Improving Fake News Detection with Linguistic Cues

Koumouridis Georgios

SID: 3308180008

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2019

THESSALONIKI – GREECE

Improving Fake News Detection with Linguistic Cues

Koumouridis Georgios

SID: 3308180008

Supervisor: Dr. Christos Berberidis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2019

THESSALONIKI – GREECE

Abstract

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

False information can be created and reproduced easily through the Web due to the rapid growth of online social media that enable sharing ideas and thoughts through virtual networks and communities. There are stories that are deliberately fabricated to influence public opinion and cause confusion to the reader. Thus, tackling fake news has become an important issue over the past few years. Several detection approaches have been proposed, such as but not limited to fact-checking and network analysis. However, to the best of our knowledge, most of them can identify fake news after its propagation. Understanding the writing style of fake news could support the early detection of disinformation. In this thesis, we investigate the use of linguistic features for identifying deception in news articles and experiment with state-of-the-art machine learning and natural language processing approaches. Furthermore, we prove that integrating linguistic characteristics into fake news detection models, can improve their prediction accuracy.

Acknowledgements

I would first like to thank my thesis supervisor Dr. Christos Berberidis for giving me the opportunity to elaborate on such an interesting field of research. His guidance and constructive feedback during the development of this work were crucial for the completion of this study. I would also like to acknowledge the PhD candidate, Konstantinidis Ioannis who provided me with his valuable experience on Fake News. Finally, I have to express my gratitude to my family and friends for their support and continuous encouragement throughout my studies.

Thank you.

Koumouridis Georgios

02/12/2019

Contents

1 Introduction

Over the recent years, the growth of social media on the Web has greatly facilitated the way people communicate and share information. Social media platforms such as Twitter and Facebook play a vital role for people to seek out and spread information due to the fact, they can connect with users all around the world and stay informed about tending events. Latest studies have shown [1] that people increasingly get their news from digital media than from traditional news sources as they are known for their immediacy and because it is a free way to read the news. Therefore, social media platforms form a very powerful network allowing the circulation of a huge amount of data.

Nonetheless, the convenience of producing and distributing information to such a substantial number of the population also fosters the risk of dissemination of unreliable information, commonly referred as false information. This can be a result of poor journalism or naïve assumptions and claims produced by individuals either as a consequence of their gullible nature or the lack of critical thinking. This type of false information is also met as misinformation. On the other hand, there is another form of false information, defined as disinformation that deliberately aims to harm the user. This type is highly associated with fake news, which are articles of intentionally false content with purpose to mislead or guide public opinion. A sudden outburst of popularity towards fake news has occurred during the US elections in 2016, where many fake news stories were published to affect public opinion against specific political parties [2]. It is no wonder then, that there has been an increase in awareness and outrage towards false information and social media.

However, people are not always in the position to distinguish between a reliable and an untrustworthy piece of news. As indicated by the authors of [3], people performed poorly when called to evaluate articles based on their content. The task was to recognize between real and hoax articles and the result was to correctly classify only the 66% of them. Given the fact that the random prediction would be 50%, there is still gap for improvement. Moreover, the enormous amount of data flowing around the Web on a daily basis, is not possible to be checked manually about its veracity. Therefore, it is essential to develop an automatic fake news detection model to prevent the dissemination of fraud claims. Attempts on solving the problem automatically, through classification with machine learning algorithms by exploiting multiple

sources (article's content, user responses, information flow) of data, have achieved great success in the past. Although, in order to prevent the expansion of false information at an early stage it is crucial to develop a model using only content information [4]. Understanding the linguistic characteristics of deceptive language could not only provide useful insights for previous works regarding model explainability [5], but also support the early detection of fake news.

In this thesis, we encounter the problem of fake news detection by applying several machine learning (ML) algorithms utilizing only textual data. Also, following the work done by the authors in [6] we test the assumption that linguistic-based features can improve the accuracy of a fake news detection model by employing more complex ML architectures such as Neural Networks. We prove that integrating linguistic characteristics into fake news detection models, can improve their prediction accuracy

The rest of the thesis is organized as follows: Chapter 2 sets the background of fake news by providing on overview of past research studies regarding the definition of fake news, the detection models and methods, the different feature extraction techniques and the available datasets. Chapter 3 describes the dataset used for out experiments and the implemented text classification approaches. Chapter 4 presents and compares the results on different machine learning architectures. In chapter 5 we discuss the findings of our study and detect some limitations. Finally, chapter 6 gives a conclusion of what we have achieved and sets some future direction to overcome the limitations.

