instantly share information with others. Recent studies have shown that most people use social media to get informed as it is a free and quick way to read the news [1]. Furthermore, it is considered a great asset for keeping the public updated to the current events and on top of that, everyone can choose between numerous news sources with several topics that satisfy their needs. Therefore, social media is a very powerful network, consisting of a huge amount of data and information.

However, this framework also allows the dissemination of false information. It can be caused by poor journalism or naive claims made by individuals, who wrongfully believe certain facts and want to share their thoughts. This form of false information is defined as misinformation. Another type of false information is disinformation, which is information that is intended to harm the users. Consequently, disinformation is considered the most dangerous form of false information denoting the importance of detecting such content and quickly hampering its spread. A type of disinformation is fake news, which refers to articles that are intentionally and verifiably false and could mislead people. A characteristic example of disinformation dissemination has occurred during the 2016 US election cycle, where many fake news stories where published and shared to affect public opinion by opposing them against specific political parties [2].

Nevertheless, people cannot easily identify the verifiability and authenticity of the news pieces. As it is shown in [3], after having given people the task to recognize between real and hoax articles, they managed to accurately predict only the 66% of the articles. Everyone is vulnerable to disinformation regardless of their social, financial or educational status. CoInform[2], a research project funded by the European Commission aiming at increasing resilience to misinformation, states that false information can cause the decay of people's belief over media and institutions and can mislead the political campaign in the 2019.

[2] https://coinform.eu/how-will-co-inform-tackle-misinformation-and-make-a-difference-for-our-society

The term “fake news” has become highly popular over the last 3 years, (Figure 1.1.) Thus, disinformation on the Web poses an important problem to the society. There is a need of detecting such content to provide trust to the users. Fake news is mainly identified by fact-checking agencies. They refer to group of journalists that, given a news content, manually check if an article is fake news. The best practice is to search on the Web evidence that supports or contradicts the claims of the article, which requires a significant amount of time. However, as the size of data flowing on the Web exponentially increases, it is not feasible to manually check the verifiability of all news content. Several research studies have tried to develop automatic fake news detection methods. Yet, most of the models have high complexity without providing transparency of their behavior. It is important to know how a black box model works, in order to ensure that the algorithm is not biased to certain parties. In other words, users need to be confident that the fake news detection algorithms are fair. Furthermore, it is essential to identify fake news content by its source to prevent it from disseminating using only content information.

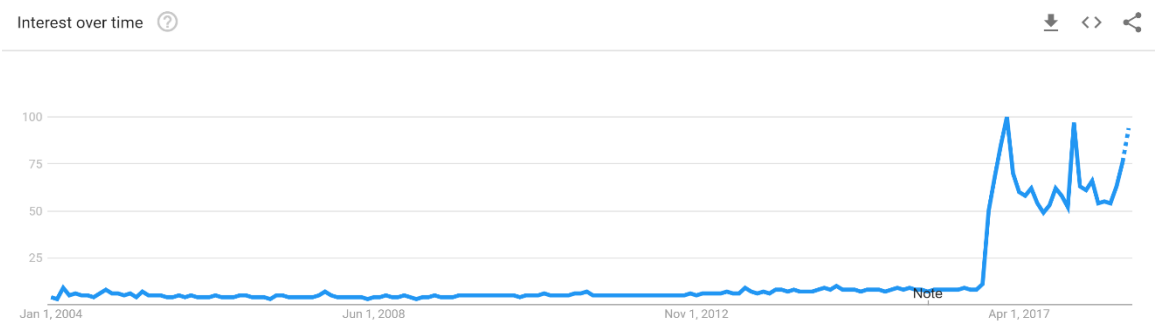

Figure 1.1. Fake news popularity on Google since 2004

In this thesis, we tackle this problem by employing several fake news detection algorithms over text data only and utilizing LIME and EDC algorithms to explain individual predictions of the models.

The rest of the thesis is organized as follows: Chapter 2 provides an overview of research studies regarding disinformation definition and impact, fake news detection models and methods for explaining the classifier’s decisions. Chapter 3 gives an analysis of the dataset used for the experiments, the implemented text classification approaches and model explanation algorithms. Chapter 4 presents the experimental results of the employed approaches. Chapter 5 provides a discussion of the findings of this thesis and

future directions to overcome some limitations. Finally, chapter 6 gives a conclusion of what has been achieved.

2 Literature review

This chapter provides a review of the research literature on fake news detection and model explanation methods for text classification.

2.1 Fake news definition

The definition of fake news originates from the question "What is real news?" and can be found in various versions in the literature [4]. According to [5], news is "an account of a recent, interesting and significant event". Moreover, in [6], news is defined as "an account of events that significantly affect people". Another formal definition was stated by [7], who claim that news is "a dramatic account of something novel or deviant". News is considered as an outcome of journalism which should give "independent, reliable, accurate, and comprehensive information" [8]. Journalism must equip people with the necessary information to be "free and self-governing" and hence all these definitions must report the truth.

Nevertheless, news is socially constructed for better comprehensibility for the readers and hence journalists need to practice subjective expression and decide on which particular information should include or not in their report. Therefore, news is not only exposed to journalists' beliefs, but also to external groups such as audiences, the government and even advertisers [9]. Furthermore, news is considered a specific product, which at first is sold in audiences and subsequently these audiences are sold to the advertisers. Consequently, the news audiences are being exposed to market goals.

Yet, news must be a source of precise and truthful information, which is a total contradiction with the term "fake news", as the latter means in general copy, counterfeit, inauthentic [10]. According to the Oxford dictionary, the word "fake" means not genuine or counterfeit or imitation. On top of that, the authors in [11] claim that there is no agreed definition of "fake news". According to [2], fake news is defined as news articles that are intentionally and verifiable false and could mislead readers". Other research papers connect fake news with satire news because the content of satire is considered false. Although, it is mostly for entertainment purpose, it reveals misleading intentions to the

audiences. Moreover, the authors in [12] define fake news as news originating from websites that falsely state to be news organizations, whereas it is about organizations that publish false pieces aiming at maximizing advertising earnings.

The authors in [13], make a general taxonomy of false information. They claim that false information is based on intent or on knowledge (Figure 2.1). The first type can also be extended to misinformation and disinformation. Misinformation is defined as the inadvertent sharing of false information while disinformation is, by definition, the spread of false information with the intent to deceive. Disinformation is considered the most dangerous type of false information, as it intends to harm the audiences and understanding its motives is considered to be highly important. They also claim that fake news is fitted as a sub-category of disinformation. A characteristic example of fake news is the spread of political disinformation during the 2016 US elections, which even resulted in public shootings [14], [15].

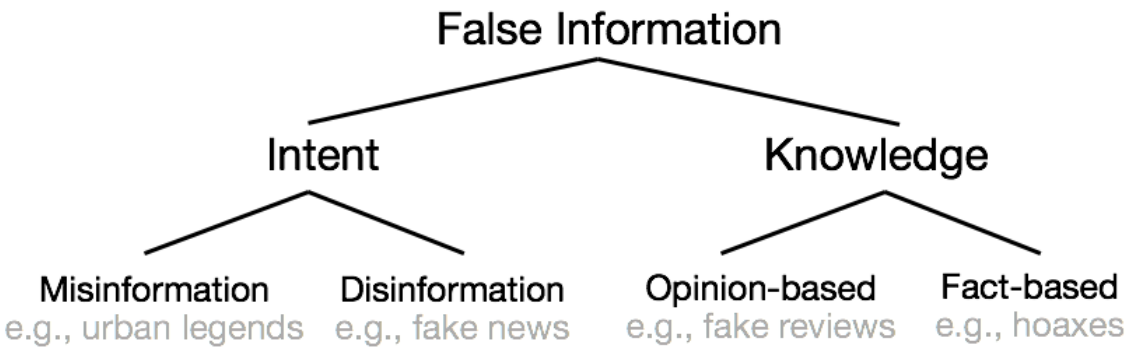

Figure 2.1. Taxonomy of false information according to [13]. Fake news is considered a sub-category of false information based on intent.

In [16], the authors claim that there is no agreed definition of the term "fake news", as it has been given multiple interpretations. They also suggest the government declines this term and utilize a more general definition of misinformation and disinformation, which can assist companies and governments for better enforcement and regulation.

2.2 Fake news impact / dissemination

False information has major influence as its intention is to harm the consumers, as described in [13]. More specifically, it has a great impact on the stock market [17], on

impeding reciprocation at the time of natural disasters [18] and terroristic activity [14]. Its effect on web and social media is measured by the degree dissemination of its content to the readers, taking into account statistical metrics, such as number of reads or shares and number of days without the fake news piece being removed.

Measuring impact of false information has been highly studied in the literature. According to [19], the dissemination of false information is far and wide, as its debunking takes 12 hours on average since the content is published. During this time period, false information propagates expeditiously, because an unverified and recent rumor is likely to become viral [20]. In [3], the authors calculated the effect of Wikipedia hoaxes using the number of views and the number of days, during which the content had not been debunked. Their results are demonstrated in Figure 2.2.a which presents the relation between the time duration of a hoax article until it is found to be fake. It indicates that even though 90% of hoax articles are debunked within an hour, approximately 1% can survive for more than a year. Yet a successfully spread hoax article needs also to be viewed. In Figure 2.2.b the duration curve of average number of views of hoaxes (or non-hoaxes) articles that sustain over a week is presented. Generally, hoaxes are seen less regularly than non-hoaxes pieces, however 1% has at minimum 100 daily views. Moreover, the influence of hoaxes can be calculated through their dissemination on the web, using the clicked links by readers to attain the news content. It is shown that from approximately 7% of these articles, at least 5 unique links were clicked. Consequently, even though most hoax articles are unsuccessful, a low percentage of Wikipedia hoaxes has a thorough effect.

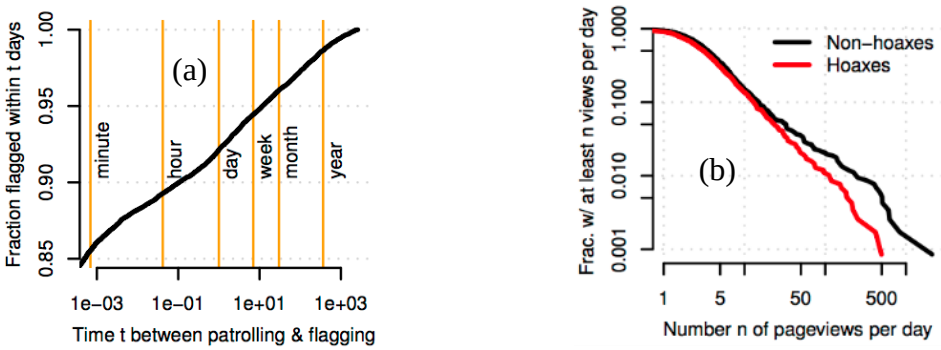

Figure 2.2. (a) Distribution of time survival of hoax articles and (b) the number of daily hoaxes and non-hoaxes views [3]

Buzzfeed news examined fake political news on the web with the highest effect, by analyzing real and fake election-related articles that have the highest spread on Facebook during the 2016 US elections [21]. The degree of impact was measured using the shares, responses and comments counts on the news content. They compared the 20 top fake election news produced by known fake websites and blogs, with the top 20 real news originating from famous websites, such as Washington Post, New York Times etc. Fake news pieces had 8.7 million involvements, while the real news contents had 7.3 million involvements.

The authors in [22] studied the dissemination of real and false information on Facebook. They gathered, 4.7 thousand rumor contents from snopes.com website, which classifies known news pieces on social media and examines their veracity. Their analysis indicated that rumors have high visibility and propagation of false information is deeper than true stories, due to the higher number of reshares.

Another research of false information dissemination is presented in [23], where the authors analyze 126 thousand rumors on Twitter. They used real and false information contents obtained by fact-checking websites and followed their dissemination until their origins. They demonstrate that false tweets have higher number of users that retweeted and they are diffused much faster as they have higher number of retweets within a brief time period. On top of that, they are deeper, as they generally have more retweet hops and they are broader, since they have more twitter users in a specific retweet depth. These facts were highlighted in all topics of fake news like science, urban legend, politics (in higher degree), business etc. They also observed, that the top false-information tweets attained over 1 thousand users contrary to tweets with true information. Furthermore, it is shown in the literature, that false information on social media and web has attained remarkably high numbers a fact that indicates their thoroughly contagious nature.

2.3 Fake news categories

According to research studies [4], [16], [24], the categories of fake news can be presented as follows:

- **Propaganda** [4], [24]. Propaganda is defined as news contents that are created by a political party to affect public opinions and aim to improve profit of public individuals, government or organization. This type of fake news derives from the World War and are mainly used for political reasons, in particular to deceive people with the purpose to damage a specific political group. Therefore, propaganda is considered an impactful type of disinformation as it can result in severe history changes, such as the perturbation of the election results.
- **News fabrication** [4], [16], [24]. Fabricated news is defined as a false content that has no factual evidence and is published in a legitimate style of news content. Therefore, it intends to misinform the audiences. This type of fake news is usually issued on a blog, website or generally on social media interfaces, such as Twitter and Facebook. The problem of discovering fabricated false information arises especially when partisan groups share such content, which seems to be unbiased and fair reporting.
- **Imposter content** [16], [24]. Imposter is a type of fake news that try to look like the famous media brands. In other words, is impersonates genuine sources to mislead the audiences. It can be verified by using the information of source and author.
- **Satire** [4], [16]. Satire news is articles that have the intention to mock news programs by producing humorous, yet fake, content. A related example is the comedy shows, where they concentrate on current events and individuals, usually utilizing a television news report style. Contrary to the previous fake news types, satire is for entertainment purposes that is performed by comedians rather than journalists and uses humor to attract the readers. However, this type of false information can inadvertently mislead audiences.
- **Parody** [16], [24]. Parody type of fake news is very similar to satire as both use humor to attract readers. Furthermore, it imitates the current popular news content and highlights its belief using a humiliating manner. Political parody has a major influence on the audience as it can be affected due to hyperpartisan and biased political beliefs.
- **Manipulated content** [4], [16]. Manipulated content refers to the twisting of original information, which could be either a photo, a video or a text. Image manipulation is produced by automatic tools and is difficult to distinguish with

human eyes. Therefore, it can significantly deceive readers judgements. Some plain techniques of photo manipulation are the increase in color saturation, the removal of some elements and on top of that, the inclusion of specific individuals in a photo which can significantly distract readers.

- **Advertising and public relations** [4]. Advertising and public relations refer to advertising in an illegal way certain product within an authentic news statement with the purpose of financial profit.
- **False context of connection** [16]. False context of connection is a reliable content that also includes false contextual information. An ordinary example is a title of an article which does not appeal to the news piece.
- **Conspiracy theories** [24]. Conspiracy theories are news pieces that use conspiracy to interpret an event without any evidence. They often refer to illegitimate activities of the government or known political parties, which consequently can severely harm the audiences and create anger. Moreover, they use unverified information as evidence to boost their credibility.

The authors in [24] also make a taxonomy of false information according the degree of impact on the recipient users. They distinguish 3 different scales; fake news, biased or inaccurate news and misleading or ambiguous news. Fake news has the greatest impact and includes fabricated content, propaganda, imposter content and conspiracy theories. Less severe is considered the biased news group, which consists of hoax articles, hyperpartisan content (extremely supporting a specific party) and fallacy (news that use faulty reasoning to support its claim). Misleading news is classified as the gentlest group and has minor effects on the audiences. It includes rumors (news content on the web that is ambivalent or not verified), clickbait (articles that use a misleading title that has no connection with the main body with purpose to attract many “clicks”) and satire news.

2.4 Fake news data types

The authors in [25] categorize the different data types that fake news can be presented as follows:

- **Text**: Text consists of the headline and the main article. It presents an intense appearance with capitalized letters, high polarity (usually negative) and certain grammar characteristics. The headlines in some occasions have no semantic connection with the main article. Text-linguistic and semantic techniques can be utilized to detect such attributes.
- **Multimedia**: This category consists of video, images, graphics and audio. They are distorted contents, or they present graphic contents to attract viewers' awareness. Computer vision and deep learning techniques can be used to detect specific patterns and outliers on the content.
- **Hyperlinks or Embedded content**: Hyperlinks allow writers to connect to external news sources that can be used as evidence for their arguments and hence obtain audiences' trust.
- **Audio**: This data type is appeared less frequently as a means of reporting news. It is implemented in radio/e-radio broadcasts and use only voice and sounds. Speech recognition methods are utilized to detect irregularities in the tone or the words used.

This thesis studies fake news detection using only raw text data type.

2.5 Fake news detection approaches

In [11], [13], [25], the authors categorize fake news detection methods as linguistic-based, visual-based, user-based, post-based, and network-based.

- **Linguistic-based**. Linguistic-based methods are utilized to discover irregularities in the language style and content, such as clickbait articles, which do not have any connection with the main body of the text with the purpose to gain users attention. They refer to linguistic or readability features for document organizations of characters, words, sentences and paragraphs, such as the number of words, number of characters for each word, frequency of big words and number of unique words. Moreover, research studies suggest the utilization of LIWC (Linguistic Inquiry Word Count) [26], [27], which is a transparent text analysis program that counts the number of words associated with specific psychological types and extracts

psycho-linguistic features. This software tool can extract information about language emotion, text statistics and part-of-speech (POS) tagging. Furthermore, linguistic-based approaches include syntactic features such as bag-of-words text representation [28]. In the bag-of-words models, every unique word or n-gram (sequence of n words) is a feature and is represented with its frequency in the document or its term frequency penalized by the frequency in the collection of documents (Term Frequency-Inverse Document Frequency, TF-IDF), which indicates the degree of importance of a word. Other examples of syntactic features refer to existence of specific punctuations (question mark and exclamation mark), which help to distinguish between real and deceptive news articles. Another syntactic method for feature extracting is called CFG (Context-Free-Grammar), which is a "set of rules utilized to construct patterns of strings" [25]. Finally, lying-detection features can be utilized to identify deception [29].

- **Visual-based**. Visual-based approaches refer to detection of fake news characteristics in images or video. Visual features in fake images are extracted using machine learning and deep learning techniques. They can also be detected by analyzing user and tweet attributes. Moreover, visual features such as clarity score and coherence score can be used to improve performance of the machine learning model [30].
- **User-based**. Other research studies suggest analyzing user-level characteristics. Understanding user profiles and engagements is critical for fake news detection. As an example, fake profiles (i.e. automatic bots) are used to provide support to fake news items and consequently amplify their dissemination. These characteristics are classified in individual and group level. Individual level features refer to specific user's characteristics, such as number of tweets and followers as well registration date. Group level features correspond to groups of users with certain attributes that can form communities and can be calculated using aggregation of user profiles.
- **Post-based**. Another approach for fake news detection is post-based, which studies user responses in a fake news content. Skeptical opinions and intense reactions could indicate existence of fake news piece and provide valuable information. Each user's comment can be represented using bag-of-words or word embeddings [31] to obtain user opinion, topic and the degree of reliability. Topic features can

be obtained by the Latent Dirichlet Allocation (LDA) technique [32]. Finally, all these features can be aggregated to extract group level characteristics.

- **Network-based**. Spread of fake news composes an echo chamber cycle that suggests exploiting network-based techniques. Network features can be obtained by creating several types of networks with similar media posts, such as stance, cooccurrence and friendship networks. More specifically, stance networks are graphs having as nodes all the posts that are relevant to the news and as edges the weight of similarity between two articles. Another type of network is the cooccurrence network, which evaluates if users write articles in the same category of news. Finally, friendship networks (followers) indicate the dissemination degree of a news piece.

The authors in [33], studied rumors on Twitter. They gathered 10,000 posts, which were manually labeled by fact-checking agents and implemented bag-of-words text representation using also n-grams and POS tagging. They also extracted information from users' profile and specifically if a user has previously shared fake news content as well as they utilized information from hashtags and links on Twitter. The results indicated 95% score of mean average precision in this dataset with the text-related features having higher contribution to the classifier's decisions.

The authors in [26] studied the use of content-based features for fake news detection. They gathered fake news articles annotated by Amazon Mechanical Turk (AMT) workers. Subsequently, they extracted features, such as bag-of-words (TF-IDF), LIWC, readability, syntactic and punctuation features and fed them as input to the machine learning model. The model reached 74% mean accuracy which even outperformed human annotators' effectiveness (70%).

In [3], the authors study user, network and meta-data-based approaches for hoax articles detection in Wikipedia. They deployed four distinct types of features: appearance, network, support and user features. Appearance features correspond to the article's length, the number of linked sources, etc. Network features evaluate the relation between external linked sources and the Wikipedia hyperlink network. Support features check whether the headline of news content has the same name with other articles and calculate the time difference since the generation of the news piece. User features refer to users' profile history such as the user's registration date and assess whether the user has

previously published fake news content. Network and user attributes achieved highest individual accuracy scores and the combination all features boosted the performance of the model.

A research study in [18] deals with fake twitter posts that also contain images. They extracted user-related features such as number of connections and time since registration. From the tweet content, they utilized linguistic features regarding words count, sentiment scores, POS tagging, etc. They also utilized meta-data information such as count of hashtags, links and retweets. The dataset consisted of 11500 tweet posts with balanced proportion of labels. Decision trees were selected as machine learning model and achieved 97.7% accuracy.

A more advanced approach for fake news detection is presented in [34]. The researchers propose a deep neural network called TI-CNN, which consists of multiple convolutional layers and combines as input text and image information. This model can also capture hidden patterns found in text and images or latent relations between them due to the deep learning architecture. Their results indicated that deep learning approaches can boost the accuracy of fake news detection methods, especially when combining text and image knowledge.

2.6 Fake news datasets

Previous research studies have dealt with the problem of disinformation and created datasets that can be used to implement and evaluate supervised learning techniques.

- **BS Detector**[3]. It contains only unreliable articles, most of them referring to politics. This collection of news items represents a basis for generating a fake news corpus to train text classification models. Subsequently, the author in [35] constructed a fake news dataset, which is evenly split in reliable and unreliable news contents. More specifically, he incorporated articles from Kaggle fake news dataset to comprise the unreliable news pieces and scraped articles from All Sides, which is a website dedicated at hosting news from widely known news providers that are considered reliable. This dataset was selected for conducting the

[3] https://www.kaggle.com/mrisdal/fake-news/home

experiments of this thesis, as its topic corresponds to political news and provides binary annotations (real news or not). It is described in more detail in chapter 3.

- **LIAR**[4] [36]. The dataset contains 12836 short news statements over the past 10 years, obtained from politifact.com, which is a website dedicated to check the truthfulness of social media content by using fact-checker agents. The annotated labels include 6 truthfulness ratings: true, mostly-true, half-true, barely-true, false and pants-on-fire. However, it is difficult to automatically distinguish between these types, as the current machine learning methods implemented on this dataset achieved nearly 30% accuracy.
- **CREDBANK**[5] [37]. CREDBANK is a dataset that consists of 80 million tweets (metadata only) that refer to specific topics (consisting of the top 3 terms returned of a topic modeling method called Latent Dirichlet Allocation). The tweets were gathered from October 2014 to February 2015 (96 days). CREDBANK also contains a file that consists of 62000 topics with labels indicating that it is an actual event or not. Finally, it includes a file comprised of 1300 events. These events were evaluated by 30 Turkers with credibility rankings of ranging from -2 to +2 and their analogous explanations for this classification. Nevertheless, it limited to keywords of the topics without including the event's full text as well as the related tweets.
- **BuzzFace**[6] [38]. BuzzFace contains 2263 news stories on Facebook and is empowered by 1.7 million comments discussing these news contents. Comments provide essential information regarding the veracity of the content, as it denotes public reaction to a news piece and how likely it is to be spread over the social media. Nevertheless, the number of the news stories is small for developing a natural language understanding model that can identify the intent to harm from the text.

[4] https://www.cs.ucsb.edu/~william/data/liar_dataset.zip
[5] https://github.com/compsocial/CREDBANK-data
[6] https://github.com/gsantia/BuzzFace

2.7 Black-box model interpretation methods

2.7.1 Introduction

Several fake news detection algorithms have been developed, yet, to the best of our knowledge, no one provided explanations of the black box model predictions. Model explanations can provide transparency of the classifier's behavior, helping the user to better understand the reasons that lead the machine learning model to classify a news item as fake.

In [39], the author defines interpretability as the level at which a human can understand the reason of a classification. He also provides an alternative definition which is the level at which a human can predict the classifier's response. Higher interpretability of a black box means easier comprehension of specific outcomes. However, interpretability differs from explanation. Explanation is defined as an answer to a why question, which in case of machine learning, is "Why the black box model predicted an instance as a specific label?" or "Why should I trust this model?". It provides an overview of the most notable features for individual predictions that will help users understand if it works correctly or is biased against certain attributes. For example, a model that predicts if an applicant for a loan should be accepted or rejected can be affected by the person's gender or living location. Subsequently, any problem observed can be corrected, for instance, by eliminating these attributes. In text classification, an explanation could be the set of words that contribute more to the classification. Humans prefer short explanations, because they are easier to comprehend.

A good explanation comprises of some of the following properties:

- It is **counterfactual**: Many people prefer to know why the classifier predicted a specific label instead of another prediction. In other words, humans can better comprehend when comparing to opposing examples.
- It is **selected**: Reading a whole document is difficult and time consuming for the human to understand the decisions, hence the explanation should comprise a limited number of words or phrases that have higher contribution to the model's response.
- It is **social**: Explanations need to adapt to the specific user preferences. For example, a user on social media that uses a fake news detection model would prefer a

simple explanation for predicting an article fake, while a journalist would prefer a more descriptive and knowledge-based approach.

- It focuses on **abnormal causes**: If an input attribute for a classification is abnormal (such as a rare category of a categorical feature) and has high influence on the prediction, it should be mentioned in the explanation.
- It is **truthful**: Explanations should be also able to predict a label in similar examples, i.e. they should be faithful.

Methods for black box model interpretability can be categorized according to different criteria:

- **Intrinsic** or **post hoc**. Intrinsic approach indicates the selection of an interpretable model for training using the given dataset. It is considered as the simplest form of transparency of the model's decisions, however the predicted accuracy is limited as complex models (such as deep neural networks) outperform the plain machine learning models. Post hoc approach uses the predictions from a pretrained classifier to obtain model transparency, which is the method to extract explanations for black box model's responses and constitutes the goal of this thesis.
- **Type of explanation outcome**. Many methods provide a feature summary statistic indicating the degree of the feature's influence on the model outcomes. This statistic can also be visualized using graphs that are more comprehensible to humans. A different approach is the learned weights using a surrogate linear model that is trained on the predictions of the black box model.
- **Model agnostic or model specific**. Model specific methods provide interpretability for specific types of classifiers, such as neural networks and tree ensembles. Model agnostic approaches are not based on the type of the model and are usually post hoc. They work by examining different combinations of features and their respected model responses. These methods use no information about the internals of the machine learning models, like weights or structural representation.
- **Global or local**. Global approaches explain the model behavior, while local methods explain individual predictions of a black box classifier.

2.7.2 LIME

The authors in [40] propose a model called LIME (Locally Interpretable Model-agnostic Explanations). This method provides explanations of individual predictions by creating a neighborhood sample of the predicted instance, which is used as training set for a surrogate interpretable model that subsequently learns the weights of the features (words). The words with highest weights represent the explanation of the classifier prediction. It is based on two fundamental characteristics: interpretability and local fidelity. The explanations must be fully comprehensible to humans and explain model prediction. Furthermore, the created neighborhood must be locally faithful. In other words, it must be a good representation of how the model behaves in the surrounding of the tested instance. Figure 2.3 shows the areas (in the feature space) that the black box model learned to classify. It is observed that the separating curve of the decisions of the classifier is highly complex, however, if we zoom in, it can be locally approximated with a straight line (linear model). The bold red cross denotes the predicted instance that is explained, while the size of each sample indicates how distant it is to the explained instance.

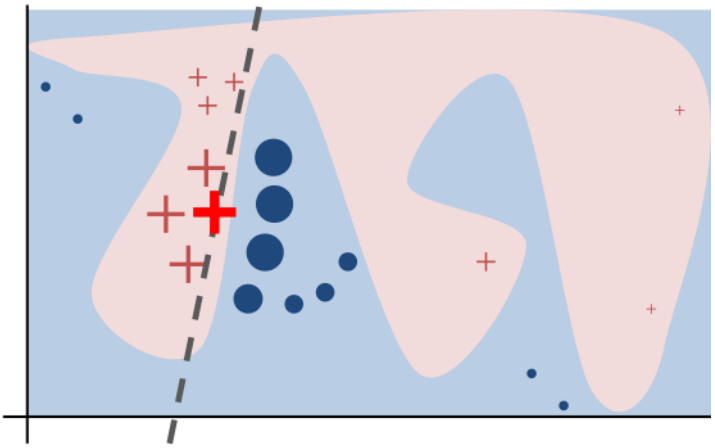

Figure 2.3. Areas of the decision function of a binary classification with two input features. The points denote the sample created around the predicted instance (bold red cross) [40]

The general steps of LIME method are as follows:

1. Choose instance to explain the classifier prediction
2. Perturb dataset and obtain the classifier predictions for these new points.
3. Weight the new samples according to the distance from the tested instance to create new dataset.
4. Train a weighted, interpretable (linear) model on dataset.
5. Explain prediction by interpreting the local model (using the weights).

In text classification, the text is transformed in the bag of words approach where each word has a binary representation denoting its presence. These extracted features are sampled uniformly around the instance in the vector space to create the training set. Given this dataset of perturbed samples, the goal is to minimize $L(f, g, \pi_x)$ which is a measure of how unreliable the interpretable model g is.

$$\xi(x) = argmin_{g \in G}\ L(f, g, \pi_x) + \Omega(g)$$

where f is the black box classifier, π_x is a measure that indicates the distance from the tested instance and Ω denotes the complexity of the interpretable model. More specifically, g is a linear model such that $g(z) = w_g \cdot z$, where z is a vector, $\pi_x(z) = \exp(-D(x, z)^2/\sigma^2)$ is a kernel for the distance function, where D is the cosine similarity and Ω is the number of features for the linear model. The loss function that must be optimized, is weighted by the kernel function π_x which leads to:

$$L(f, g, \pi_x) = \sum_{z, z' \in Z} \pi_x(z) \cdot (f(z) - g(z')^2$$

The user can provide the number of words k that constitute the explanation and the algorithm returns the k words with the highest weights of g and hence the higher contribution to the individual prediction.

2.7.3 EDC

In [41], the authors define explanation as a minimal set of words in the classified document, so that deleting all these words, changes the outcome of the black box model. For example, in a sentiment analysis task, removing all the positive words {good, excellent, wonderful, love, etc.) from the instance, results in different prediction of the classifier (negative or neutral). Therefore, this set of words determines the model's response and can be used as an explanation. We used verbatim definition of document explanations from the [41] as follows: "*Consider a document D consisting of m_D unique words W_D from the vocabulary of m words: $W_D = \{w_i, i = 1,2, \dots, m_D\}$, which is classified by classifier $C_M: D \rightarrow \{1,2, \dots, k\}$ as class c. An explanation for document D's classification is a set E of words such that removing all words in E from the document leads C_M to produce a different classification*". In other words, set of words *E* is an explanation (1) if the words appear in the text, (2) if *E* is removed the predicted class changes and (3) *E* is minimal.

Selecting the minimal set of words is intractable, due to the vast number of word combinations. Consequently, they suggest a greedy approach for extracting the optimal

explanations of a document classification. The representation of a text instance is transformed to a bag of words representation indicating the presence or absence of each word (feature). It is also considered that the black box model has a predict-probability function, which returns the prediction probability of a given feature vector for a specific class. Then, the algorithm uses a heuristic best-first search, where at each iteration it selects the word that when is removed, the probability of classifying the given document to the predicted label changes at most and the word is appended to the explanation set. When the removal of this set changes the outcome of the classification, it can be used as an explanation. For every minimal set of words obtained, there is no need to further search for candidates that contain these words. Therefore, they include a pruning step, that removes these candidates. The algorithm terminates after a specific number of iterations given by the user. The general steps of EDC procedure are demonstrated as follows:

1. For each word: remove word from document and predict label probability.
2. Choose word that when is removed changes prediction probability the most.
3. Append to set E.
4. If removing all words from E, changes the predicted label, then E is an explanation.
5. Else repeat with remaining words.

This method is model-agnostic and selects features in an instance-wise manner, because it only uses the outcomes of the complex classifier given an input and explains individual predictions without using information from the rest of the dataset that the black box model was trained.

2.7.4 Other explanation algorithms

Based on LIME approach, KLIME method [42] splits the data into k partitions using unsupervised clustering techniques (e.g. k-means) and trains linear interpretable model for each cluster separately. The value of k is selected aiming at maximizing the R^2 of the local models' predictions. This algorithm nonetheless lacks in preserving the underlying model structure due to the unsupervised approach. A solution to this problem is LIME-SUP, which is based on KLIME method but uses supervised techniques (decision tress) for splitting the dataset into k partitions. Therefore, it resembles the original model and creates more meaningful clusters. Given a new instance, both interpretability models

classify it to the most similar cluster and use the local linear interpretable model to explain the prediction.

The authors in [43] propose an extent on LIME that is based on high-precision rules called anchors. They define anchors as the subset of words that, even if the remaining words in the text instance are perturbated, the model predicts the same outcome with high probability. Anchors are easy to understand and are only implemented when all predicates in rule are satisfied. In text classification, instead of using perturbation technique proposed in [40], the creation of text's neighborhood happens by randomly replacing words with words with the same POS tag and with high similarity in an embedding vector space.

The interpretable representation we use is the presence of individual tokens (words) in the instance. The perturbation distribution D replaces "absent" tokens by random words with the same POS tag with probability proportional to their similarity in an embedding space.

Formally, given a black box model f and a text instance x, an anchor A is a set of words from x that has greater precision, $prec(A)$, of the model's predictions than a threshold τ, where:

$$prec(A) = E_{D(z|A)}[f(x) = f(z)]$$

However, computing this precision directly is prohibitive and therefore the authors propose a probabilistic definition that anchors must meet the aforementioned constraint with big probability as follows:

$$P(prec(A) \geq \tau) \geq 1 - \delta$$

In case of multiple anchors satisfying this constraint, those with the highest coverage of the input space are selected. Subsequently, the algorithm extracts candidate anchors using a multi-armed bandit formulation and selects those that optimize this criterion.

This algorithm is very useful as it can also handle explanations of phrases of words, such as bigrams. For example, the phrase "not bad" indicates positive sentiment, but if the word "not" is missing, it could confuse the explainee as it would provide an opposing explanation. It anchors the subset {not, bad} and understands that these words together have extremely high precision $prec(A)$ of the classifier's predictions.

The authors in [44], discuss the fundamental problem of creating the neighborhood of an instance in locally surrogate models used for model explanations and propose a novel approach that is relevant to the predictions made. They claim that sampling instances globally from the input vector space focus mostly on features that have global influence, while concealing the features with local impact. Furthermore, they state that randomly sampling around a center point could provide a highly imbalanced training set for the local surrogate to fit the black box decisions.

The general idea of the proposed method is that it also uses the boundary of the decisions of the black box classifier to construct the neighborhood instance. First, the algorithm produces instances in a hypersphere of growing radius which is centered on the instance vector. Once the radius reaches the boundary of the black box model's decisions, new instances are randomly created uniformly in a hypersphere S centered on the boundary point with a specific radius r_{Sx}, which eventually creates a balanced training set for the local model. Radius r_{Sx} is considered a measure of locality.

The authors in [45] propose the use of hierarchical interpretation to extract explanations of deep neural networks' predictions through agglomerative contextual decomposition (ACD), which provides a hierarchical clustering representation of word features and finds the contribution of each cluster to the classifier's outcome. This hierarchy is trained to identify the most representative groups of words to the predictions.

For a given deep neural network *f(x)*, we can denote its outcome as a SoftMax process applied to sigmoid functions *g(x)*. These functions are the composition of L layers, such as convolutional, rectifier linear units (ReLU), max pooling and dropout layers. Contextual decomposition algorithm $g^{CD}(x)$ decomposes the sigmoid functions g(x) into a sum of two terms, the first one indicating the importance measure of a feature group and the second one highlighting contributions to g(x) not comprising the first term. Once the contextual decomposition scores are calculated for each attribute, this algorithm iteratively keeps the 90% of the text that has the highest score to create the hierarchy. The procedure is terminated when all words are selected.

This method is considered successful at identifying incorrect classifications and dataset bias and can be for any neural network architecture and data type. Furthermore, it is very robust to adversarial perturbations of the data. For example, in sentiment classification the phrase "not very good" indicates negative sentiment but many algorithms can be confused and provide only the phrase "very good" as positive contribution.

The authors in [46], define an explainer as an instance-wise feature selector, which returns the subset of features that are considered the most informative for the specific model response. The goal is to maximize the mutual information of the selected subset and the model outcomes. Mutual information is a measure of dependence between two random variables X, Y and is defined as:

$$I(X,Y) = E_{X,Y}[log\frac{p_{XY}(X,Y)}{p_X(X)p_Y(Y)}] \Rightarrow$$

$$I(X,Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\cdot \log\frac{p(x,y)}{p(x)p(y)}$$

For example, if *X*, *Y* are independent, then $p_{XY}(X,Y) = p_X(X)p_Y(Y)$ which entails that:

$$I(X,Y) = 0$$

The general idea of this method is that it learns globally a local explainer by taking into consideration the distribution of inputs in relation with the specific model outcomes.

3 Materials and methods

This chapter describes the methods implemented on this thesis for fake news detection as well as the explanation algorithms for the predictions of the developed text classification models.

3.1 Introduction

As previously mentioned, the goal of this thesis is to detect fake news by using only raw text and give explanations of the machine learning model decisions for better transparency and, hence, evaluate if the classifiers predict in an unbiased and fair manner. We implement a framework that uses raw text (news article) as input and predicts if the instance is a real news piece or not and explains the prediction by highlighting the most significant words for the classification. According to [11], fake news detection is defined as "the task of predicting whether a news article is fake news piece or not" which is a binary classification task. In this thesis, we examine the task in a different perspective and define fake news detection as the task of predicting if a news piece is real or not. The reason for this approach is that it is easier to distinguish reliable from unreliable news content, as it is factual, precise and needs little interpretation. On the contrary, distinguishing fake from non-fake news is harder; there are cases (e.g. satirical news) where it is extremely difficult to identify and distinguish from (actual) fake news.[47]. An overview of our models is illustrated in Figure 3.1. First, input text is filtered so as to only keep the essential information from the content. Then, we transform cleaned text to a vector representation that constitutes the input features for the machine learning model. To evaluate the classifier's performance, we randomly split the dataset into training set and test set. The machine learning model is trained on the selected training dataset and is evaluated with the accuracy of the test set predictions. The trained classifier, which is considered a "black box" model, is interpreted using post-hoc approaches and an explanation set of words is returned to better explain why the model predicted the specific instance as real or not real. The design and experiments of this model were deployed by using Python programming language, which is an object-oriented language that is

regularly utilized by the data science research community, due to the huge number of libraries that are publicly shared online.

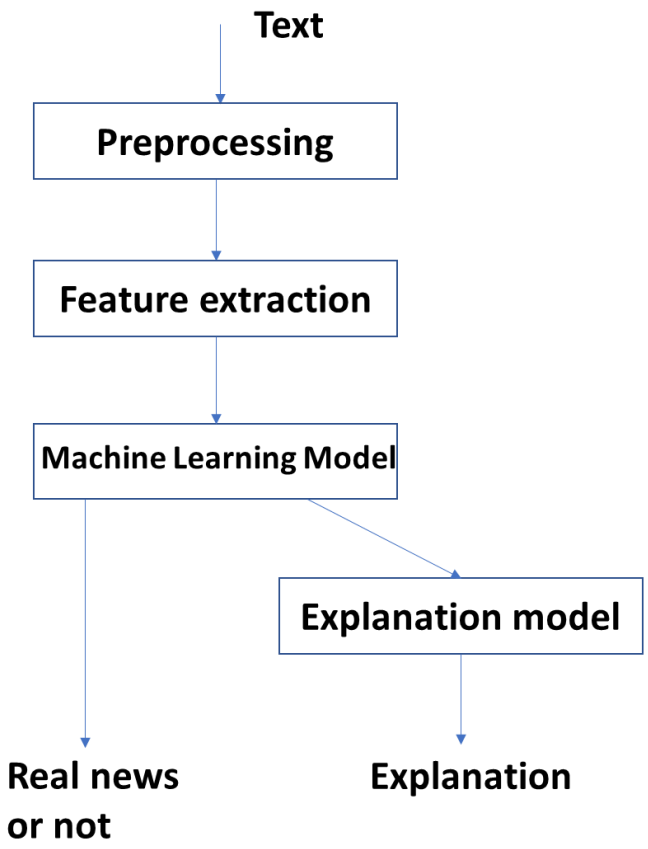

Figure 3.1. Overview of the proposed model. Text is cleaned and then converted into vector representation to be fed as input in a machine learning model. The interpretable model utilizes the predictions of the machine learning model to extract explanations.