2 Literature review

This chapter provides a review of state-of-the art algorithms for text classification. Moreover, it covers the research literature on linguistic features and fake news detection.

2.1 Text Classification techniques

Text classification is the procedure of assigning labels or categories to text according to its content and it is a fundamental Natural Language Processing (NLP) task. After having preprocessed and cleaned text from redundant words and characters to reduce its dimensionality, the next step towards training a text classifier with machine learning is feature extraction. Documents are unstructured data and must be converted into a numerical representation in the form of a vector. These vectors are used as input for training machine learning models for text classification. This section provides an overview of the methods that have been widely acquired for document classification.

2.1.1 Traditional (Non-Deep) Learning Approaches

Rocchio Algorithm

Rocchio Algorithm's primary use was to return relevance feedback when querying full-text databases and later, it has found many applications in text classification problems [7]. The algorithm exploits the TF-IDF weights for every word and creates a prototype vector for each class by averaging the vectors of the training documents within the same class. The test documents are labeled by calculating the maximum similarity between them and the prototype vectors [8]. The average vector derives from the centroid of the corresponding class and the formula is given below:

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \overrightarrow{v_d}$$

where D_c is the subset of documents in the original dataset *D*, belonging in class C and $\overrightarrow{v_d}$ is the vector representation corresponding to the document *d*. The documents are assigned to each class with the smallest Euclidean distance from the centroid [8].

Logistic Regression

Logistic Regression is a linear classifier that uses a decision boundary [9]. In other words, it predicts the probability of an instance to belong in a class rather than the actual class. It is a regression model that learns the weights of every feature as indicated below:

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

where w_i are the weights of the features and x_{ij} are the features of document i.

In a binary classification problem the probability of a record to be classified in one of the two classes is given as:

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

where $y_j \in \{1, -1\}$ is the label. This equation is the sigmoid function and takes values between 0 and 1, therefore producing probability models. The record is labeled with 1 if the probability is above the threshold (i.e., 0.5) or with -1 if it is below.

In case of a multi-classification problem, the instances are classified according to the highest value of $P(Y = y_j|X)$.

During the training phase of a Logistic Regression model, the parameters w_i are calculated with the maximum likelihood (ML) estimation. For the sake of easy computing, the logarithmic likelihood is optimized and is formulated as:

$$w = argmax_w \sum_i^n \frac{1}{1 + e^{-y_j f(X_i)}}$$

Naïve Bayes Classifier

Naïve Bayes classifier has been widely used in text classification due to its simplicity. It assumes that all the attributes are independent to each other, given the class, hence the name "naïve" [10]. The attributes to be classified are words and the number of unique words can be very large. Generally there are two different approaches [11]. The first one assumes that each document in the dataset is represented by a vector with binary values. The values indicate if the word appears or not in the document. The number of times each word appears in the text is not taken into account. The probability of a document to belong in a specific class is given by:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

where *d* is the document and *c* is the class. The decision rule is:

$$\hat{y} = argmax_{c \in C} P(d|c)P(c)$$

The second approach is called the multinomial model and captures the number of times the word *w* occurs in a document. The probability in that case can be written as:

$$P(c|d) = \frac{P(c)\Pi_{w \in d} P(d|c)^{n_{wd}}}{P(d)}$$

Where n_{wd} is the number of times the word *w* appears in the text.

Support Vector Machine

Support Vector Machine (SVM) is a linear classifier usually used for binary classification. The output of the model is a hyperplane which best separates the two classes [12]. This hyperplane is called the decision boundary. For document classification tasks, the text needs to be transformed into a vector of numbers. The most common approach is the frequency of the words within the text, just like the multinomial naïve Bayes model.

Binary classification

Given a set of *n* tagged points $x_1, \dots, x_n$ where the tags $y_i \in \{-1, 1\}$ the goal is to find the hyperplane with maximum margin that divides the two groups. Figure # indicates the linear SVM for a 2-D dataset. The hyperplane can be written as:

$$\vec{w} \cdot \vec{x} - b = 0$$

Where $\vec{w}$ is the vector of the hyperplane.

However, SVM can perform efficiently on a non-linear problem with the kernel trick. The idea is to map the non-linearly separable data points into higher dimensional feature space and find a hyperplane that can divide the transformed samples. The kernel function that is responsible for the data transformation is illustrated below:

$$k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$$

Multi-Classification

There are two techniques to solve a multi-class problem [13]. The first one, one-against-one, is to think of it as a collection of two classification problems. A classifier is used to draw a hyperplane between each class and the remaining classes. Majority voting is employed to classify unseen data. The one-against-all technique constructs boundaries, separating each class from each other and the decision is taken in a similar way.