3.2 Dataset

The dataset used in this thesis, is described in [35] and can be found online on GitHub. It consists of 6335 articles equally divided into real and non-real labels. Non-real news contents were obtained from a fake news dataset comprising of 13000 articles published during the US election period on November 2016 and can be found online on Kaggle. This dataset was created with the utilization of BS Detector[7], which is a chrome extension tool that uses a curated list of non-reliable websites from *opensources.co* and warns users about unreliable news sources. The articles were scraped using the *webhose.io* API

[7] https://bsdetector.tech/

annotated with labels such as fake news, satire, extreme bias, conspiracy theory, state news, junk science, hate group and bs. BS is a label that BS detector returns when it cannot specifically identify the correct annotation, but the news item is considered unreliable. It is mentioned, that for our problem all these different labels were considered generally as non-real news pieces. On the other hand, real news pieces were gathered from All Sides, which is a website that hosts news and stance articles from across the political spectrum and is a considered reliable source of information.

Each article is associated with the title and main text, which we chose to combine as a single feature, because words in title tend to be repeated in the main text. And, consequently, the contribution of these words is increased.

3.3 Data cleaning

Text is an unstructured form of data and could contain noisy content and thus text cleaning actions pose a necessary step for classification. In many data science problems, it requires almost 60% of research time, because the documents must be manually checked for inconsistencies that could prevent incorrect training of the classifier and have to be removed. We used the Natural Language Toolkit (NLTK) and Regular Expressions (RE) Python libraries for the implementation of data cleaning that perform string operations. The text cleaning techniques used in the problem, are presented as follows:

- **Lowercase**. Every alphabetical character was converted to lowercase, because Python language is case sensitive and hence cannot understand that two words with only different letter cases have the same meaning. As an example, first word in a sentence is always capitalized but its meaning remains the same. This technique can also help shrink the size of the vocabulary.
- **URL removals**. Many articles share external links to refer to specific sources. However, in text classification task, they have no actual meaning. Therefore, we replaced any string containing “http” or “https” with “<url>” to keep only the information that it is a link.
- **Punctuations**. In many cases, punctuations do not share any explicit information to the readers. We chose to keep dots (.), because they show the break of the sentence, as well as we did not remove question marks (?) and exclamation marks (!), because they indicate polarity in the text, such as anger or admiration.

- **Numbers removal**. Usually, numbers indicate a scale about something or denote the time, date, month etc. This makes difficult for the machine learning models to understand their meaning, because numbers can take infinite values, but on top of that they depend on the subject that they refer to. For example, the sentence "Greece has a population of 1 billion citizens" it is known false information because Greece has only 10 million citizens and this value is considered significantly high. Nevertheless, the sentence "A year consists of 5 months" is also wrong, even though 5 is considered a small number. Therefore, we deleted any numerical character from the text.
- **Tags removal**. Tags such as "@username" are widely used on social media to refer to a unique user. However, usernames are hard to be interpreted from the machine learning model and hence we chose to remove related tags.
- **Replace words containing apostrophe**. An apostrophe can be used to replace a letter (or letters). For example, word "won't" comes from the phrase "will not". Therefore, we replaced such words containing apostrophe with the original phrases using vocabulary of such cases.
- **Tokenization**. After removing noise found in the text, it can be tokenized. In other words, the document, which is a large string, is converted to a list of words (tokens)
- **Stop-words removal**. Stop-words are words that are used frequently in the text but have no actual meaning, such as the articles a, the, etc. There is no agreed list of stop-words in the literature that outperforms in text classification tasks. We utilized the vocabulary of stop-words provided from the NLTK library and removed such words from the documents.

Another efficient technique for text preprocessing, but have not been implemented, is stemming because stemmed words cannot be easily interpreted by humans. Stemming is applied to recognize and maintain the root (stem) of a word with the purpose of dimensionality reduction. As an example, the words "accept", "accepted", "accepting", "acceptance", which have the same sense, are converted to the word "accept". We avoided utilizing this approach, because explanations that contain stemmed words are hard to be interpret by humans. Lemmatization refers to morphological examination of the words [48]. This method clusters the diverse forms of a word into a single term. However, it requires POS tagging of each word in the text which is vulnerable to errors.

3.4 Feature extraction techniques

3.4.1 TF-IDF

A widely known feature extraction technique is TF-IDF (Term Frequency – Inverse Document Frequency) [49]. TF-IDF is a bag of words approach where each unique word is considered a feature. This can also be extended in phrases of words (n-grams) such as bigrams and trigrams. This method determines how relevant a given word is in a specific document. It is a common term weighting scheme in information retrieval, that has also found effective use in document classification.

The TF-IDF value for a given term *t* in a document *d* is defined as:

$$tfidf(t,d) = tf(t,d) \cdot idf(t)$$

where $tf(t,d)$ is the term frequency of term *t* in document *d* and $idf(t)$ is the inverse document frequency.

Term frequency can be calculated in numerous ways:

1. **Raw frequency**:
   $$tf(t,d) = f(t,d)$$
   where $f(t,d)$ is the number of times that term *t* occurs in a document *d*.

2. **Boolean frequency**:
   $tf(t,d) = 1$ if *t* occurs in *d* and 0 otherwise.

3. **Logarithmically scaled frequency**:
   $tf(t,d) = 1 + \log f(t,d)$ and 0 when $f(t,d) = 0$

4. **Normalized frequency**:
   $$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d)\}}$$
   which is the raw frequency of term *t* in document *d* divided by the maximum frequency of any term *w* in the document and can prevent bias towards longer documents.

In this thesis, we used the raw frequency to denote the term frequency metric. Therefore, a word is more relevant to a document, if it appears many times in the document.

Inverse document frequency of a term *t* in a collection of documents is defined as:

$$idf(t) = \log\left(\frac{N}{df(d,t)}\right) + 1$$

where N is the total number of documents in the collection and $df(d,t)$ is the number of documents that contain the word *t*. This metric indicates that terms appearing in fewer documents (rare terms) will have higher weights. In other words, it penalizes words that appear in many documents without giving specific information regarding the uniqueness of the document. For example, stop-words occur in every document without providing any meaning to the document's subject.

We utilized the TfidfVectorizer function provided by sklearn library, which receives specific parameters determined by the user. Particularly, we limited maximum number of features (max_features) to 100000. This parameter considers the top max_features ordered by term frequency across the corpus, while ignoring the rest of features, which assists in reducing the dimensionality of our problem as a corpus could include millions of words. Furthermore, another parameter that can be configured is ngram_range. Generally, in text classification a feature is considered a single word. This can also be extended to n-grams of words, because phrases of words may have different meaning from a single word. For example, the sentence "I live in New York" contains the bigrams "I live", "live in", "in New" and "New York". It is observed that the phrase "New York" refer to a specific city and each word has different meaning. At the same time, it may have a high term frequency in the collection of documents. We selected ngram_range to contain unigrams, bigrams and trigrams. The selected parameters were obtained by tuning the hyperparameters with purpose to maximize accuracy in validation set using a simple Logistic Regression classifier.

An extension of n-grams is skip-grams [50], which is a widely used technique in text classification. This method also forms n-grams but on top of that, allows adjacent sequences of words. For example, the sentence "I live in Thessaloniki" forms three word-level bigrams: "I live", "live in" and "in Thessaloniki". However, the most important phrase of two words in this sentence is considered "live Thessaloniki", which can be traced using the skip-gram modeling by omitting the word "in". The authors define k-skips-n-grams for a sentence $w_1 \ldots w_m$ as the set:

$$\{w_{i_1}, w_{i_2}, \ldots w_{i_n} | \sum_{j=1}^{n} i_j - i_{j-1} < k\}$$

k refers to the number of skips and *n* to the number determines the type of n-grams (bi-grams, trigrams ,etc.) with 3-skip-n-grams (n-grams include unigrams, bigrams and tri-grams) case achieving highest accuracy. This method was implemented by manually configuring the analyzer parameter of TfidfVectorizer. After determining the TF-IDF parameters, the algorithm is fitted in the training set (corpus) and transforms documents into bag of words consisting of the selected features (n-grams, k-skip-n-grams) with the related TF-IDF value.

Nevertheless, 100000 features are considered a significant dimensionality size for text classification. To overcome this problem, dimensionality reduction techniques can be utilized such as Truncated Singular Value Decomposition (TSVD) [51]. Singular Value Decomposition (SVD) is a matrix analysis technique that reduces dimensionality of a high-dimensional array. In other words, it determines a smaller number of "concepts" that link the rows and columns of the array as well as approach the original array in good proximity. TSVD applies a variant of SVD that only calculates the *k* greatest singular values, where *k* is a hyperparameter determined by the user. Therefore, in our problem, given the transformed TF-IDF matrix *X* with *m* documents and *n*=100,000 features, the algorithm returns an mxn ($k \ll n$) matrix that can approximate the original matrix with high precision. In document matrices with bag of words representation, it is also referred as Latent Semantic Analysis (LSA), because it converts matrix to a vector space of concepts with low dimensionality. SVD is a technique that decomposes matrix *X* into a product of three matrices as shown in Figure 3.2. It is defined as:

$$X \approx X_k = U_k \Sigma_k V_k^T$$

where X_k is the transformed matrix with *k* selected dimensions, U_k is an mxk column-orthonormal matrix indicating documents-to-concepts similarities, Σ_k is a diagonal matrix where all the elements are zero except those in the main diagonal, which have positive values sorted in decreasing order denoting the weight for each concept and V_k is an nxk column-orthonormal matrix indicating features-to-concepts similarities. After this operation, the original training and test sets are transformed into two matrices with *k* features using the following formulas:

$$X'_{train} = U_k \Sigma_k^T$$

$$X'_{test} = X_{test} V_k$$

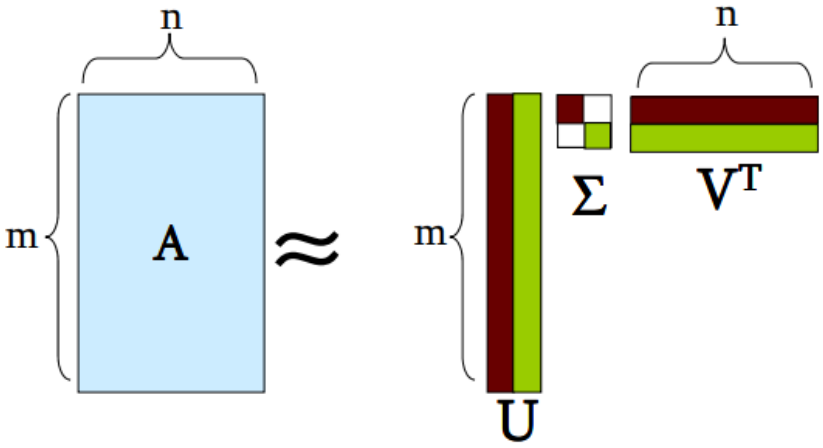

Figure 3.2. Singular Value Decomposition method [52]. A matrix is decomposed in a product of three matrices indication correlation with general concepts.

3.4.2 Graph of words

The authors in [53], [54], propose a novel approach for feature extraction in text classification tasks, which implements a Graph of Words (GoW) representation. More specifically, each given document *d* in a collection *D* is transformed into a graph $G_d = (V_d, E_d)$ where V_d corresponds to the set of nodes and E_d to the set of edges (links). Each node refers to the term *t* of a document and each edge to the number of times each pair of terms cooccur in document *d* within a sliding window of fixed size. An example of a document's GoW representation is demonstrated in Figure 3.3. This approach reproduces information about the relationship between pair of words or n-grams.

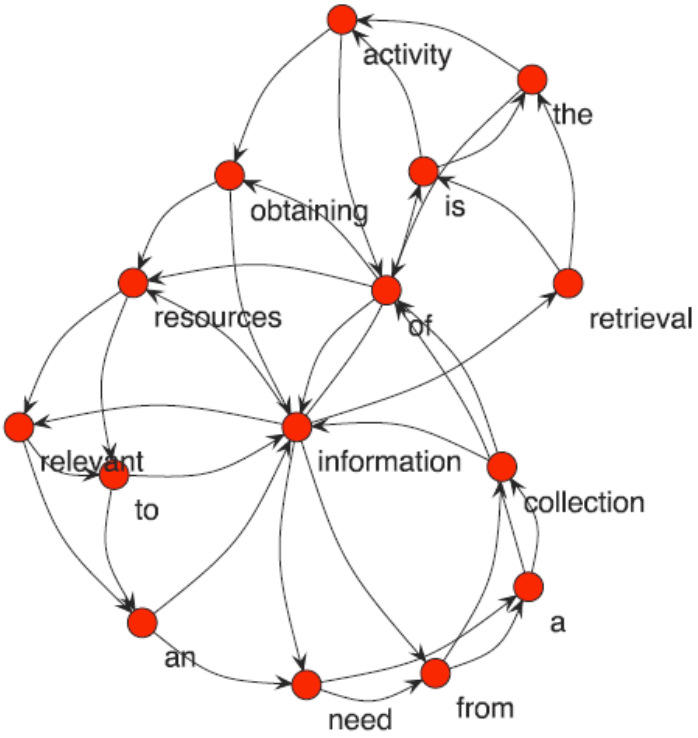

Figure 3.3. Graph of Words representation of text [54]. Nodes refer to words in document and edges correspond to cooccurrences between words within a sliding window.

Using network analysis techniques such as centrality criteria, a term weighting scheme that captures its importance in a document, can be correspondingly calculated with the term frequency in TF-IDF. More precisely, Term Weight (TW) of term *t* in a document's *d* GoW is defined as:

$$TW(t,d) = centrality(t,d)$$

As centrality measure, we utilize degree centrality is formulated as:

$$degree\ centrality(t,d) = \frac{\deg(t,d)}{|V_d| - 1}$$

where $\deg(t,d)$ is the degree of node (term) t and $|V_d|$ is the total number of nodes in graph G_d.

Other centrality metrics presented in [53] are closeness centrality and PageRank centrality. Closeness centrality corresponds to the length of shortest path between two terms. PageRank centrality is based on PageRank which is the core algorithm used by search engines to rank search results. It counts the number and quality of links to a term to estimate the ranking of a node. This metric can be extended by taking into consideration the

IDF value, which penalizes terms that appear in many documents, as defined in the equation. Therefore, the TW-IDF model is formulated as follows:

$$\text{TW-IDF}(t,d) = TW(t,d) \cdot IDF(t)$$

A similar approach to IDF is Inverse Collection Weight (ICW), which is a graph-based penalization concept that is based on a collection level. Given the graphs $G_1, G_2, \dots, G_n$ of documents $d_1, d_2, \dots, d_n$, collection-level graph *G* is defined as the union of each document's graph in the collection.

$$G_D = G_1 \cup G_2 \cup \dots \cup G_n$$

This graph is considered the degree of general dependencies between terms in the collection of documents. Through this definition derives the ICW metric, which denotes the maximum TW value in the collection divided by the TW of term *t*:

$$ICW(t,D) = \frac{max_{u \in D} TW(u,D)}{TW(t,D)}$$

TW-ICW model is formulated as:

$$\text{TW-ICW}(t,d) = TW(t,d) \cdot \log(ICW(t,D))$$

$TW(t,d)$ can be any centrality measure, nonetheless computational limitations must be taken into consideration.

The authors in [53] also propose a supervised term weighting scheme using GoW. First, we construct a graph for each label with terms as nodes appearing in documents with the specific classification. They define Label Weighting (LW) scheme as follows:

$$LW(t) = \frac{\max(\deg(t,L))}{\max(avg(\deg(t,L)), \min(\deg(L)))}$$

which is the maximum degree of term *t* in all label graphs (L) divided by the maximum value between the average degree of term *t* in L and the minimum degree of all terms in L. This metric is combined with $TW(t,d)$ and $ICW(t,D)$ as follows:

$$\text{TW-ICW-LW}(t,d) = TW(t,d) \cdot \log(ICW(t,D) \cdot LW(t))$$

3.4.3 Word2Vec

A different approach for text representation in natural language processing is word embeddings. Word embeddings capture the semantic information of words by reproducing them in a semantic vector space, where each dimension corresponds to a "concept" similarly with the TSVD method. A widely known framework that learns word embeddings representation is Word2Vec, which is available online on Gensim Python library. Word2Vec is an unsupervised learning technique that learns word embeddings from a collection of documents using contextual information integrated in a shallow neural network, i.e. a neural network that contains only one hidden layer. There are two main approaches of Word2Vec: Continuous Bag-of-Word (CBOW) model [55] and Skip-Gram model [56].

CBOW uses the word's context within a fixed-sliding window as input and the examined word as output. In other words, the neural network is trained to predict the current word from a window of surrounding context words. The simplest case of CBOW model is if we consider a sliding window size with only one word in the context and is demonstrated in Figure 3.4. Given a vocabulary of size *V* and the hidden layer with *N* units, each word is represented as one-hot vector of size *V*, with zero values except the cell that denotes the input word.

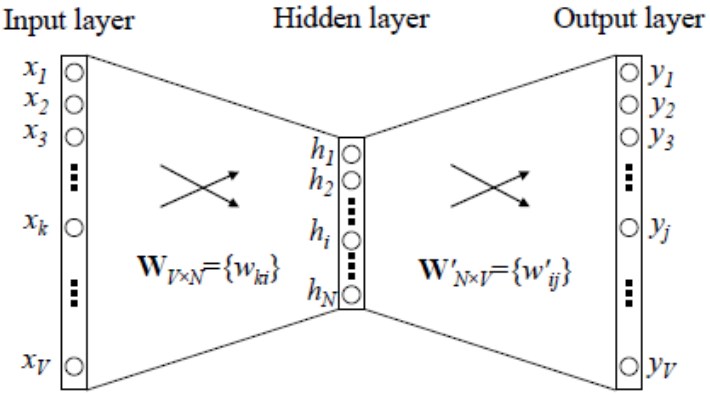

Figure 3.4. CBOW model considering only one-word context [57]. Input layer is a one-hot vector indicating the word unique ID. It is a shallow neural network that is trained to predict the next word of another term.