Ensemble Methods

Ensemble methods is a very popular machine learning technique when it comes to text categorization. This technique combines several weak learning models in order to provide one optimal model [14]. It forms a committee of models and each member of the committee is trained individually. The final output is a combination of the results from the multiple models. In classification the majority voting is employed. The methods discussed below are Bagging Algorithm, Boosting Algorithm and Random Forest (RF) Algorithm.

Bagging Algorithm

Bagging algorithm (Bootstrap Aggregation) creates an ensemble of M weak learning models. Each model is trained on a different bootstrap dataset created from the original dataset. The bootstrap datasets have fewer elements than the original dataset *D* and are formed by randomly selecting data with replacement from it. The outputs are combined by (weighted) majority voting.

Adaboost Algorithm

Boosting is a repetitious method aiming to enhance the predicting performance of weak classifiers. Adaboost (Adaptive Boosting) is a boosting algorithm that through an iterative process focuses on the data points that are harder to classify. More specifically, the first weak classifier is trained using a bootstrap dataset *D1* from the initial dataset *D*. The elements of *D1* are selected with equal probabilities (weights). Next, the probability of choosing a misclassified pattern is increased while decreasing the probability of choosing a correctly classified one. At this stage, a second bootstrap dataset is formed with the updated weights. These stages are repeated until all the members of the committee are trained. The final decision of the committee is the average of the members.

Random Forest Algorithm

Random Forest (RF) is an extension of the bagging algorithm. It consists of a big number of relatively uncorrelated decision trees that form a committee. Each individual tree makes a decision about the existing instance and the class with the most votes becomes the prediction. The structure of the model is presented in figure 2.1.

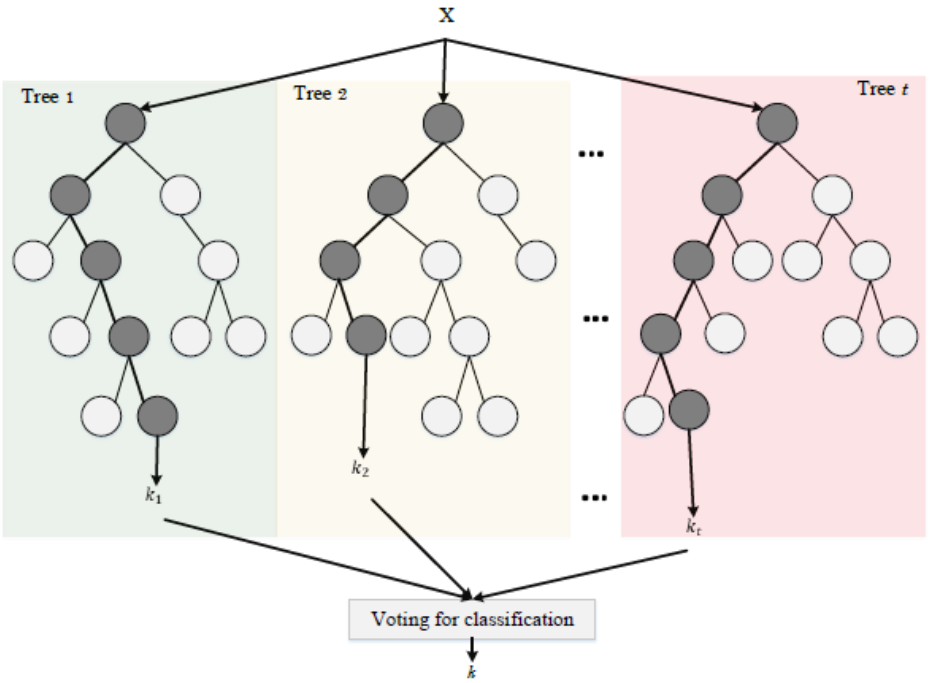

Figure 2.1 [15] Random Forest structure

2.1.2 Deep Learning Approaches

Deep Feed-Forward Neural Networks (DNN)

Neural Networks are based on human brain activity, where the information is passed through specific neurons, leading to the final decision. A typical DNN is designed with multiple connections between consecutive layers of neurons. The connections are assigned to weights. The first layer is the input layer and the last is the output layer. The intermediate layers are called hidden layers. In text classification the input is vectorized strings that derive from the documents. The output layer is only one if we have a binary classification problem or equal to the number of different classes in multi-classification problems. The goal is to predict the relationship between the input and the target through the hidden layers [15].

The training of a DNN is achieved with the back-propagation algorithm with sigmoid (equation 2.1) as the activation function of the output layer and Rectified Linear Unit (ReLU) (equation 2.2) as activation function of the hidden layers in order to avoid the vanishing gradient problem.

$$f(x) = \frac{1}{1 + e^{-x}} \in (0,1) \quad (2.1)$$

$$f(x) = \max(0, x) \quad (2.2)$$

In multi-classification problems, the output layer uses the Softmax function (equation 2.3), and the predicted class is the one with the highest probability [15].

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \; \forall \, j \in \{1,2,\dots,K\} \quad (2.3)$$

where k is each of the classes.

Recurrent Neural Networks

The second method of deep learning neural networks is Recurrent Neural Networks (RNN). RNN are networks with loops, allowing the information to preserve [17]. We can imagine it as multiple copies of a specific neural network, forming a chain that passes the information to a successor. This technique is very powerful for sequential data. Figure # depicts a typical RNN and the unfolded version where Xt is the input vector.

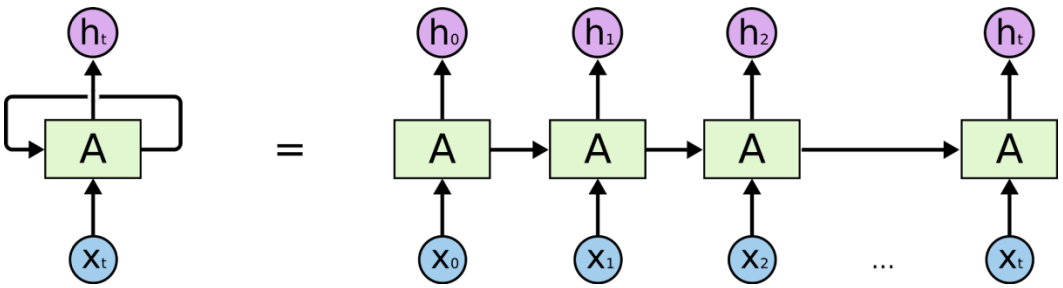

Figure 2.2 A typical[1] RNN architecture and the unfolded version.

The activation function for each layer is the tanh function, returning values in the range (-1, 1). This iterative method can be formulated as:

$$h_t = \tanh(W_{hx}\, x_t + W_{hh} h_{t-1} + b_h)$$

$$y_t = (W_{yh} h_t + b_0)$$

W_{hx} is the input to hidden weight matrix,

W_{hh} is the hidden to hidden weight matrix,

W_{yh} is the hidden to output weight matrix,

[1] https://medium.com/@jianqiangma/all-about-recurrent-neural-networks-9e5ae2936f6e

b_h and b_0 are the biases.

However, as gradient descent is the main algorithm to train a RNN model, the issue of vanishing gradient arises. This inability of preserving long-term dependencies is managed with a special type of RNN, long short-term memory (LSTM). LSTM has a similar structure to a basic RNN although, is uses multiple gates to determine the amount of information that is allowed to pass into each node stage (figure 2.3).

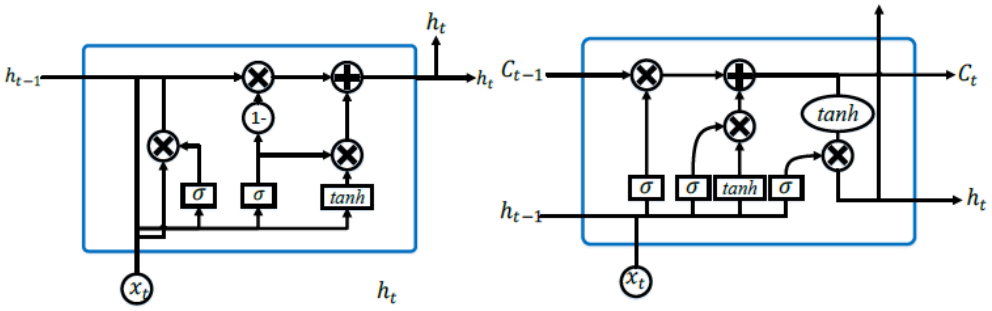

Figure 2.3 [16] The left figure is a GRU cell while the right figure is a LSTM cell.