Therefore, the neural network in Figure 3.4 is mathematically formulated as:

$$\boldsymbol{h} = \boldsymbol{W}^T \boldsymbol{x}$$

$$\boldsymbol{y} = \boldsymbol{W'}^{T}\boldsymbol{h}$$

where $\boldsymbol{W}$ is a $V \times N$ matrix corresponding to the weights between the input and hidden layer, while $\boldsymbol{W'}$ is a $N \times V$ matrix referring to the weights between the hidden and output layer. The learning weights are obtained via backpropagation, which is a state-of-the-art technique for machine learning in multilayer neural networks [58]. After training, matrix $\boldsymbol{W}$ captures the semantic information of each word, where the i-th row of the matrix expresses the vector representation of i-th word in the vocabulary V. Matrix $\boldsymbol{W'}$ gives information about how the embedded word $\boldsymbol{h}$ relates to its context vector $\boldsymbol{y}$

This can also be extended to higher window size k of surrounding context words as shown in Figure 3.5, where the input layer is the average vector of one-hot vectors of surrounding words and the equations are formulated as follows:

$$\boldsymbol{h} = \frac{1}{C}\boldsymbol{W}^{T}(\boldsymbol{x_1} + \boldsymbol{x_2} + \cdots + \boldsymbol{x_k})$$

$$\boldsymbol{y} = \boldsymbol{W'}^{T}\boldsymbol{h}$$

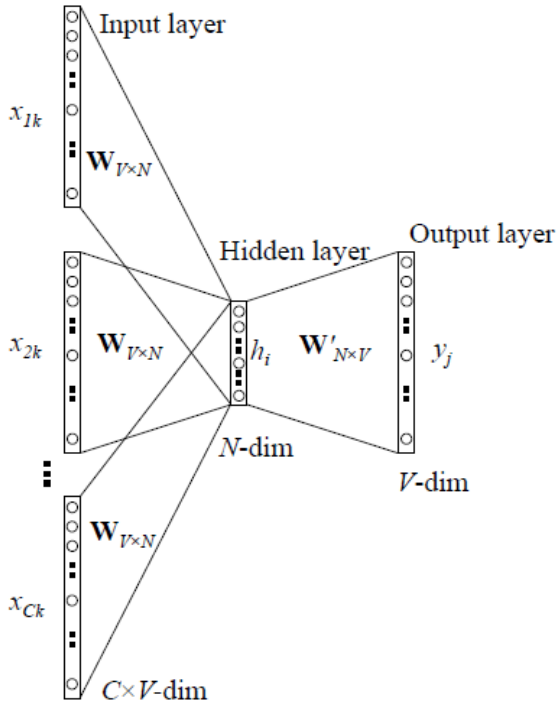

Figure 3.5. Word2Vec Continuous Bag-of-Words (CBOW) model [57]. Input layer refers to one-hot vectors that are aggregated indicating each word's unique ID. It is a shallow neural net-work that is trained to predict the word given a context within a sliding window.

On the other hand, Skip-Gram has the opposite approach of CBOW. The neural network is trained to predict the surrounding context (for a fixed-sliding window) of a current word. Therefore, the one-hot vector of current word comprises the model's input and the output corresponds to the context words. The general idea of this framework is presented in Figure 3.6.

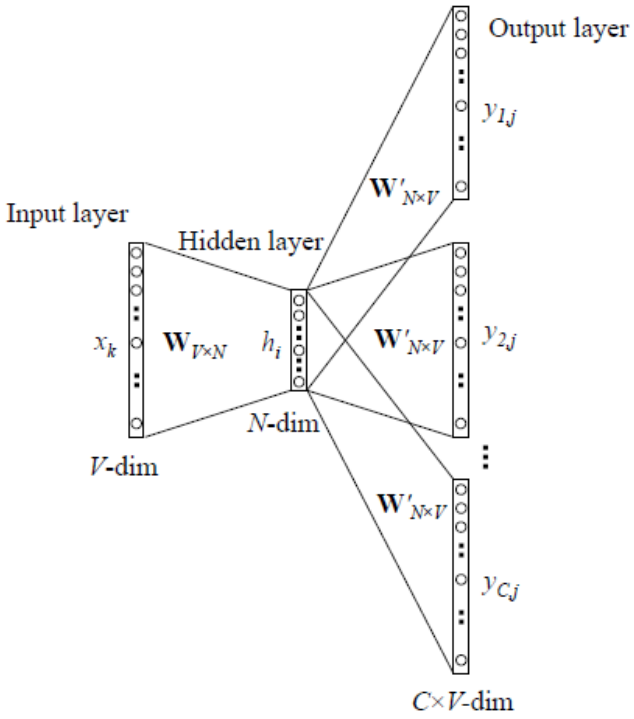

Figure 3.6. Word2Vec Skip-Gram model [57]. Input layer refers to a one-hot vector indicating word's unique ID and the output layer denotes the context (one-hot vector words) within a sliding window, that the shallow neural network is trained to predict.

The word embeddings are learned using the training set as vocabulary and the window size is specified by the user. In order to extend word embeddings to document embeddings, two approaches can be implemented: One is to aggregate the word vectors in a document by either calculating the average word vector or summing the vectors altogether. The second approach is to merge the word vectors forming a matrix and feeding them to a deep neural network architecture like Long-Short-Term Memory network or a Convolutional Neural Network, which are complex models that are very efficient on sequential data.

Apart from training word embeddings in the given dataset, pretrained word embeddings can also be utilized, which capture the concept of transfer learning. Transfer

learning is the notion of transferring knowledge from an already trained machine learning model to another task. It is usually implemented on deep neural networks where the lower-level layers capture the most general concepts. For example, a machine learning model that is trained to detect cars, can transfer its knowledge to a truck classification task by using it as lower-level information and then adding a higher-level layer that distinguishes a truck. This technique is powerful when training small datasets as it reduces the number of learning parameters and prevents overfitting issues. Furthermore, it improves the training performance, because it speeds up the training phase without requiring heavy computational resources. In text classification tasks, pretrained word embeddings form a language model that can also generalize well. They are trained in large corpus with billion words and can be used as a vector representation of text that captures general language aspects.

A widely known pretrained embedding model was trained on Google News corpus[8], which contains 3 billion running words. This model was implemented on Word2Vec framework and includes word vectors with 300 hidden units for a vocabulary of 3 million words.

3.4.4 Doc2Vec

An extension of Word2Vec is Doc2Vec [59]. Doc2Vec is an unsupervised framework that extracts paragraph or document vectors from a collection of documents. It uses the same idea as Word2Vec, which is trained to predict the current word from a context (CBOW approach) or to predict the surrounding context of a given word. However, it includes a paragraph ID one-hot vector that captures the semantic representation of the document. Likewise, there are two approaches for learning the document embeddings: Paragraph Vector – Distributed Memory (PV-DM) and Paragraph Vector – Distributed Bag of Words (PV-DBOW).

PV-DBOW is like CBOW in Word2Vec but adds a document ID one-hot vector as input together with the word's surrounding context one-hot vectors. The document vector and word vectors are either averaged or concatenated to predict the upcoming word of a surrounding context. An example of this method is presented in Figure 3.7. After

[8] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM

obtaining the learning weights, the weight matrix between the input and hidden layer encodes the document embeddings.

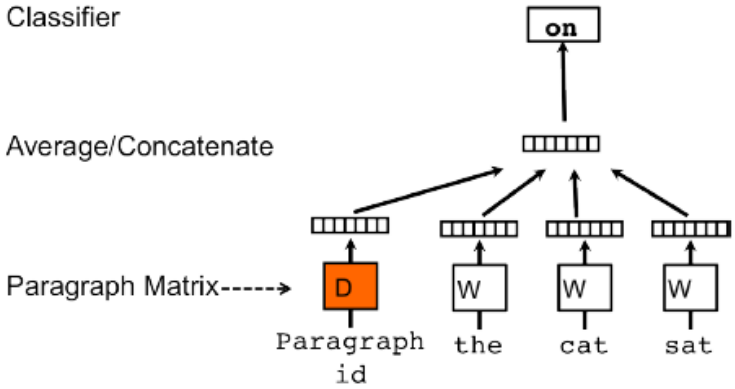

Figure 3.7. Doc2Vec PV-DM approach [59]. It is a shallow neural network that is trained to predict a one-hot vector word given a context of one-hot vectors which includes a one-hot vector representing the document's unique ID.

PV-DBOW captures the same concept with Skip-Gram modeling in Word2Vec. Contrary to PV-DM, it ignores the one-hot word vectors within a window but is trained to predict a context of words that occur in the document. More specifically, given a one-hot document vector, a text window within document is randomly sampled to depict the output. A typical architecture of PV-DBOW model is demonstrated in Figure 3.8.

In order to obtain document embeddings, after selecting the number of units in the hidden layer and the learning rate, we train the Doc2Vec model multiple times, because as mentioned, it uses randomized techniques for learning document embeddings. Furthermore, at each training iteration the learning rate is slightly decreased and the collection of documents is randomly sampled to avoid biasness in the order of documents.

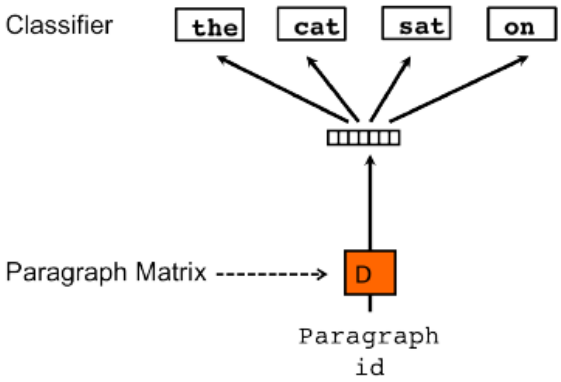

Figure 3.8. Doc2Vec PV-DBOW approach [59]. It is a shallow neural network that is trained to predict a context of one-hot vectors given as input a one-hot vector representing the document's unique ID.

3.4.5 Google universal sentence embeddings

Another approach of transfer learning is by utilizing pretrained sentence embeddings. The authors in [60] have trained on a large collection of texts a model that converts text into high dimensional vectors (512 dimensionality). Text corresponds to English phrases, sentences or paragraphs. The pretrained model is provided in Tensorflow Hub[9] and in our problem, is used to obtain document vectors and subsequently a semantic representation of documents for text classification.

3.5 Machine learning models

The examined machine learning models implemented in this thesis are Logistic regression, Neural Networks and deep learning models such as Long short-term memory networks and Convolutional neural networks that according to [61] achieve very good results.

3.5.1 Logistic regression

Logistic Regression is a state-of-the-art algorithm for classification tasks [61]. It is a regression-based model, which is a method that learns weights for each feature. In other words, it captures the importance of each attribute to the classification task. Instead of modeling the outcome y promptly, it models the probability that y belongs to a specific label. A linear regression model is formulated as follows:

$$f(X_i) = w_0 + w_1 x_{i1} + \cdots + w_m x_{im} = \boldsymbol{W}\boldsymbol{x}$$

where $f(X_i)$ is the prediction of the regression model, x_{ij} are the features of document i and w_i is the weight of each feature. The probability that an instance is classified as real or not real is calculated as follows:

$$P(Y = y_j|X) = \frac{e^{y_j \cdot f(X_i)}}{1 + e^{y_j \cdot f(X_i)}} = \frac{1}{1 + e^{-y_j \cdot f(X_i)}}$$

[9] https://tfhub.dev/google/universal-sentence-encoder-large/3

where y_j is the label that can be 1 (real) or -1 (not real). This formula is called sigmoid function and its graph representation is shown in Figure 3.9. It is indicated that sigmoid function takes values in the range [0,1] and therefore can form probability models.

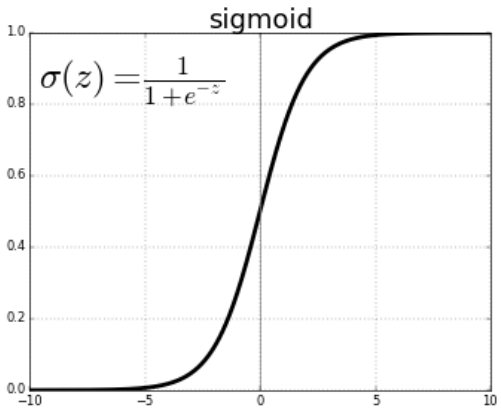

Figure 3.9. Sigmoid function. It maps a real number in the range of [0,1] indicating probability distribution.

The method used to fit the model and calculate the regression coefficients, is called maximum likelihood. The goal of this method is to maximize the joint probability of all instances being correctly classified and if we assume that the variables are independent, the optimization problem is formulated as:

$$\boldsymbol{w} = argmax_{\boldsymbol{w}} \left[log \left(\sum_{i}^{n} \frac{1}{1 + e^{-y_j \cdot f(X_i)}} \right) \right]$$

or

$$\boldsymbol{w} = argmin_{\boldsymbol{w}} \left[log \left(\sum_{i}^{n} (1 + e^{-y_j \cdot f(X_i)}) \right) \right]$$

After finding the regression coefficients, the probability $P(X) = P(Y = 1|X)$ is calculated and by using the 0.5 threshold the instance will be classified as real if $P(X) > 0.5$ and as not real if $P(X) < 0.5$. The algorithm was implemented, using the LogisticRegression function provided in Python sklearn library.

3.5.2 Neural networks

Neural networks originate several decades ago. According to the author in [62], a gentle definition of a neural network, which is also called Artificial Neural Network (ANN), is: "Neural network is a computing system made up of a number of simple, highly

interconnected processing elements, which process information by their dynamic state response to external inputs." It is based on human brain concept, where for each observation specific neurons are activated that lead to a final decision. A typical neural network architecture is presented in Figure 3.10. Neural networks are modeled with layers. The first layer is the input layer, which denotes the attributes (units) of each instance. The input layer is mapped to a hidden layer space, which is also mapped to another hidden layer until it reaches the output layer, which denotes the prediction of the neural network model. The first layers of a neural network capture lower-level concepts, while the last layers capture notions that are mostly relevant to the classification task. For example, in text classification the first layers provide language modeling information and the last layers capture concepts regarding the prediction (real news or not).

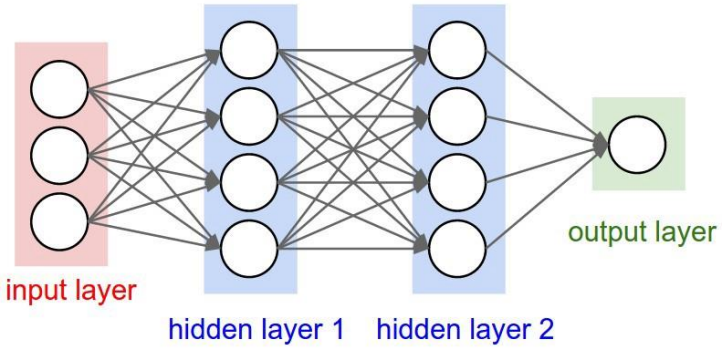

Figure 3.10. Neural network architecture with 2 hidden layers. Each layer is fully connected to the next one.

Each neuron in a neural network model has the architecture presented in Figure 3.11.a. The mathematical formula is:

$$u = b + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

$$y = f(u)$$

where w_i refer to the weights and x_i related features. $f(u)$ is the activation function, which indicates the degree of activation of a neuron or in other words, how much information pass through the specific neuron. It can be any continuous function but for the output layer sigmoid function is preferred because it models probabilities for the outcomes.

The training of a neural network is accomplished with the backpropagation algorithm. Backpropagation uses stochastic gradient descent with the purpose to minimize the loss function of the outcomes and to converge the values of the weights. In addition to that,

sigmoid function should not be used as an activation for the hidden layer units, because its derivative has small values which decelerates the convergence of the learning weights (vanishing gradient problem). In this case, Rectified Linear Unit (ReLU) activation function is preferred which is linear in the positive area and rectifies negative values (Figure 3.11.b).

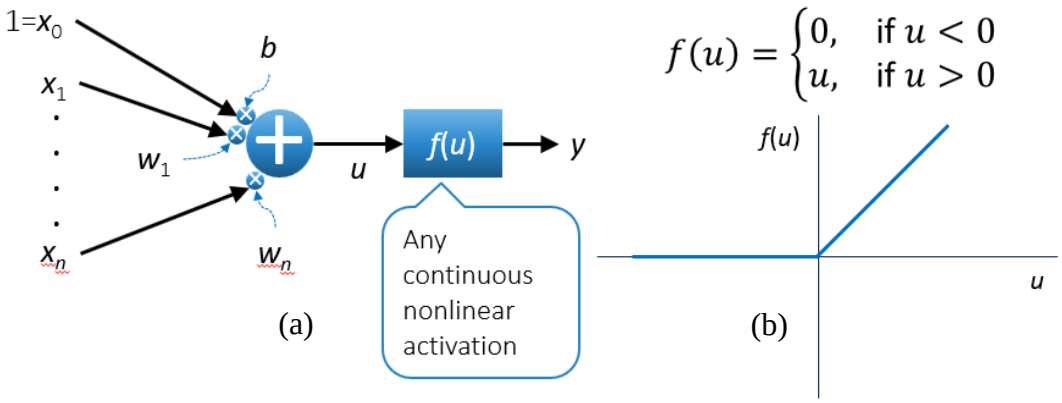

Figure 3.11. (a) A single neuron architecture: Each neuron x_i is multiplied with its weight w_i and the sum of them u is passed through an activation function $f(u)$ which restricts the outcome in a specific range, (b) ReLU activation function: It keeps only positive values and is often used in the hidden layers.

The neural network framework was implemented on Keras[10], which is a high-level neural networks API written in Python language. Keras includes the Sequential model which is a user-friendly function that allows constructing and combining multiple neural network architectures.

3.5.3 Long Short-Term Memory networks

The authors in [63] propose a novel deep neural model, which is called Long Short-Term Memory (LSTM) network. LSTM works very well for sequential data, such as time-series data and text data which is a sequence of words. It is a complex form of Recurrent Neural Networks (RNN), which are neural networks with loops, allowing information to persist. A typical RNN architecture is demonstrated in Figure 3.12, where X_t is a vector. If we unfold an RNN it can be observed that it is similar to a multi-layer neural network, where each data in the sequence represents a hidden layer with the difference that each layer has additional input. The activation of each layer is established by the tanh function, which

[10] https://keras.io/

returns values in the range (-1,1). In text classification, it is based on predicting the next word given a previous fixed-size context.

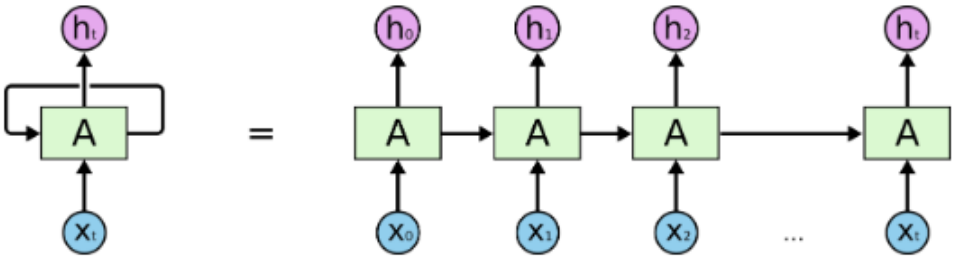

Figure 3.12. A typical RNN architecture and the unfolded version[11]. A one-hot vector word $\boldsymbol{X_i}$ refers to a word that appears before another word $\boldsymbol{X_t}$.

Nevertheless, in many cases we also want information from a word or phrase that appears earlier in the document, to be able to predict the upcoming word. In other words, we also need "long-term" dependencies. For example, in the beginning of a paragraph, the phrase "I grew up in Greece" could occur and in the end of the paragraph the phrase "I speak fluent Greek". In order to persist this information, a typical RNN requires a large context to consider. However, larger context means greater number of layers, which could lead to the vanishing gradient problem.

LSTM can deal with this problem by including gates that decide how much information will be maintained from the previous instances and the degree of updating the information for the new instance. An unfolded version of an LSTM network is depicted in Figure 3.13.

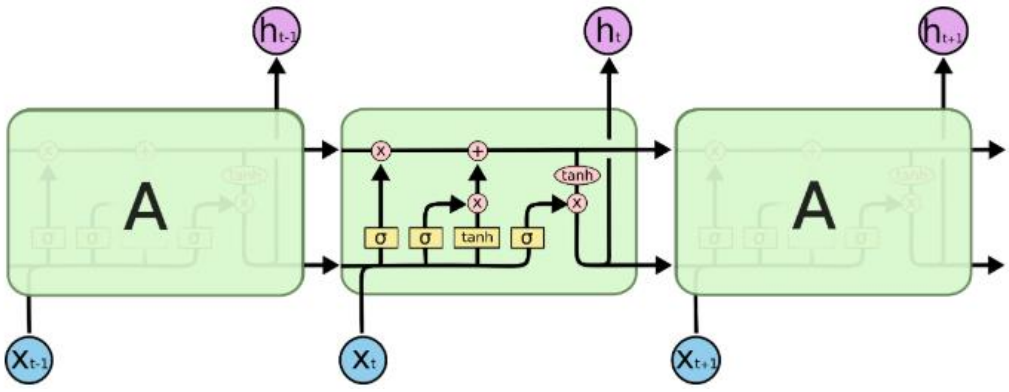

Figure 3.13. An unfolded LSTM network. It contains two outputs: one regarding the information that will persist for the next word and one for the result of the specific word[9]

[11] http://colah.github.io/posts/2015-08-Understanding-LSTMs

The LSTM framework was implemented on the Keras Python library. It requires as input for each feature a fixed-size 2-dimensional array, which consists of the word embeddings within the document. Because Keras requires numerical values as input, each word in the vocabulary of the word embedding model is mapped to a unique integer and each integer is mapped to the embedding vector space. However, each document has different number of words. To avoid this problem, we utilize a technique called padding. We initially specify the maximum size of documents and each document (previously converted to a sequence of integers) that has smaller length will be padded with zeros to persist the size of the feature array and map these values to a zero vector in the embedding space. We also included a fully-connected (Dense) layer to calculate the probability distribution of the classification task. The general structure of the implemented model is depicted in Figure 3.14.