Convolutional Neural Networks

Convolutional Neural Networks (CNN) have originally used for image classification although, they have also performed equally well in text classification tasks [18]. Instead of having fully connected layers, as it happens in a typical neural network, the convolution layers are connected only to a subset of its preceding layer. These subsets are formed with a pre-defined sliding window (i.e. n-grams). The convolutional layers are called feature maps. A feature c_i is generated for a window of *n* words by:

$$c_i = f(W \cdot X_{i:i+n-1} + b)$$

Where *b* is the bias, *W* is the weight vector and *f* is the activation function (again ReLU is preferred so to avoid the vanishing gradient problem). This filter is applied every sliding window and produces the feature map. To reduce the computational cost, the convolutional layer is connected to a pooling layer while keeping the most important features. The most common pooling technique is the max pooling where the maximum instance is selected. The output of the stacked convolutional layers is mapped to a fully connected softmax layer that calculates the probability distribution over the classes.

Transfer Learning

Traditional machine learning (ML) algorithms are designed to work in isolation, based on specific tasks and datasets. No knowledge is retained and the models have to be rebuilt from scratch when moving to a different task. Moreover, it is difficult to have a dataset for every domain. Supervised ML algorithms fail when we do not have enough training data. Transfer learning allows us to encounter that problem and is considered a state-of-the-art for natural language processing [19].

Transfer learning allows us to utilize knowledge (features, weights etc.) gained from previous trained models and apply them to newer, similar to them. More specifically, given a source domain D_s with a large number of labeled examples and a task T_s , the objective is to transfer the information to another domain, the target domain D_t with a limited number of labeled data and a task T_s.

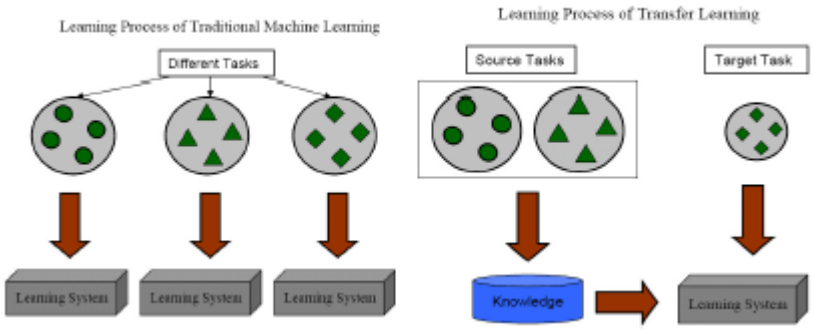

Figure 2.4 [19] Different Learning Processes between Traditional Machine Learning and Transfer Learning

Transfer Learning Strategies for text classification

Inductive Transfer learning: In this case, the target and the source task are different from each other while, there is no matter if the source and the target domains are the same or not. Depending on whether the source domain contains labeled data or not, inductive transfer learning is similar to multitask learning and self-taught learning, respectively.

Transductive Transfer Learning: In this scenario, the source and target tasks are the same, but the two domains are different. In this situation, the source domain has a lot of labeled data, but the target domain has no labeled data available.

Deep Transfer Learning

Deep learning models are an example of inductive transfer learning. As previously mentioned, deep learning models are layer-based architectures. Each different layer learns different

features. Finally, the whole system is connected to the last layer (usually fully connected), to produce the final output. The initial layers capture more generic features, while the latter ones provide features that are more task-related. This architecture allows us to use a pre-trained network on a specific dataset without the final layer and use it as a feature extractor for other tasks with limited data. There are two popular approaches when we want to transfer the knowledge from pre-trained models to new models. The first one is to use them only as feature extractors. The main idea is to exploit the pre-trained model's weighted layers without updating them during the training on the new dataset for the new task. The other one is called fine-tuning. This is a more demanding technique where instead of only replacing the final layer, we also retrain some of the previous layers (i.e. with backpropagation). The authors in [20] observed that Universal Language Model Fine-tuning (ULMFiT), an inductive transfer learning model, performs really well in many NLP tasks. ULMFiT pre-trains a language model (i.e. LSTM) on a general-domain corpus and fine-tunes it on the target task.