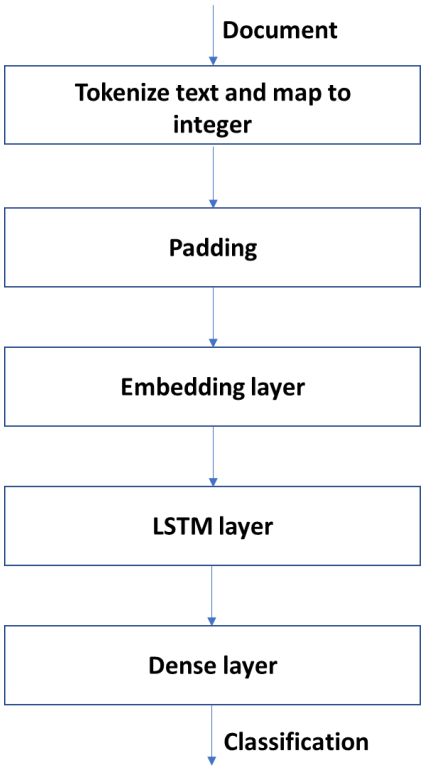

Figure 3.14. The implemented LSTM model architecture. Document is tokenized and each word is mapped to a unique integer. Each document is converted in a fixed with missing words covered with zeros. Then each word is converted into vector using Word2Vec models and they are fed as input to a LSTM layer. The outcomes of LSTS are fed in a fully-connected (Dense) layer to create probability distribution of the predicted classes.

3.5.4 Convolutional neural networks

Convolutional Neural Networks (CNNs) are a widely known method for pattern recognition tasks, such as computer vision [64] and text classification [65]. The general idea is that instead of having fully connected layers like a simple neural network, the connections happen in certain regions within a sliding window. Furthermore, the connected neurons have the same weights for all sliding windows. A typical example of a CNN architecture is presented in Figure 3.15. The input is a sequence of words mapped in the embedding space, thus each word is a vector. The sliding window size is 2, which means that the model considers bigrams of words. Therefore, each convolutional neuron h_i (A) that reflects to a specific bigram, is mathematically formulated as:

$$h_i = f(w_0 x_i + w_1 x_{i+1} - b)$$

where w_0, w_1 are the learning weights for the convolutional layer and f is the neuron activation function (usually ReLU function is preferred to avoid the vanishing gradient problem). Subsequently, the convolutional layer is connected to a max pooling layer, which is a layer that selects the most important bigrams in the text similarly to image classification tasks, where we are not interested in the value of specific pixels, but in the maximum value of a group of pixels. Finally, a fully connected layer F is included to calculate the probability of the classification task.

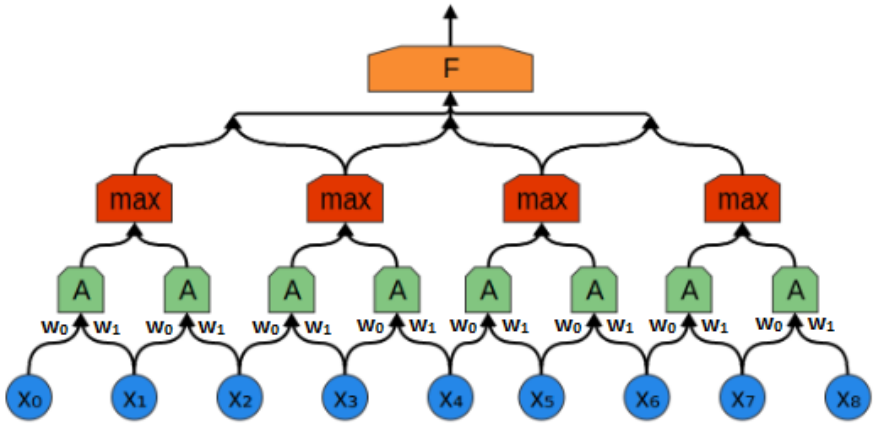

Figure 3.15. A typical CNN architecture considering bigrams and including max pooling and fully connected layers[12].

[12] http://colah.github.io/posts/2014-07-Conv-Nets-Modular

Convolutional neural networks were implemented on Keras Python. The preprocessing of the input data is implemented the same way as for LSTM models by tokenizing, padding and converting to word vectors.

3.6 Model explanations

In order to explain the decisions of the machine learning models, we utilized LIME and EDC explanation algorithms.

3.6.1 LIME

LIME algorithm is provided in a Python library that is available on GitHub[13]. This library includes a class called LimeTextExplainer that extracts model explanations for text classification tasks. First, the text is tokenized and neighborhood data is constructed by randomly deleting features from the instance. Then, the algorithm learns locally weighted linear models on this data to explain each of the classes in an interpretable way. The explanations are presented as a feature importance output, where the top-k important words for the classification are extracted with their contribution degree to classification. LimeTextExplainer uses by default exponential kernel that takes as input the Euclidean distances and kernel width. It includes a function that explains the classifier's decision (explain_instance). This function requires as input the document that will be explained, a probability function, the number of words that will be selected as explanations and the number of perturbation samples that will be created which form the neighborhood of the text instance. The probability function returns the black box model's prediction probabilities of a set of documents.

3.6.2 EDC

As previously mentioned, in EDC an explanation is defined as the minimal set of words that if is removed from the document the black box model predicts a different class. EDC was implemented on Python using the pseudocode provided in [41]. More specifically,

[13] https://github.com/marcotcr/lime

we created a function that receives as input parameters: the instance to be explained, a "pipeline" function that receives a set of documents and returns the prediction probability of the text classifier for each sample. The source code is written in Python and is demonstrated in Appendix.

4 Experimental results

In this chapter we describe the experiments conducted for our proposed disinformation detection methods. We compare different feature extraction techniques (TF-IDF, graph of words, Doc2vec, Word2Vec, google universal sentence embeddings), that transform each document in vector representation, in combination with different machine learning models (Logistic regression, neural networks, LSTM and CNN) as described in chapter 3. We also extracted model explanations for the decisions of the machine learning models by utilizing LIME and EDC frameworks. However, LIME uses randomized approaches to create the neighborhood dataset from the text instance, which makes it difficult to robustly compare our models. To overcome this problem, we used a specific random seed which is a number that initializes a pseudorandom number generator and subsequently the constructed dataset will be the same for each experiment. Furthermore, we selected the size of the neighborhood data to be 20000 and that LIME returns the top 30 words that contribute to the decision of the black box model.

We used the same data preprocessing approach for our experiments. First, we randomly split the dataset into training set (80% 5068 instances) and test set (20% 1267 instances) and merge the title with the main body of each article. Then, we perform text preprocessing as described in section 3.2. The explanations algorithms were tested on a real news article and on a non-real news article. After exploring the dataset, it is observed that fake news pieces have less document length that real news. Specifically, fake news articles have on average 389 words per article while real news have 487 words.

This chapter demonstrates the conducted experiments and compares the accuracy and prediction explanations of the black box models for fake news detection.

4.1 Fake news classification using TF-IDF

As previously mentioned, TF-IDF was implemented by using the TfidfVectorizer object provided by sklearn. We considered the top 100000 words with the highest term frequency in the training corpus to reduce the dimensionality of our problem. We conducted experiments using different combinations of n-grams as follows:

(a) unigrams
(b) unigrams and bigrams
(c) unigrams, bigrams and trigrams

Moreover, we used Truncated Singular Value Decomposition (TSVD) to further reduce the dimensionality of our problem and compared different dimensionalities.

In addition to that, we examined neural networks (NN) with various layer architectures with one or two hidden layers implemented on Keras library with the combination of all n-grams as features. TfidfVectorizer returns a sparse array of the TF-IDF values of each term in a document, which reduces the memory usage. Nevertheless, Keras requires dense array as input which causes inefficiency. To avoid this problem, Keras provides a function (fit_generator) that fits the machine learning model to the training data by using an iterator that yields batches of data. We also use a technique called early stopping, which is a regularization method in neural networks that monitors the results of the classifier in a validation set for each training epoch and interrupts the training process when the accuracy in the training data increases but not in the validation data. In other words, this method learns in which epoch to stop training in order to make the model more generalizable. The results of these experiments are presented in Tables 4.1, 4.2, which show the accuracy of Logistic Regression and neural network classifiers respectively. Table 4.1 presents the results of various TF-IDF configurations with Logistic regression classifier. Table 4.2 demonstrates the results of TF-IDF with n-grams (unigrams + bigrams + trigrams), using various neural network architectures with or without TSVD.

Table 4.1. Accuracy results of TF-IDF with n-grams and Logistic Regression classifier

TF-IDF with n-grams	**Accuracy**
Unigrams	0.9361
Unigrams and bigrams	0.94
Unigrams, bigrams and trigrams	**0.94**
All words in the corpus	0.9353
TSVD with 256 dimensions	0.9227
TSVD with 512 dimensions	**0.9361**
TSVD 1024 dimensions	0.9321

Table 4.2. Accuracy results of TF-IDF with n-grams (unigrams + bigrams + trigrams) and neural network classifier

TF-IDF with n-grams	**Accuracy**
one hidden layer with 128 units	0.9448
one hidden layer with 256 units	0.9479
one hidden layer with 512 units	0.9463
two hidden layers 32x32	0.9471
two hidden layers 64x32	0.9495
two hidden layers 128x32	**0.9519**
TSVD and one hidden layer with 128 units	0.9353
TSVD and one hidden layer with 256 units	0.9392
TSVD and one hidden layer with 512 units	0.9376
TSVD and two hidden layers 32x32	0.9345
TSVD and two hidden layers 64x32	0.94
TSVD and two hidden layers 128x32	**0.9432**

The results indicate that the combination of unigrams, bigrams and trigrams achieved the highest accuracy (0.94) on a Logistic Regression Classifier, while TSVD does not improve the accuracy of the model (0.9361). Furthermore, it is observed that considering all n-grams in the corpus reduces the accuracy of the results (0.9353) which is caused by high dimensionality. Neural networks were implemented on n-gram range (1,3). Extending Logistic regression to neural networks boosts the result, where a neural network with two hidden layers (128 and 32 units respectively) outperformed the rest of the methods with accuracy 0.9519.

Subsequently, we implemented the skipgrams approach on different combinations of number of skips and n-grams. NLTK provides a function which converts a document into skipgram features. We created a wrapper function as analyzer parameter in TfidfVectorizer, which uses NLTK to create skipgrams for combination of unigrams, bigrams and trigrams. The results of our experiments are demonstrated in Tables 4.3, 4.4. Table 4.3 shows the prediction accuracy of different combinations of skip steps and n-grams using the Logistic Regression classifier. Table 4.4 demonstrates the results obtained using TF-IDF with 2-skip unigrams, bigrams and trigrams fitted in various neural network architectures.

Table 4.3. Accuracy results of TF-IDF with n-grams and Logistic Regression

TF-IDF with skipgrams	**Accuracy**
2-skip unigrams and bigrams	0.94
2-skip unigrams, bigrams and trigrams	**0.94**
3-skip unigrams and bigrams	0.9432
3-skip unigrams, bigrams and trigrams	**0.944**
4-skip unigrams and bigrams	0.9376
5-skip unigrams and bigrams	0.9384
6-skip unigrams and bigrams	0.9376
TSVD 2-skip unigrams and bigrams	0.9345
TSVD 2-skip unigrams and bigrams	0.9385
TSVD 3-skip unigrams and bigrams	0.9361
TSVD 3-skip unigrams and bigrams	0.9353

Table 4.4. Accuracy results of TF-IDF with 2-skip unigrams, bigrams and trigrams and neural network classifier

TF-IDF with skipgrams	**Accuracy**
one hidden layer with 128 units	0.9448
one hidden layer with 256 units	0.9455
one hidden layer with 512 units	0.9471
two hidden layers 32x32	**0.9495**
two hidden layers 64x32	0.9432
two hidden layers 128x32	0.9455

Next, we show the results of two explanations algorithms, namely LIME and EDC, over the best results acquired from the previous step.

4.1.1 Explanations using LIME

Figures 4.1, 4.2 present the LIME explanations of best TF-IDF models with n-grams using Logistic regression and neural networks respectively. Figure 4.3 depicts the explanations of TF-IDF with skipgrams using Logistic regression. Words with positive weight (green) indicate contribution to real news classification and with negative weight contribution to fake news. It can be seen that real news contains words that refer to third person (said, told, say) and also mostly formal words. On the contrary, fake news include more

contradicting words like “lie”, “illegal”, “lies” and “breaking” as well as more informal words, such as “knew” and “know”. Furthermore, it is indicated that even though neural network classifier achieves highest accuracy, explanations of Logistic regression are clearer and less biased.

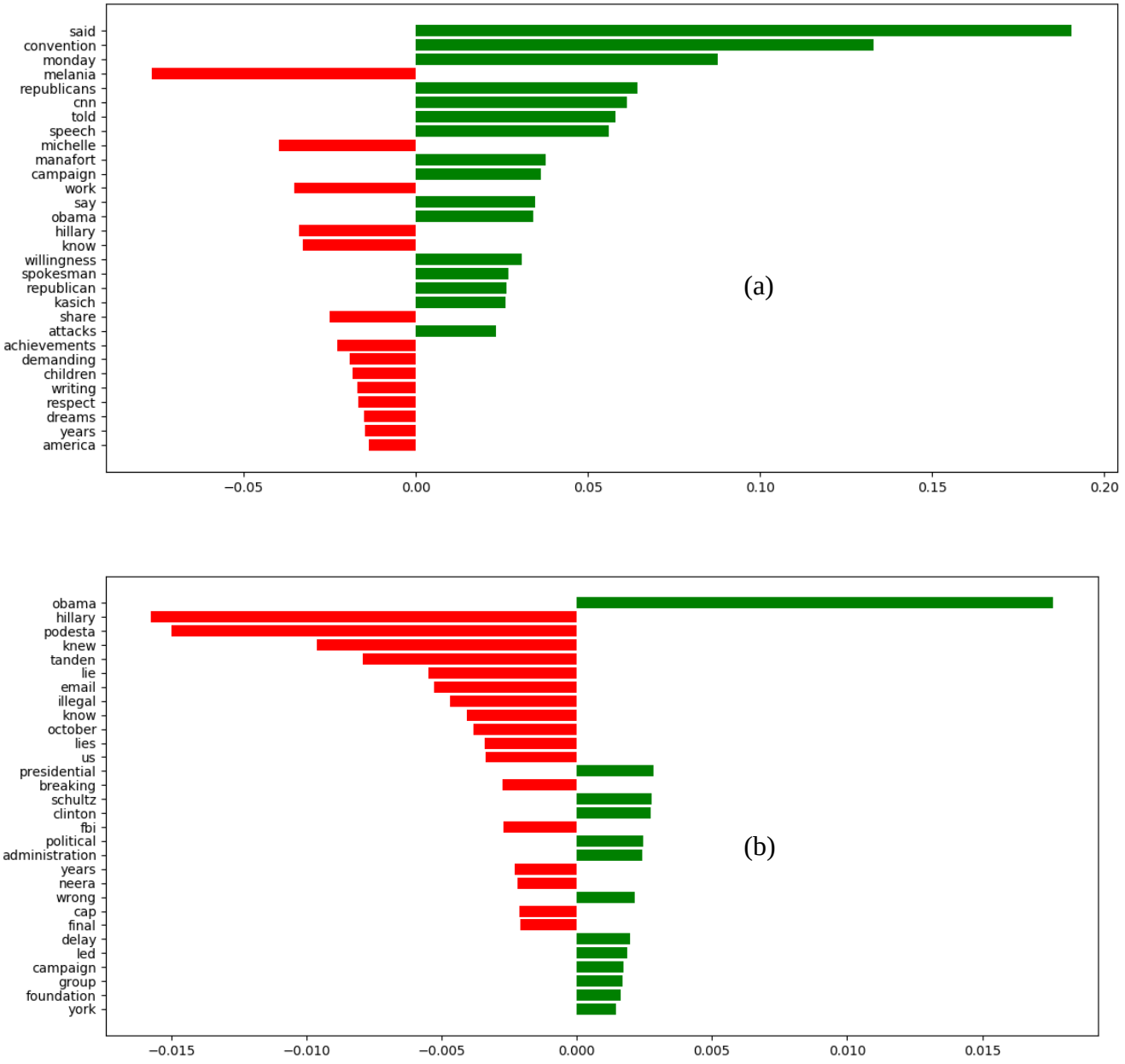

Figure 4.1. LIME explanations for TF-IDF with unigrams, bigrams and trigrams using Logistic Regression classifier. Each word has a weight denoting its contribution to the classification. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a). Real news article, (b) Non-real news article

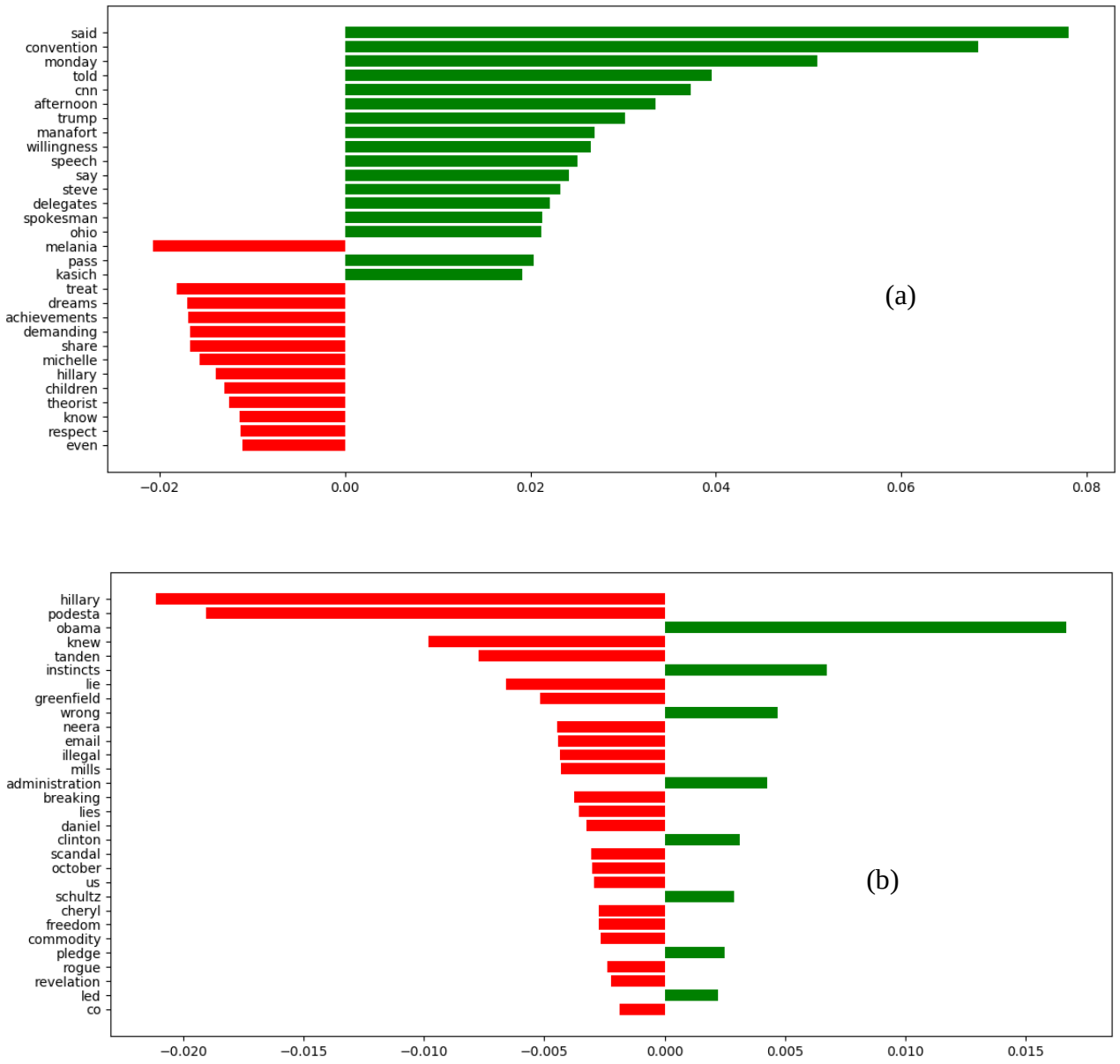

Figure 4.2. LIME explanations for TF-IDF with unigrams, bigrams and trigrams using neural network classifier with 2 hidden layers (128 and 32 units respectively). Each word has a weight denoting its contribution to the classification. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Non-real news article