2.2 Linguistic features

The "traditional" machine learning approach to an NLP task is to extract features through feature engineering, train the model on parameters with the training dataset and apply the model to the test data. Since NLP has to do with textual data, it is reasonable to exploit linguistic features that can capture the different writing style of a writer or a group of documents. These features derive from the text content from different levels of a document's organization such as words, sentences, special characters, and even the whole document. Linguistic features can be categorized based on lexical, syntactic, readability information [21]. The most common linguistic features that are used to represent textual information are lexical features that include word and character level features such as the total number of words and characters or the frequency of big words. Syntactic features are another widely met category of linguistic features that looks for certain syntactic grammar within the structure of a sentence. Some examples are the number of punctuation marks and the parts-of-speech (POS) tagging. Psycholinguistic (or psychological) features count the proportion of words that are correlated with various psychological processes and basic sentiment analysis. These features are usually extracted with Linguistic Inquiry and Word Count (LIWC) dictionaries that are basically large lexicons of word categories representing emotions, perceptual processes, etc. LIWC is also used for POS tagging. Finally, as an attempt to capture the complexity of the documents, readability metrics are utilized. Flesch-Kincaid,

Flesch Reading Ease and Gunning Fog are three different grade-level reading scores. A higher score corresponds to a document that takes a higher educational level in order to be read.

Variations of the above linguistic features have been used widely for many natural language processing tasks. In the early 2000s, the authors in [22][23][24] proposed three linguistic feature sets to detect deception in written narratives. In [22] the set used to distinguish deception from truth consists of 16 features (Feature Set 1) separated into four categories: Quantity, Grammatical Complexity, Vocabulary Complexity, and Specificity or Expressiveness. The results reached 60.72% accuracy with the C4.5 Decision Tree algorithm and 15-fold cross-validation which was considered satisfying given the small size of the dataset. The feature set used in [23] counts 29 linguistic cues (Feature Set 2) divided in three categories, namely Standard Linguistic Dimensions Psychological Processes and Relativity. The results obtained with Logistic Regression outperformed the human judge with 67% with respect to 52%. In [24] the authors performed statistical analysis for the evaluation of the linguistic features and ended up with a total of 27 (Feature Set 3) grouped in 9 categories. The three feature sets are displayed in the table below.

Table 2.1: Linguistic feature sets

Feature Set 1		Feature Set 2		Feature Set 3	
Category	**Description**	**Category**	**Description**	**Category**	**Description**
Quantity	# syllables	Standard linguistic dimensions	Word Count	Quantity	# Word
	# words		% words captured, dictionary words		# Verb
	# sentences		% words longer than si letters		# Noun phrase
Vocabulary Complexity	# big words		Total pronouns		# Sentences
	# syllables per wor		First-person singular	Complexity	Average number of clauses
Grammatical Complexity	# short sentences		Total first person		Average sentence length
	# long sentences		Total third person		Average word length
	Flesh Kincaid grad level		Negations		Average length of noun phrase
	avg # of words pe sentence		Articles		Pausality
	sentence complexity		Prepositions	Uncertainty	Modifiers
	number of conjunctions	Psychological processes	Affective or emotiona processes		# Modal verbs
Specificity and Expressiveness	emotiveness inde		Positive emotions		# Uncertainty
	rate of adjectives and adverbs		Negative emotions		# Other reference
	# affective terms		Cognitive processes	Nonimmediacy	Passive voice
			Causation		Objectification
			Insight		Generalizing terms
			Discrepancy		Self reference
			Tentative		Group reference
			Certainty	Expressivity	Emotiveness
			Sensory and perceptua processes	Diversity	Lexical diversity
			Social processes		Content word diversity
		Relativity	Space		Redundancy
			Inclusive	Informality	Spatio-temporal information
			Exclusive		Perceptual information
			Motion verbs	Affect	Positive affect
			Time		Negative affect
			Past tense verb		
			Present tense verb		
			Future tense verb		

Many research studies used variations of those three linguistic feature sets in different NLP domains. The authors in [25] worked on a fake news detection problem with three classes, Fake, Real or Satire. For their project the used an extensive collection of linguistic cues (63 in total) clustered into three broad categories: complexity, psychology and stylistic features. The machine-learning algorithm they used was SVM classifier with linear kernel on the Buzzfeed dataset and the score obtained was 78% accuracy with 5-fold cross-validation. The dataset was also enriched with satire articles.

Similarly, a recent research study aimed at combating the problem of fake news detection by utilizing a combination of state-of-the-art linguistic features enhanced with word embeddings. More specifically, the authors used the Word2Vec algorithm with pre-trained word vectors on the Google News corpus. The models were evaluated on BuzzFeed[2], PolitiFact[3], Kaggle-EXT[4] and McIntire[5] fake news datasets that have been widely used in previous studies. Furthermore, they created a larger and unbiased dataset that complies to the following rules:

- Each unreliable news article has been manually annotated by fact-checkers.
- Fake news items derive from various sources.
- Real articles have been issued by trustworthy reporting organizations.
- All articles originate from various domains.