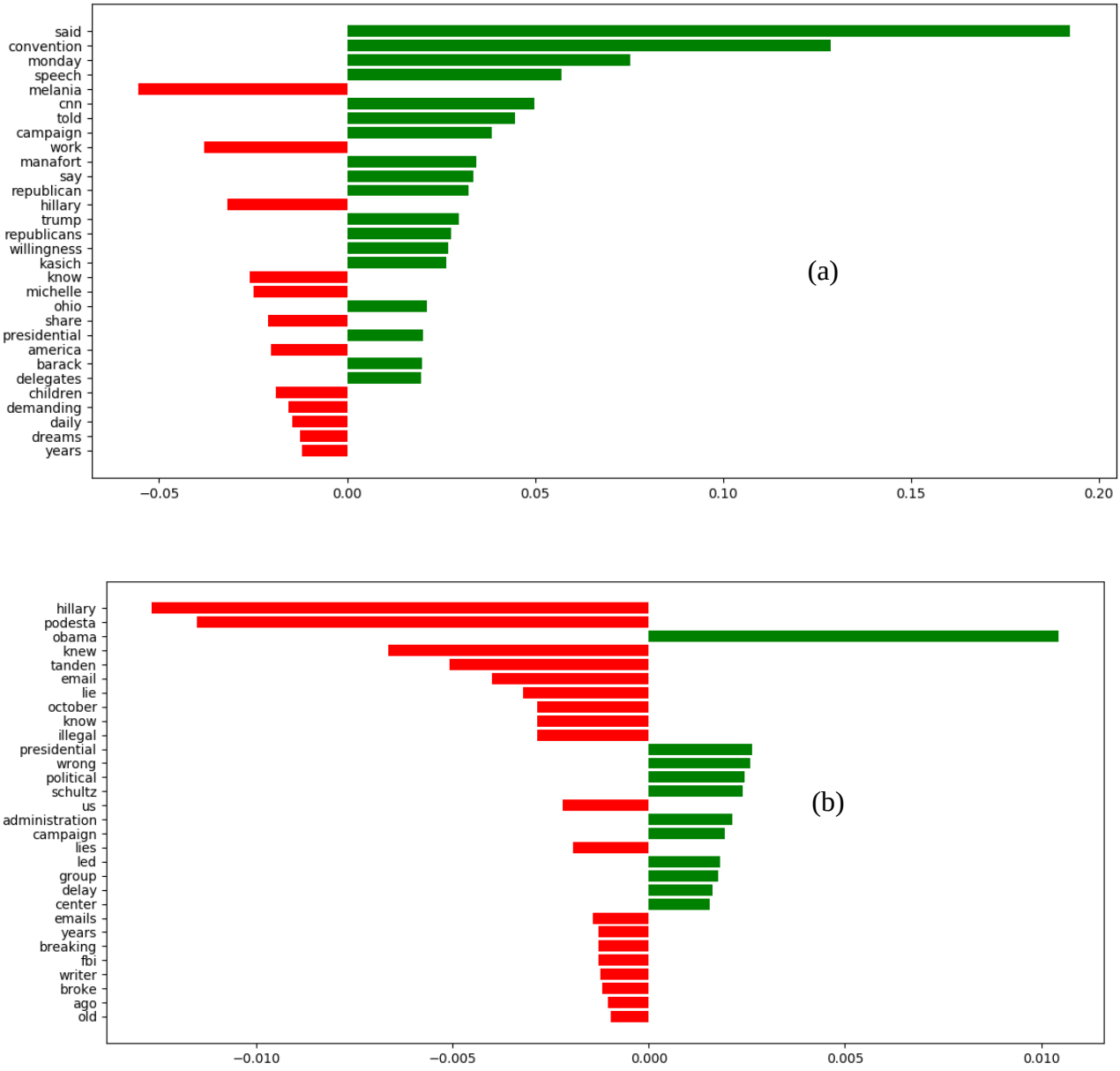

Figure 4.3. LIME explanations for TF-IDF with 3-skip unigrams, bigrams and trigrams using Logistic Regression. Each word has a weight denoting its contribution to the classification. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Non-real news article

4.1.2 Explanations using EDC

EDC produces explanations as sets of words that, if removed from the documents, would change the classification results. Those sets are considered as explanations for each class (real vs fake). Table 4.5, 4.6 show the word sets produced by EDC for three different experimental setups, namely TF-IDF with n-grams, TF-IDF TSV with n-grams and TF-IDF with skipgrams using Logistic regression and neural networks, respectively. It is observed that the model explanations that EDC returns, are similar to LIME explanations

with main difference that EDC denotes only the words related to the predicted class. Comparing the explanation results in these tables it appears that Logistic Regression presents more transparent explanations illustrating reliable or unreliable news characteristics.

Table 4.5. Model explanations of different TF-IDF implementations with Logistic Regression using EDC

Model	EDC REAL	EDC FAKE
TF-IDF n-grams	*said, convention, monday, cnn, republicans, speech*	*hillary, podesta, knew, tanden, email, illegal, lie, us, emails, know, october, exposing, ago, cover, leaked, neera, jackson, eight, fake, clean, scandal*
TF-IDF TSVD n-grams	*convention, said, delegates, monday, told, cnn, kasich, want*	*hillary, podesta, october, jackson, ap, greenfield, know, take, breaking, rituals, project, associate, cheryl, email, spin, rogue, exposing, warned, clean, government*
TF-IDF skipgrams	*said, convention, monday, cnn, speech, trump, told, republican, manafort, campaign, say*	*hillary, podesta, knew, tanden, emails, greenfield, mills, lie, email, illegal, freedom, us, eight, neera, know, breaking, october, cheryl, state, clean, gov*

Table 4.6. Model explanations of different TF-IDF implementations with two hidden layer neural network (32x32) using EDC

Model	EDC REAL	EDC FAKE
TF-IDF n-grams	*said, convention, told, monday, paul, steve, speech, spokesman, ohio, afternoon, say, willingness, kasich, delegates*	*hillary, podesta, knew, tanden, emails, greenfield, mills, lie, email, illegal, freedom, us, eight, neera, know, breaking, october, cheryl, state, clean, gov*
TF-IDF skipgrams	*convention, said, monday, campaign, cnn, trump, ohio, say, benghazi, erupted, made, success, steve, back, comments, focused, afternoon, eight, speech, manafort*	*hillary, podesta, tanden, knew, us, greenfield, email, cheryl, lie, know, headed, progress, co, illegal, rogue, final, time, anything, charge, partial*

4.2 Fake news classification using graph of words

Next, we utilized the graph of words approach and calculated the TW-IDF, TW-ICW and TW-ICW-LW values for each term in a document. We compared various parameters as follows:

(a) window size = 2
(b) window size = 3
(c) window size = 4
(d) TSVD
(e) different neural network architectures

First, we created graphs corresponding to the penalization factors (IDF, ICW, LW) and defined a wrapper function that constructs the graph of words for each new document with purpose to compute the TW-IDF, TW-ICW and TW-ICW-LW values. The experimental results are presented in Tables 4.7, 4.8 using Logistic Regression and neural networks respectively.

Table 4.7. TW-IDF with different window sizes and TSVD using Logistic Regression

TW-IDF	**Accuracy**
window size = 2	**0.9282**
window size = 3	0.9258
window size = 4	**0.9282**
window size = 2 and TSVD	0.9227
window size = 3 and TSVD	0.9155
window size = 4 and TSVD	0.9132

The experiments indicate that using a sliding window of size 2 performs better (0.9282 with Logistic Regression) and the highest accuracy is obtained by a two-hidden-layer neural network with 64 and 32 neurons respectively (0.9353 accuracy). The explanation results using LIME and EDC over the best models, are presented in the following sections.

Table 4.8. TW-IDF with window size = 2 and TSVD using various neural network architectures

TW-IDF	**Accuracy**
one hidden layer with 128 units	**0.9337**
one hidden layer with 256 units	0.9282
one hidden layer with 512 units	0.9305
two hidden layers 32x32	0.9324
two hidden layers 64x32	**0.9353**
two hidden layers 128x32	0.929
TSVD and one hidden layer with 128	0.9242
TSVD and one hidden layer with 256	0.9266
TSVD and one hidden layer with 512	0.9258
TSVD and two hidden layers 32x32	0.9274
TSVD and two hidden layers 64x32	0.9282
TSVD and two hidden layers 128x32	0.9305

4.2.1 Explanations using LIME

Figure 4.4 denotes the explanations of TW-IDF model over the best parameters using Logistic Regression and neural network architecture. It is observed that graph-of-words models perform similar results to bag-of-word approaches (TF-IDF) when using degree centrality as term weighting metric.

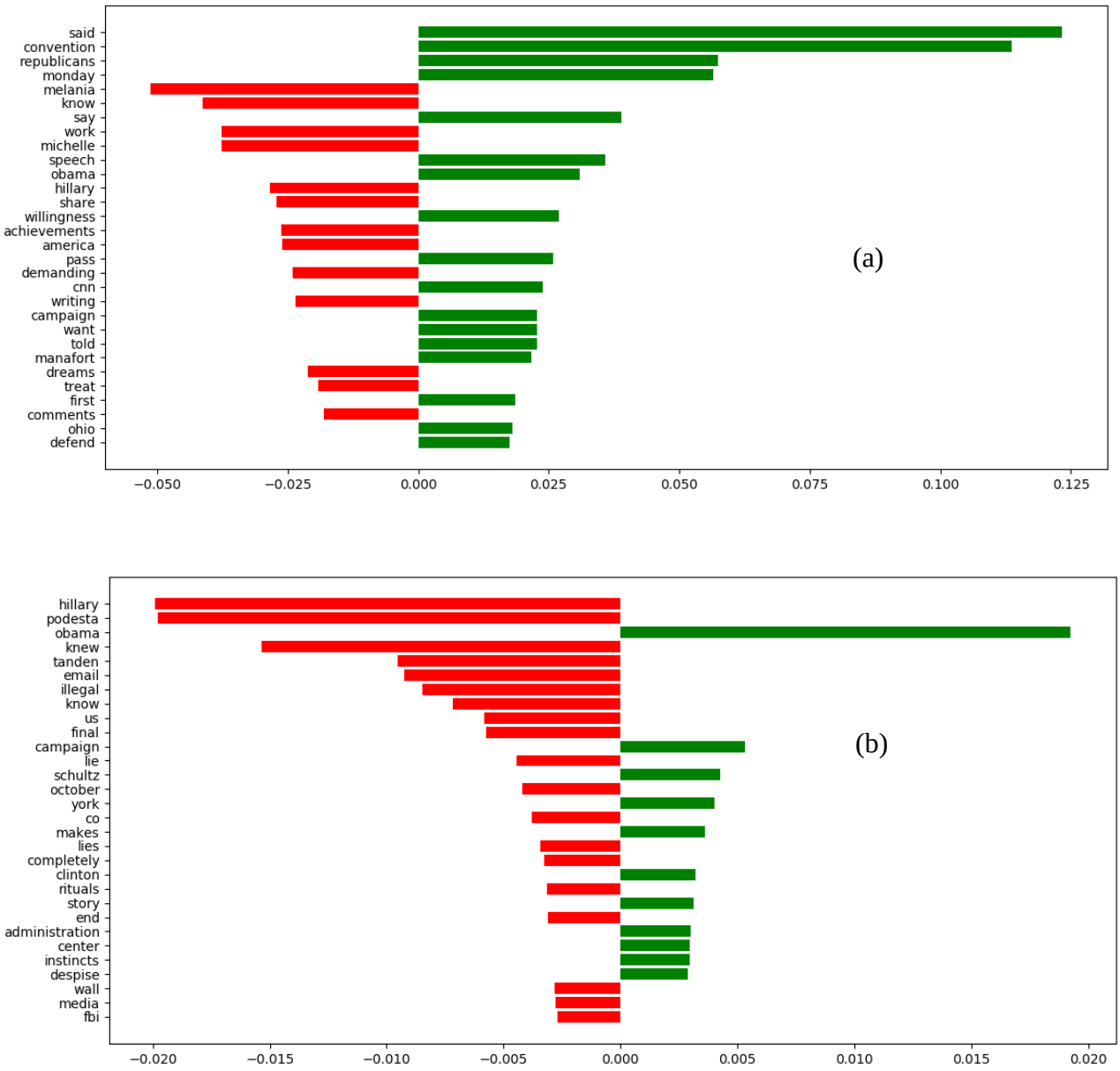

Figure 4.4. LIME explanations for TW-IDF model with window size = 2 using Logistic Regression. Each word has a weight denoting its contribution to the classification. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Non-real news article.

4.2.2 Explanations using EDC

Table 4.9 shows the explanations for the specific predictions that EDC returns for each model over the best parameters, using Logistic Regression and neural network architecture. The explanation results suggest that using TW-IDF with Logistic Regression is less biased than the rest of the graph-of-word models, containing explanations for fake news:

knew, illegal, lie, know, exposing, ago, leaked, fake, scandal which are more informal and with greater polarity.

Table 4.9. Model explanations of TW-IDF, TW-ICW and TW-ICW-LW implementations with Logistic Regression using EDC

Model	EDC REAL	EDC FAKE
TW-IDF	*said, convention, monday, cnn, republicans, speech*	*hillary, podesta, knew, tanden, email, illegal, lie, us, emails, know, october, exposing, ago, cover, leaked, neera, jackson, eight, fake, clean, scandal*
TW-ICW	*convention, said, delegates, monday, told, cnn, kasich, want*	*hillary, podesta, october, jackson, ap, greenfield, know, take, breaking, rituals, project, associate, cheryl, email, spin, rogue, exposing, warned, clean, government*
TW-ICW-LW	*said, convention, monday, cnn, speech, trump, told, republican, manafort, campaign, say*	*hillary, podesta, knew, tanden, emails, greenfield, mills, lie, email, illegal, freedom, us, eight, neera, know, breaking, october, cheryl, state, clean, gov*

4.3 Fake news classification using Word2Vec

We trained word embedding on the training corpus using Continuous Bag of Words (CBOW) and Skipgram modelling. On top of that, we utilized the pretrained word embeddings (with 300 dimensions) for google news dataset. The experiments were applied on Gensim library[14], which is a framework for training collection of documents to extract word embeddings. The defined parameters for the process are described as follows:

(a) learning rate = 0.065

(b) embedding size = 100

(c) sliding window size = 2

[14] https://radimrehurek.com/gensim

Word2Vec uses a technique called negative sampling (randomized approach) to optimize the training performance. In order to generalize the results, we train the corpus multiple times and, in each iteration, we shuffle the order of documents and slightly decrease the learning rate parameter, which facilitates faster convergence.

We experimented three different ways for document classification:

(a) Average word vector

(b) Long Short-Term Memory (LSTM) networks

(c) Convolutional Neural Networks (CNN)

We computed the average vector of word embeddings (CBOW, Skipgram and pretrained) representing the document embedding. This vector was fed as input in a Logistic Regression or a neural network classifier.

LSTM and CNN take as input each word vector in the document, which leads to high dimensionality. To overcome with this problem, we consider the 20,000 most frequent words in the corpus and keep up to 1,000 words for each document. If a document has less words, it is filled up with zeros (padding). We used the LSTM model architecture according to Figure 3.14 with 64 number of units and tanh function as activation, which is subsequently fed to a dense layer that calculates the classification probability. CNN was constructed according to Figure 3.15 with the following parameters:

(a) filters = 100 (number of output units of CNN)

(b) kernel size = 2 (CNN considers bigrams)

(c) activation = ReLU

However, when initially training the CNN model, we observed that it was overfitting to the training set. We dealt with this problem by adding a dropout layer after max pooling, which is a regularization technique, that randomly ignores specific set of neurons during the training phase avoiding the units co-adapting. Co-adapting happens when neurons adjust, aiming at reducing the errors of the other units.

The experimental results are presented in Tables 4.10-4.12. Tables 4.10, 4.11 illustrate the accuracy of different embedding models by computing the average word vector and using Logistic Regression and neural networks respectively. Table 4.12 presents the results of different embedding models combined with LSTM and CNN networks.

Table 4.10. Accuracy of different Word2Vec models by computing the average vector and using Logistic Regression classifier

Word2Vec model	**Accuracy**
CBOW	0.884
Skipgram	0.8934
CBOW + Skipgram	**0.8958**
Pretrained (Google News)	0.8777

Table 4.11. Accuracy of Word2Vec CBOW, Skipgram, CBOW-Skipgram and pretrained embeddings (Google News) by computing the average vector and using various neural network classifiers

	Accuracy			
Neural Network Architecture	**CBOW**	**Skipgram**	**CBOW-Skipgram**	**Pretrained**
one hidden layer with 128 units	0.9155	0.9219	0.9219	0.8895
one hidden layer with 256 units	0.9092	0.929	0.919	0.8895
one hidden layer with 512 units	**0.9203**	0.9226	0.9187	0.9037
two hidden layers 32x32	0.91	0.9148	0.9163	0.8958
two hidden layers 64x32	0.9092	0.929	0.9163	0.8966
two hidden layers 128x32	0.914	**0.9313**	**0.9242**	**0.9092**

Table 4.12. Accuracy of Word2Vec CBOW, Skipgram, CBOW-Skipgram and pretrained embeddings (Google News) by computing the average vector and using various neural network classifiers

	Accuracy		
Neural Network Architecture	**CBOW**	**Skipgram**	**Pretrained**
LSTM	**0.9116**	0.9021	0.9006
CNN	**0.9425**	0.9376	0.9392

It is indicated that neural network models outperform the Logistic Regression classifier, when using the average word vectors. Furthermore, pretrained models seem to have less accuracy than the models trained in the corpus, particularly the Skipgram models.

Nevertheless, pretrained models are considered a more robust language model as they are trained in a larger collection of documents. Finally, CBOW with CNN achieved greatest accuracy (0.9425). Next, we demonstrate the model explanations using LIME and EDC.

4.3.1 Explanations using LIME

Figure 4.5 denotes the prediction explanations for Skipgram average vectors with a two hidden-layer neural network (128x32), while Figure 4.6 depicts the explanations of a CNN model over the pretrained word embeddings.

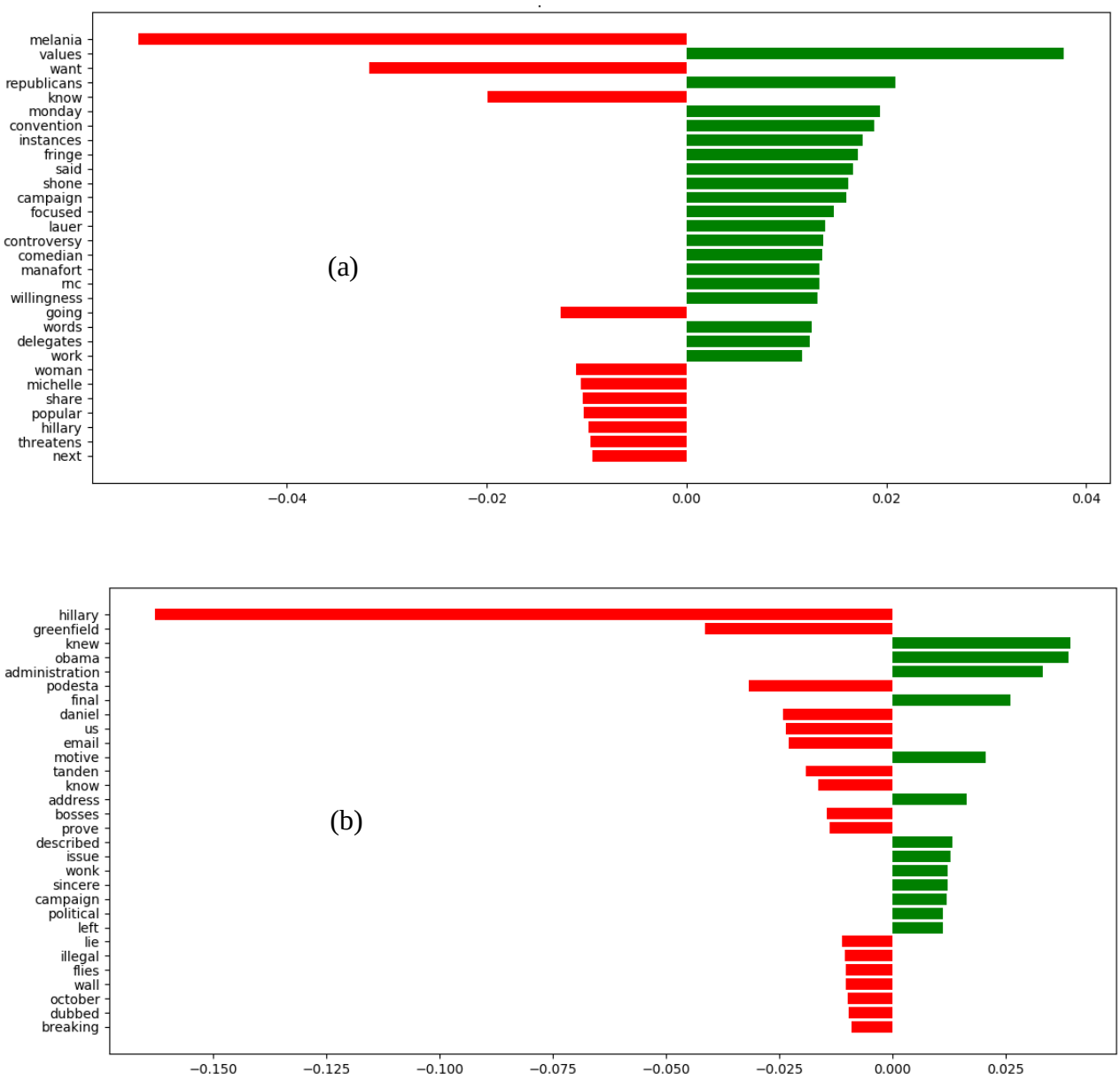

Figure 4.5. LIME explanations for a text classification model with average word vectors as document embeddings and two-hidden-layer neural network. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Non-real news article.

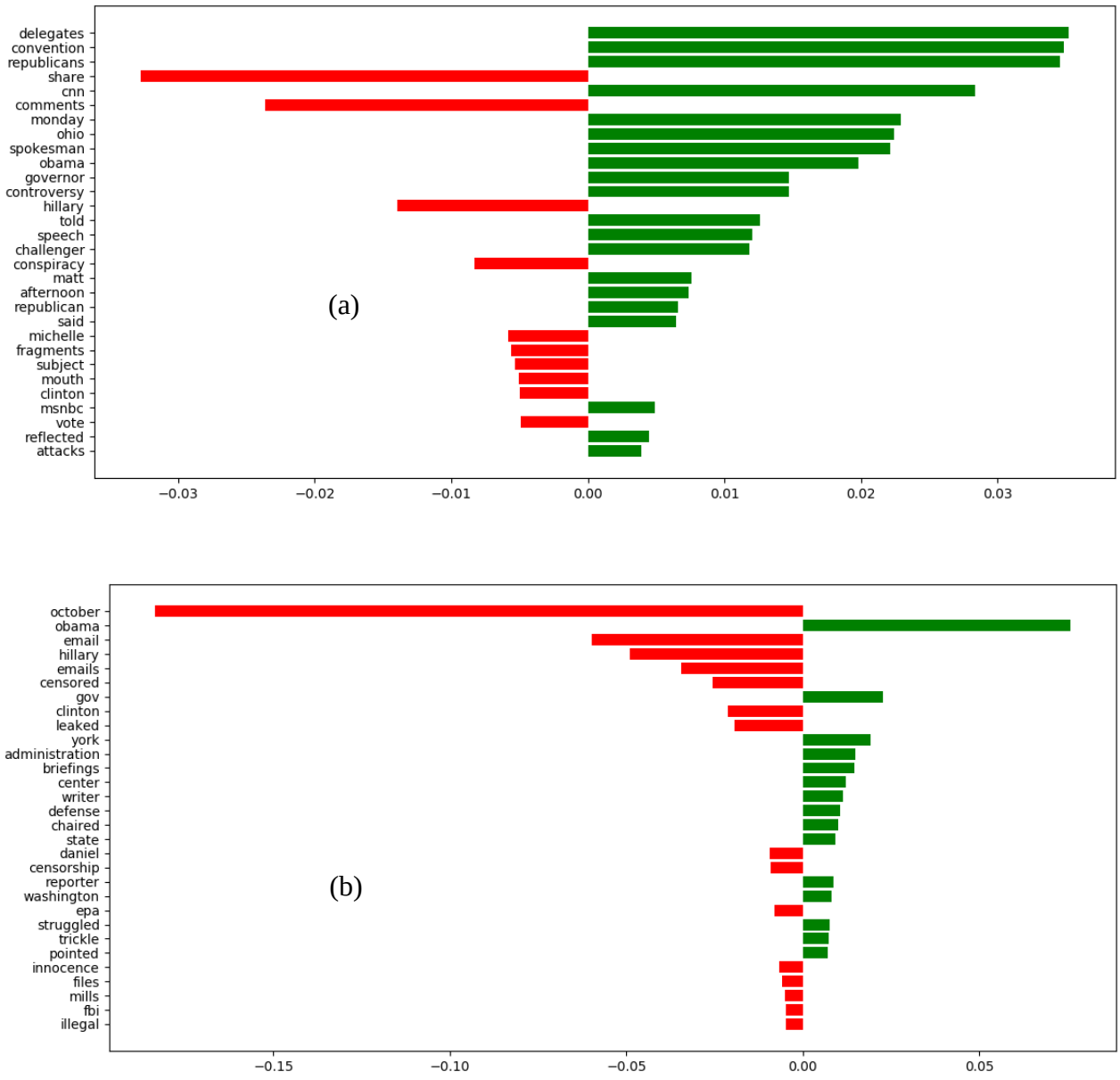

Figure 4.6. LIME explanations for a text classification model with CNN deep neural network model over pretrained word vectors and an additional dense layer. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Non-real news article.

It is observed that these text classification models are biased at certain names or months such as Hillary Clinton or October (associated to the 2016 US election campaign). Furthermore, the explanations are less interpretable than the previous examples.

4.3.2 Explanations using EDC

Table 4.13 shows the explanation results for real and fake news articles of the best performed text classification models with Word2Vec. Similarly to LIME, mostly words like

hillary, obama or october appear to have high contribution to the text classifiers, making difficult for humans to understand and have intuition about the predictions.

Table 4.13. Model explanations of text classification models with pretrained average word vectors and CNN over a neural network using EDC

Model	EDC REAL	EDC FAKE
Average vector	*values, convention, speech, said, afternoon, hard, compound, spokesman, comments, recalled, emphasis, chaotic, governor, told, long, statement, treat, dignity, better, agree*	*hillary, board, wondered, television, obama, doubt, old, dodges, contempt, outright, progress, shillman, times, often, away, tell, emails, administration, establish, factory*
CNN	*cnn, republicans, ohio, monday, delegates, convention, matt, obama, controversy*	*october, email, emails, censorship, hillary, clinton, censored, ever*

4.4 Fake news classification using Doc2Vec

We extracted document embeddings by training the corpus with two Doc2Vec architectures: Distributed Memory (DM) and Distributed Bag of Words (DBOW). We also utilized the combination of DM and DBOW for creating document embeddings. These models were implemented on Gensim library. The training process was executed similarly to Word2Vec. The parameters given for the models are described as follows:

a) learning rate = 0.065
b) embedding dimension = 100
c) window size = 4

The experimental results regarding the accuracy of various Doc2Vec models are presented in Table 4.14, which denote that DBOW outperforms the rest Doc2Vec models.

Table 4.14. Accuracy of Doc2Vec using Logistic Regression classifier

Doc2Vec model	Accuracy
DM	0.8698
DBOW	**0.9274**
DM + DBOW	0.9084

In the next subsections we demonstrate the explanation results using LIME and EDC.

4.4.1 Explanations using LIME

Figure 4.7 presents the prediction explanations for document embeddings using Doc2Vec with a one-hidden-layer neural network of 256 units. The results indicate that formal words like willingness or transparent refer mostly to real news articles, while words with high polarity, such as fake, rogue, illegal contribute mainly to fake news classification.

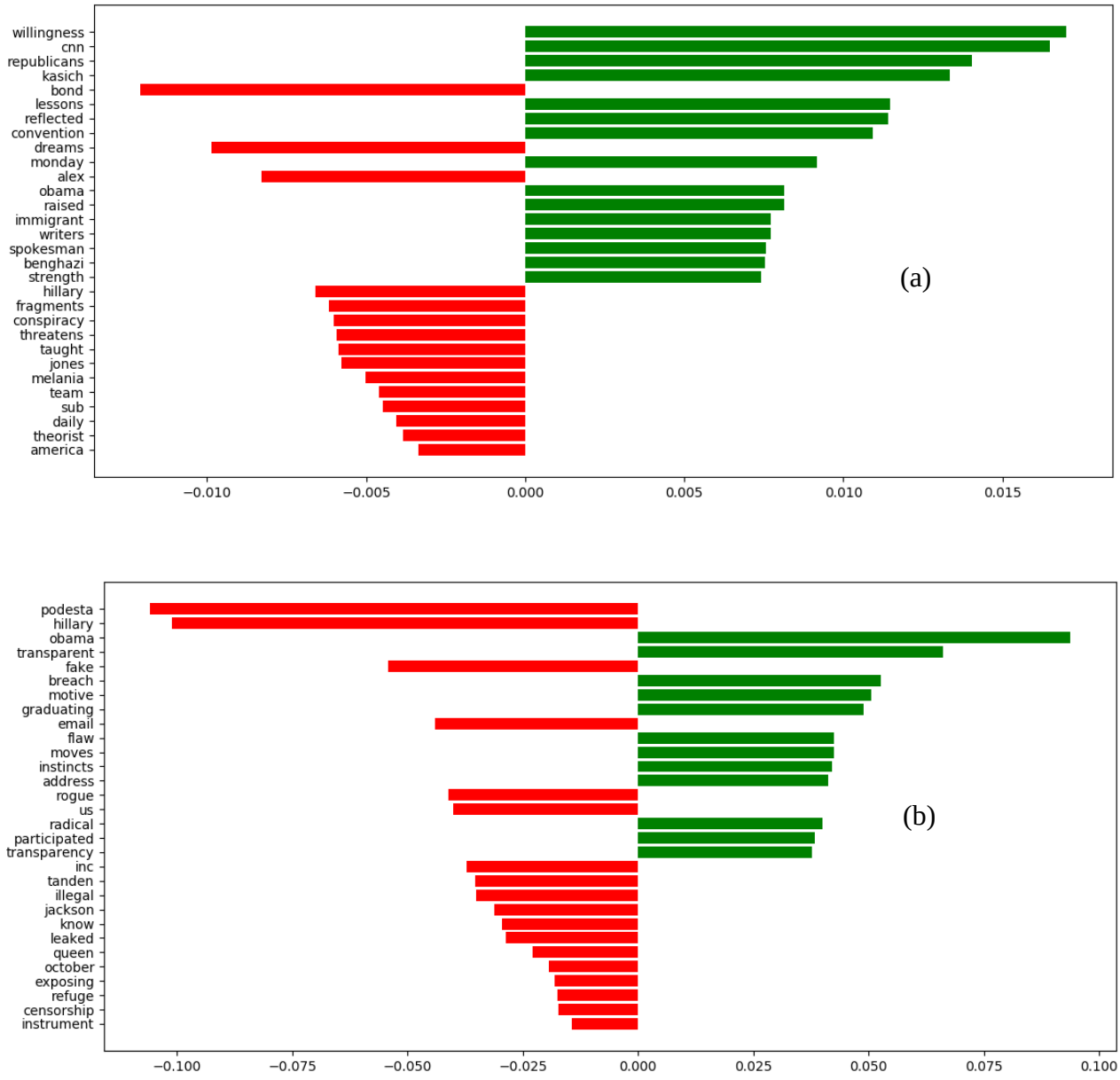

Figure 4.7. LIME explanations for a text classification model with Doc2Vec and one-hidden-layer neural network of 256 units. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Non-real news article.

4.4.2 Explanations using EDC

Table 4.15 shows the EDC model explanations for the predictions of a text classifier with Doc2Vec-DBOW as feature extractor and a one-hidden-layer neural network of 256 units as machine learning model. Contrary to LIME, in this case EDC does not provide accurate and comprehensible explanations.

Table 4.15. EDC explanations for a text classification model with Doc2Vec and one-hidden-layer neural network of 256 units

Model	EDC REAL	EDC FAKE
DBOW	-	*government, hollow, podesta, us, wanted, group, censorship, quickly, admit, project, clean, windsor, supposed, smoothly, looked, clintonworld, barack, simultaneously, part, allies*

4.5 Fake news detection using sentence vectors

Apart from pretrained word embeddings we also utilized pretrained paragraph embeddings to feed as input to the machine learning model. Specifically, we used the pretrained google universal encoder [60] which extracts 512-dimensional paragraph vectors and is provided in Tensorflow Hub. We fed these vectors as input to a one hidden-layer (with 256 units) neural network. The accuracy of this model was 0.8745, which is less than the previous models. Figure 4.8 shows the explanation results using LIME over google universal sentence embeddings with a hidden layer neural network. Also the LIME explanations indicate that the model is more biased to certain names than the aforementioned methods.

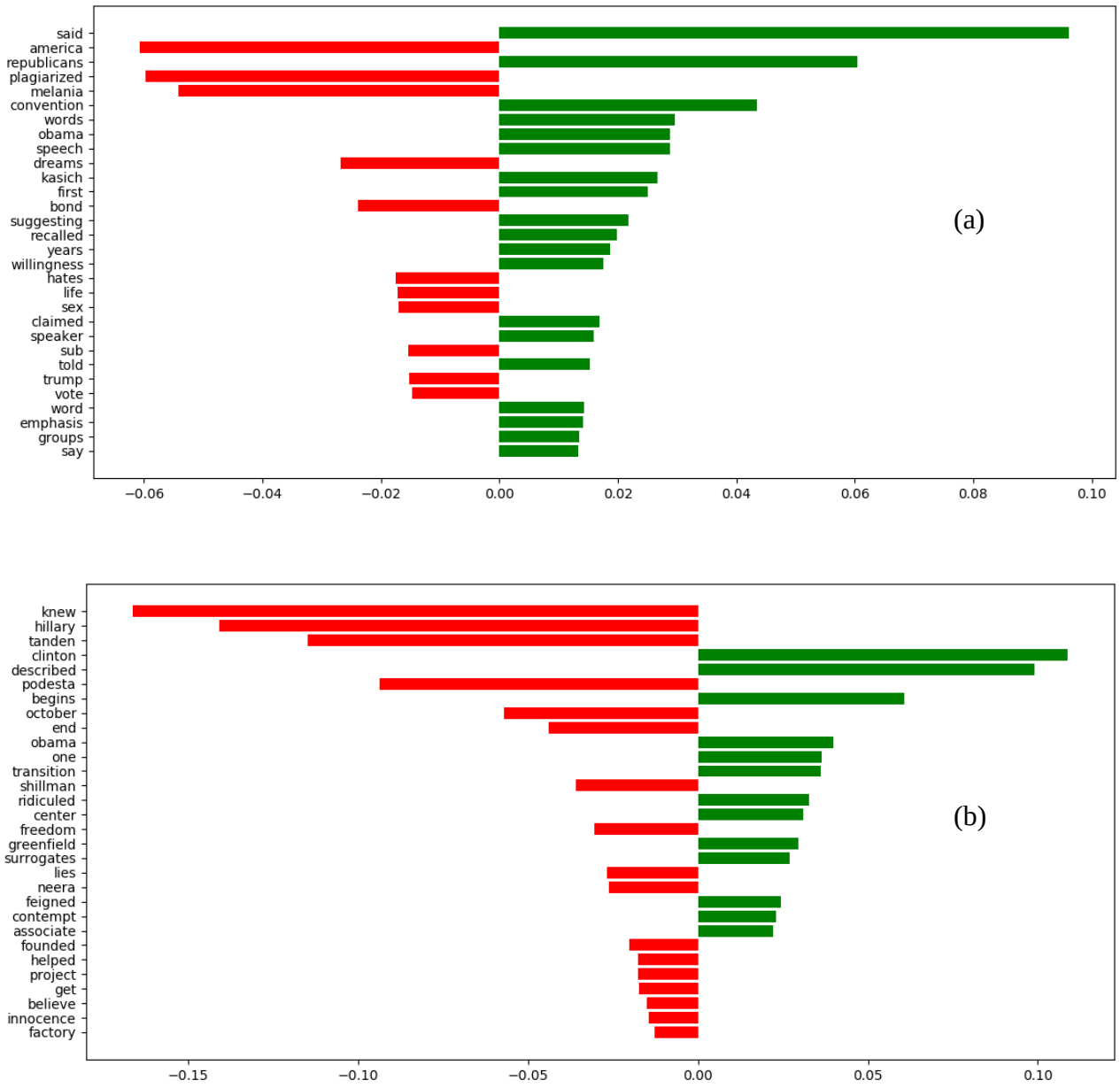

Figure 4.8. LIME explanations for a text classification model using pretrained universal sentence embeddings and one-hidden-layer neural network of 256 units. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Nonreal news article.

4.6 Bigram model explanations using LIME

LIME provides explanations as a set of unique words with their weights denoting the contribution to the black box model decisions. However, producing phrases of words as explanations is more understandable to humans. Taking this problem into consideration, we utilized the Phraser and Phrases classes provided in Gensim library, which, according to [56], [66], is an unsupervised approach that uses mutual information for detecting

bigrams in the corpus and merging them using underscore symbol. However, as the training phase was conducted with the original form of text, we configured the predict probability function of the text classifier, that LIME receives as input, to split the words containing underscore leading to the raw form of documents. Figure 4.9 illustrates the prediction explanations for a text classification model of TF-IDF and Logistic Regression classifier. The results show that bigrams are more interpretable as they contain also full names or certain phrases that give more information about the classification problem.