The results were very promising with the Adaboost algorithm achieving over 95% accuracy in all datasets.

2.3 Fake news detection

2.3.1 Fake news definition

Fake news has gained sudden popularity around the US presidential election in 2016 and has become a synonym for false information. Initially, the term was used to reference false information disseminated through the news but it has evolved and given multiple interpretations throughout the years. The authors in [26] refer to fake news as "a news article that is intentionally and verifiably false and could mislead readers". According to [27] fake news is "information, presented as a news story that is factually incorrect and designed to deceive the

[2] https://github.com/BuzzFeedNews/2016-10-facebook-fact-check/tree/master/data
[3] https://www.cs.ucsb.edu/~william/data/liar_dataset.zip
[4] https://www.kaggle.com/mrisdal/fake-news
[5] https://opendatascience.com/how-to-build-a-fake-news-classification-model

consumer into believing it is true". Additionally, fake news is usually related to the terms of rumor and hoax. A rumor is an unsupported item of information, and hence its actual value is unresolved. The term hoax refers to a false story used to masquerade the truth [28]. The authors in [29] capture the broader scope of the term and propose a new definition for fake news as "A news article or message published and propagated through media, carrying false information regardless the means and motives behind it".

The authors in [1] make a more general categorization of false information based on its intent or knowledge content as shown in Figure 2.5. The first type can be also subdivided to misinformation or disinformation. Misinformation is the unintentionally spread of false information which can be a result of misrepresentation of an original piece of valid information due to cognitive biases or lack of attention. Disinformation is the creation and distribution of false information with the intention to deceive the audience.

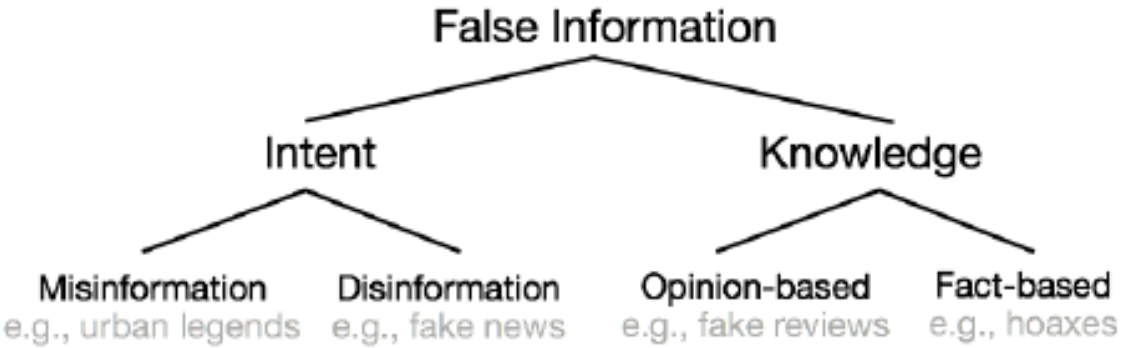

Figure 2.5 [1] categorization of false information

2.3.2 Fake news categories

According to the studies of [29][30][31] the categories of fake news can be organized as follows.

*Propaganda***.** Propaganda is defined as news content which is created by political parties to mislead public opinion. The underlying purpose is to profit a public figure, an organization or the government. The origins of the term propaganda come from World War II. This kind of fake news was used for political reasons aiming to harm a particular political party or a nation.

Satire or parody. Satire presents stories as news that might be factually incorrect, but the intent is not to deceive the consumer but instead to call out, expose or ridicule behavior and situations that are shameful, corrupt, or somehow "bad".

News fabrication. Fabrication refers to stories that have no factual basis but are published in the style of news articles to create legitimacy. The producer intends is to misinform the readers. Fabricated news is usually issued on websites, blogs or social media platforms.

Imposter content. An imposter is a type of fake news mimicking global news outlets to circulate false data. In other words, it impersonates genuine sources to mislead the audience. It can be detected by using the information of the source and author.

Manipulated content. Manipulated content refers to factual information or imagery that is manipulated to deceive. Image manipulation has become an increasingly common phenomenon with the advance of technology and the plethora of powerful media manipulation tools and is very difficult to distinguish with human eyes. The techniques range from the increase of color saturation and the removal of small elements to the insertion or exclusion of specific individuals in a photo.

*False connection***.** False connection refers to article descriptions such as headlines, visuals or captions that do not support its content.