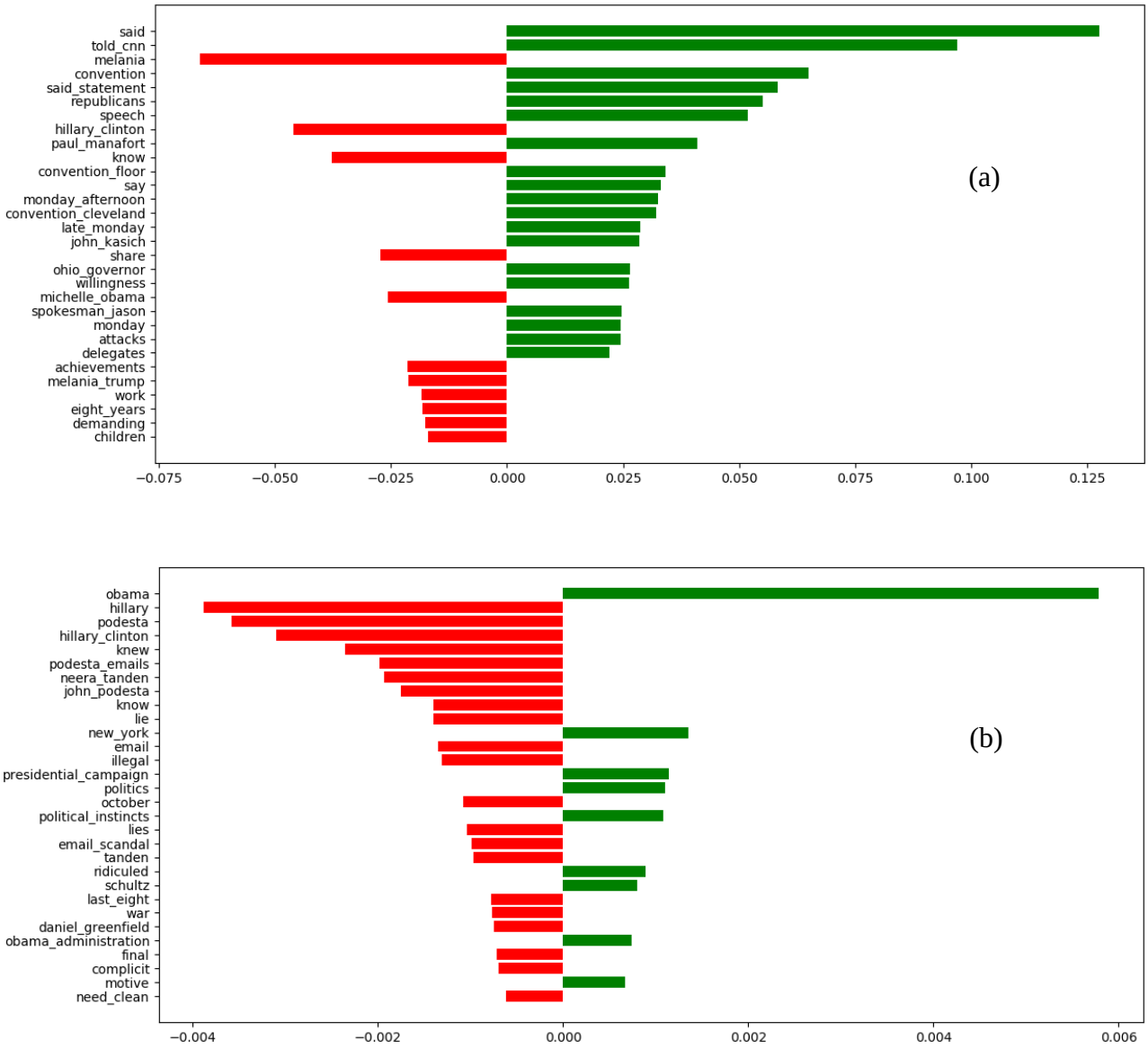

Figure 4.9. LIME explanations, containing also bigrams, for the TF-IDF model with unigrams, bigrams and trigrams using Logistic Regression classifier. Negative values (red bars) indicate contribution to fake news, while positive values (green bars) to real news. (a) Real news article, (b) Non-real news article.

5 Discussion

The outcomes presented in the previous section support the existing findings in the literature about fake news characteristics [13]. As anticipated, our experiments prove that real news tends to have more formal words and words that refer in third person (e.g. said, told, stated). In other words, they follow the writing standards for objective journalism. On the contrary, fake news use vocabulary that aims at manipulating and affecting people's opinion, as well as at contradicting them against certain parties such as lie, rogue, breaking, threatens. Moreover, they use simple words that are easier for the people to read and have less document length. Therefore, producing a set of words that explains the decisions of the disinformation detection model can provide interpretability and information to the users in respect of the fake news characteristics. In that way, the reader will read the article more carefully without being easily affected by the article, a fact which leads to the hampering of disinformation dissemination on the Web.

The experimental results indicated that even though TF-IDF is a simple feature extraction model, it still achieved the highest prediction accuracy in both machine learning models we used (Logistic Regression and Artificial Neural Network). The overall best accuracy over Logistic Regression was obtained by TF-IDF with skipgrams (3-skips considering unigrams, bigrams and trigrams), while regarding the neural networks, TF-IDF with n-grams (unigrams, bigrams and trigrams) outperformed the rest of the models with accuracy 0.9519. Table 5.1 presents an overview of the best results for each method.

Regarding the explanation algorithms, we deduce that EDC and LIME provide comparable results. However, EDC is slower than LIME, as previously demonstrated in the literature [46]. EDC's performance declines with high dimensional documents, as it requires the classifier to run multiple times, while LIME has a fixed number of perturbated samples given by the user. In addition to that, LIME provides more insights about the classifier's behavior, as it also returns words related to the opposite classification. The most comprehensible explanations were produced by TF-IDF with Logistic Regression, indicating that simple models can be less biased than complex models, which tend to overfit more easily. Explanations of LSTM and CNN did not provide accurate results,

which we believe happens because these models are trained on sequential data and are sensitive to perturbations of text, as each word is highly depended on its contextual information.

Table 5.1. Overview of the different models' accuracy

	Accuracy	
Model	**Logistic regression**	**Neural network**
TF-IDF with n-grams	0.94	**0.9519**
TF-IDF with skipgrams	**0.944**	0.9495
TW-IDF	0.9282	0.9353
TW-ICW	0.9361	0.9384
TW-ICW-LW	0.9376	0.9369
Word2Vec average vector	0.8958	0.9313
Word2Vec LSTM	-	0.9116
Word2Vec CNN	-	0.9425
Doc2Vec	0.9274	0.9376

The results were promising, however, we are aware that our research may have some limitations. First, in our experiments we used a relatively small dataset, making it difficult for Deep Learning models (CNN and LSTM) to overcome the state-of-the-art machine learning models. Second, the results indicate that the algorithms are biased at certain names and dates. Third, we removed stop-words, punctuation marks and numbers while we did not split the document in sentences and paragraphs. Finally, we only considered textual information without including metadata, network data or multimedia data like video and images.

In order to overcome the aforementioned limitations, the following measures are suggested: First, the results indicate that training the corpus in Word2Vec framework to extract word vectors, outperformed the pretrained models, denoting the importance of creating a bigger dataset that can capture most of the fake news characteristics. Second, we propose the removal of names using named entity recognition techniques leading to a fair algorithm. We believe that disinformation detection using only text should detect the indent of the writer to harm. Third, we address the need of a consistent dataset that contains apart from textual information user profiles, metadata and video or image content. In that

way, network analysis techniques can be applied to obtain more insights regarding an imminent spread of a fake news content.

In addition, there are further challenges about disinformation detection that shall be considered. The existing model explanation algorithms for text classification provide a set of words as explanations, which is not entirely comprehensible to humans. In this thesis, we dealt with this problem by using an unsupervised approach to detect bigrams in the document that have a unique meaning, something that appeared to be more understandable. However, this can be further improved by providing certain sentences as explanations or using natural language generation techniques to create a better description of the explanations. Furthermore, we believe, that is also important to understand the reasons for an imminent dissemination of disinformation. For example, a fake news article referring to a politician or a celebrity in general is more probable to be spread across the Web. Therefore, we propose, as future direction, the development of a more sophisticated model that contains components for detecting deceptiveness intent in a text, as well as components for discovering whether a news article will become viral.

6 Conclusions

In this thesis we tackled the problem of disinformation detection by implementing various text classification models using supervised approaches and enhancing transparency of our findings by producing classifier's prediction explanations using LIME and EDC algorithms. The algorithms were tested on a balanced dataset containing news articles with reliable or unreliable annotations. The evidence of this study suggests that fake news dissemination can be hampered by automatically detecting such content and providing model explanations. This way a reader can be informed that an article is considered unreliable with some probability and on top of that, the model provides explanations of the predictions as evidence supporting its claim. Moreover, model explanations can show inconsistencies in the disinformation detection problem to further improve model efficiency. The results indicated that TF-IDF with Logistic Regression produces the most reasonable explanations and achieves high accuracy of 94%. We also concluded that LIME outperformed EDC algorithm in terms of interpretability and performance, proving that it remains one of the most reliable model-agnostic explanation methods for text classification.

Finally, a few potential limitations should be considered. First, in our experiments we used a dataset of 6335 news articles. Since the dataset is relatively small, it shall be difficult to utilize Deep Learning methods, which can overperform the state-of-the-art machine learning models when they are trained on big amount of data. Second, the disinformation detection models were biased against certain names and dates.

We believe that utilizing named entity recognitions techniques can improve the algorithm's fairness aiming only at predicting the intent to harm or spread false information, which is the definition of disinformation. Another future direction is to create a new rich fake news detection dataset of bigger size that will also contain image and video content (if needed), author profile information, users' comments and other metadata, which can assist researchers to train also Deep Learning models and validly compare different fake news detection methods.

Bibliography

[1]. Kwak, H., Lee, C., Park, H. and Moon, S., 2010, April. What is Twitter, a social network or a news media?. In *Proceedings of the 19th international conference on World wide web* (pp. 591-600). AcM.

[2]. Allcott, H. and Gentzkow, M., 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, *31*(2), pp.211-36.

[3]. Kumar, S., West, R. and Leskovec, J., 2016, April. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web* (pp. 591-602). International World Wide Web Conferences Steering Committee.

[4]. Tandoc Jr, E.C., Lim, Z.W. and Ling, R., 2018. Defining "fake news" A typology of scholarly definitions. *Digital Journalism*, *6*(2), pp.137-153.

[5]. Kershner, J. (2018). *The Elements of News Writing.* Boston: Pearson Allyn and Bacon.

[6]. Richardson, B. (2007). *The Process of Writing News: From Information to Story*. Boston: Pearson

[7]. Jamieson, K.H. and Campbell, K.K., 2000. The interplay of influence: News, advertising, politics, and the mass media.

[8]. Kovach, B. and Rosenstiel, T., 2014. *The elements of journalism: What newspeople should know and the public should expect*. Three Rivers Press (CA).

[9]. Shoemaker, P.J. and Reese, S.D., 2013. *Mediating the message in the 21st century: A media sociology perspective*. Routledge.

[10]. Mecacci, A., 2016. Aesthetics of Fake. An Overview. *Aisthesis. Pratiche, linguaggi e saperi dell'estetico*, *9*(2), pp.59-69.

[11]. Shu, K., Sliva, A., Wang, S., Tang, J. and Liu, H., 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, *19*(1), pp.22-36.

[12]. Tucker, J., Guess, A., Barberá, P., Vaccari, C., Siegel, A., Sanovich, S., Stukal, D. and Nyhan, B., 2018. Social Media, Political Polarization, and Political Disinformation: A Review of the Scientific Literature.

[13]. Kumar, S. and Shah, N., 2018. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559*.

[14]. Fisher, M., Cox, J.W. and Hermann, P., 2016. Pizzagate: From rumor, to hashtag, to gunfire in DC. *Washington Post*.

[15]. Forelle, M., Howard, P., Monroy-Hernández, A. and Savage, S., 2015. Political bots and the manipulation of public opinion in Venezuela. *arXiv preprint arXiv:1507.07109*.

[16]. House of Commons (2018). *Disinformation and Fake News: An Interim Report*. Fifth Report of Session 2017–19. House of Commons: Digital, Culture, Media and Sport Committee.

[17]. Bollen, J., Mao, H. and Zeng, X., 2011. Twitter mood predicts the stock market. *Journal of computational science*, *2*(1), pp.1-8.

[18]. Gupta, A., Lamba, H., Kumaraguru, P. and Joshi, A., 2013, May. Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 729-736). ACM.

[19]. Shao, C., Ciampaglia, G.L., Flammini, A. and Menczer, F., 2016, April. Hoaxy: A platform for tracking online misinformation. In *Proceedings of the 25th international conference companion on world wide web* (pp. 745-750). International World Wide Web Conferences Steering Committee.

[20]. Zubiaga, A., Liakata, M., Procter, R., Hoi, G.W.S. and Tolmie, P., 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, *11*(3), p.e0150989.

[21]. Silverman, C., 2016. This analysis shows how viral fake election news stories outperformed real news on Facebook. *BuzzFeed News*, *16*.

[22]. Friggeri, A., Adamic, L., Eckles, D. and Cheng, J., 2014, May. Rumor Cascades. In *Eighth International AAAI Conference on Weblogs and Social Media*.

[23]. Vosoughi, S., Roy, D. and Aral, S., 2018. The spread of true and false news online. *Science*, *359*(6380), pp.1146-1151.

[24]. Zannettou, S., Sirivianos, M., Blackburn, J. and Kourtellis, N., 2018. The Web of False Information: Rumors, Fake News, Hoaxes, Clickbait, and Various Other Shenanigans. *arXiv preprint arXiv:1804.03461*.

[25]. Parikh, S.B. and Atrey, P.K., 2018, April. Media-Rich Fake News Detection: A Survey. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)* (pp. 436-441). IEEE.

[26]. Pérez-Rosas, V., Kleinberg, B., Lefevre, A. and Mihalcea, R., 2017. Automatic Detection of Fake News. *arXiv preprint arXiv:1708.07104*.

[27]. Pennebaker, J.W., Francis, M.E. and Booth, R.J., 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, *71*(2001), p.2001.

[28]. Fürnkranz, J., 1998. A study using n-gram features for text categorization. *Austrian Research Institute for Artificial Intelligence*, *3*(1998), pp.1-10.

[29]. Afroz, S., Brennan, M. and Greenstadt, R., 2012, May. Detecting hoaxes, frauds, and deception in writing style online. In *Security and Privacy (SP), 2012 IEEE Symposium on* (pp. 461-475). IEEE.

[30]. Jin, Z., Cao, J., Zhang, Y., Zhou, J. and Tian, Q., 2017. Novel visual and statistical image features for microblogs news verification. *IEEE transactions on multimedia*, *19*(3), pp.598-608.

[31]. Ruchansky, N., Seo, S. and Liu, Y., 2017, November. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (pp. 797-806). ACM.

[32]. Ma, J., Gao, W., Wei, Z., Lu, Y. and Wong, K.F., 2015, October. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (pp. 1751-1754). ACM.

[33]. Qazvinian, V., Rosengren, E., Radev, D.R. and Mei, Q., 2011, July. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1589-1599). Association for Computational Linguistics.

[34]. Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z. and Yu, P.S., 2018. TI-CNN: Convolutional Neural Networks for Fake News Detection. *arXiv preprint arXiv:1806.00749*.

[35]. McIntire, G. (2018). *How to Build a "Fake News" Classification Model*. [online] Open Data Science - Your News Source for AI, Machine Learning & more. Available at: https://opendatascience.com/how-to-build-a-fake-news-classification-model/ [Accessed 4 Dec. 2018].

[36]. Wang, W.Y., 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

[37]. Mitra, T. and Gilbert, E., 2015, May. CREDBANK: A Large-Scale Social Media Corpus With Associated Credibility Annotations. In *ICWSM* (pp. 258-267).

[38]. Santia, G.C. and Williams, J.R., 2018. BuzzFace: A News Veracity Dataset with Facebook User Commentary and Egos. *ICWSM*, *531*, p.540.

[39]. Molnar, C., 2018. Interpretable Machine Learning. *A Guide for Making Black Box Models Explainable*.

[40]. Ribeiro, M.T., Singh, S. and Guestrin, C., 2016, August. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135-1144). ACM.

[41]. Martens, D. and Provost, F., 2014. Explaining data-driven document classifications. *MIS Quarterly*, *38*(1), pp.73-100.

[42]. Hu, L., Chen, J., Nair, V.N. and Sudjianto, A., 2018. Locally Interpretable Models and Effects based on Supervised Partitioning (LIME-SUP). *arXiv preprint arXiv:1806.00663*.

[43]. Ribeiro, M.T., Singh, S. and Guestrin, C., 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*.

[44]. Laugel, T., Renard, X., Lesot, M.J., Marsala, C. and Detyniecki, M., 2018. Defining Locality for Surrogates in Post-hoc Interpretablity. *arXiv preprint arXiv:1806.07498*.

[45]. Singh, C., Murdoch, W.J. and Yu, B., 2018. Hierarchical interpretations for neural network predictions. *arXiv preprint arXiv:1806.05337*.

[46]. Chen, J., Song, L., Wainwright, M.J. and Jordan, M.I., 2018. Learning to Explain: An Information-Theoretic Perspective on Model Interpretation. *arXiv preprint arXiv:1802.07814*.

[47]. Horne, B.D. and Adali, S., 2017. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *arXiv preprint arXiv:1703.09398*.

[48]. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K., 2017. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*.

[49]. Ramos, J., 2003, December. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, pp. 133-142).

[50]. Guthrie, D., Allison, B., Liu, W., Guthrie, L. and Wilks, Y., 2006, May. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)* (pp. 1-4).

[51]. Schütze, H., Manning, C.D. and Raghavan, P., 2008. *Introduction to information retrieval* (Vol. 39). Cambridge University Press.

[52]. Leskovec, J., Rajaraman, A. and Ullman, J.D., 2014. *Mining of massive datasets*. Cambridge university press.

[53]. Skianis, K., Malliaros, F. and Vazirgiannis, M., 2018. Fusing Document, Collection and Label Graph-based Representations with Word Embeddings for Text Classification. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)* (pp. 49-58).

[54]. Rousseau, F. and Vazirgiannis, M., 2013, October. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (pp. 59-68). ACM.

[55]. Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[56]. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

[57]. Rong, X., 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

[58]. Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., 2016. *Deep learning* (Vol. 1). Cambridge: MIT press.

[59]. Le, Q. and Mikolov, T., 2014, January. Distributed representations of sentences and documents. In *International Conference on Machine Learning* (pp. 1188-1196).

[60]. Cer, D., Yang, Y., Kong, S.Y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C. and Sung, Y.H., 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

[61]. Aggarwal, C.C. and Zhai, C., 2012. A survey of text classification algorithms. In *Mining text data* (pp. 163-222). Springer, Boston, MA.

[62]. Caudill, M., 1987. Neural networks primer, part I. *AI expert*, *2*(12), pp.46-52.

[63]. Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

[64]. Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[65]. Kim, Y., 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[66]. Bouma, G., 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pp.31-40.

Appendix

Explaining Document Classifications (EDC) algorithm in Python:

```
# Import the necessary Python libraries
import numpy as np
import pandas as pd
import itertools
from tqdm import tqdm

def delete_from_text(words, text):
    alist = [word for word in text.split() if word not in words]
    return ' '.join(alist)

def combine_lists(base_list, expand_list):
    out = []
    comb = list(itertools.product([base_list], expand_list))
    for item in comb:
        out.append(item[0] + [item[1]])
    return out

def prune_list(list2d, R):
    for i in range(len(list2d)):
        for j in range(len(R)):
            list2d[i] = list(set(list2d[i]) - set(R[j]))
    return [el for el in list2d if el]


def explain_instance(text, predict_proba, max_epochs=30):

    # text: instance to explain
    # predict_proba: predict probability function
    # epochs: maximum number of iterations

    # Correspoding probability score
    # predict_proba should return 1 value
    p = predict_proba([text])[0][1]
    # Class predicted by trained classifier
    c = 1 if p > 0.5 else 0

    # Explanation list
    R = []
    # Probabilities explanation list
    P = []
    combinations_to_expand_on = []
    P_combinations_to_expand_on = []

    # List of distinct words in document
    Wd = list(set(text.split()))

    max_pnew_diff = 0
    counter = -1
```

```
max_index = 0
d = 0
for i, word in enumerate(Wd):
    new_text = delete_from_text([word], text)
    pnew = predict_proba([new_text])[0][1]
    cnew = 1 if pnew > 0.5 else 0
    # print(word, pnew, cnew)

    if cnew != c:
        R.append([word])
        P.append((p, pnew))
        del Wd[i-d]
        c += 1

    else:
        combinations_to_expand_on.append([word])
        P_combinations_to_expand_on.append(pnew)
        counter += 1
        if abs(p - pnew) > max_pnew_diff:
            max_pnew_diff = abs(p - pnew)
            max_index = counter



combo = []
for epoch in tqdm(range(max_epochs)):

    # max_index = np.argmax(abs(p - np.array(P_combinations_to_ex-
    # pand_on)))
    combo = combinations_to_expand_on[max_index]
    combo_set = combine_lists(combo, Wd)
    combo_set = prune_list(combo_set, R)

    for comb in combo_set:
        new_text = delete_from_text(comb, text)
        pnew = predict_proba([new_text])[0][1]
        cnew = 1 if pnew > 0.5 else 0

        if cnew != c:
            R.append(comb)
            P.append((p, pnew))


        else:
            combinations_to_expand_on.append(comb)
            P_combinations_to_expand_on.append(pnew)
            counter += 1
            if abs(p - pnew) > max_pnew_diff:
                max_pnew_diff = abs(p - pnew)
                max_index = counter
return np.array(R), np.array(P)
```