False context. False context is a reliable content shared with false contextual information.

Conspiracy Theories. Conspiracy theory is the interpretation of a situation or an event that invokes conspiracy without any proved evidence. They often refer to illegal activities, political in motivation, which consequently can severely harm the audience and create anger and disbelief towards the government. Additionally, the use of unverified information as evidence, boost their credibility.

Advertising and public relations. It is the act or practice of advertising products and services within an official news statement with the purpose of financial profit.

2.3.3 Data for fake news detection

Documents and articles are raw data that need to be preprocessed to create a valuable dataset. Features are not only extracted from the content of the text but from other sources too. The data types for fake news detection are categorized below:

Source/promoters. Online resources and social media have enabled the spread of false information. Modified names of well-known web addresses aim to add credibility to misleading articles. Moreover, the constant creation of bot accounts reproducing the articles and forming communities of followers and followees accelerate the speed of the fake news distribution.

Information content. The content of the documents is the primary source of information for our classifier to distinguish our news as true or false. The documents consist of the title and the

main body. Longer and more capitalized words in the header and technical or high polarity words suggest fake articles [25]. Apart from the textual information, fake news articles include video, images, graphics or audio. The distorted content aims to misinform the viewers but computer vision and deep learning methods can be used to recognize abnormalities.

User responses. User responses can be beneficial in fake new detection since they are harder to manipulate in contrast to the content of the information that is crafted to avoid detection. User engagements in social media include shares, likes, comments or replies and contain valuable information for the detection such as textual features from the comments, the information flow of the news and temporal data from the timestamps. Moreover, user-based features like the number of followers and personal details can be extracted. Finally, sentiment analysis in user comments revealed that fake news is connected with replies with negative emotions [32].

2.3.4 Fake news detection approaches

Fake news diffusion patterns, especially on social media have been widely studied to identify the characteristics that differentiate deceitful news from real. Most existing works treat the problem as a binary classification task (fake/true, hoax/not hoax). Other studies choose a multi-class approach with different levels of veracity (true/mostly true/half true/false). The authors in [32] propose five methods for fake news detection.

Linguistic-based: The underlying basis of that method is to discover irregularities in the writing style and content of the articles that are associated with misleading information, such as clickbaits. Clickbaits are articles that use descriptions or headers which do not have any connection with the main body of the document with the purpose to entice the viewer to click on them and read the whole article.

This content-based detection employs linguistic and readability features to distinguish fake from real news by forming a set of linguistic cues. The features can be extracted from the whole text or it can be analyzed in smaller parts such as paragraphs, sentences or even words. The earliest studies focused on deception detection, include features such as the number of words, the lexical diversity, the average word length and the amount of passive and modal verbs. Moreover, Part-of-Speech tags are utilized to capture the linguistic characteristics of the text. Each word is assigned to a tag (noun, pronoun, adjective, etc.) according to its syntactic function. In bag-of-words text representation, every unique word or n-gram (a sequence of n continuous words) serves as a feature and is represented with Term Frequency-Inverse

Document Frequency (TF-IDF) value. TF-IDF is the frequency of each word or phrase in the corresponding document divided by the frequency in the whole collection of documents and indicates the importance of the word. Other linguistic features that are often examined for the detection of deceptive articles are the number of punctuation marks used.

Visual-based: Visual-based approach is based on features extracted from visual elements (i.e. image and video). Both visual and statistical features are utilized for the identification of distorted content. Visual features are obtained with machine learning techniques and include clarity and coherence score and tonal distribution histograms. Statistical features encompass count, image ratio, hot image ratio, etc.

User-based: Bots are the main contributors to the dissemination of fake news, thus analyzing user's characteristics and behavior could provide us with useful information to classify news. These characteristics can be divided into individual and group level. Individual-level features record personal information such as age and profile information as the number of followers and the number of tweets. The purpose is to determine the credibility of each user. Group level features mean to capture the typical behavior of the fake news spreaders.

Post-based: Social media is the primary medium for people to express their opinion about an article of questionable content, thus analyzing the users' responses is an effective way to detect fake news. The comments of each user can be represented with linguistic features or word embeddings to obtain the general opinion, the degree of reliability and the topic. Topic features can be generated by the Latent Dirichlet Allocation (LDA) technique [33]. Aggregating features from individual posts can capture group-level characteristics.

Network-based: Users that publish related posts on social media form networks in terms of interests and topics. This highlights the need to extract network-based features to represent the spread of fake news. Different forms of networks are stance, co-occurrence and friendship networks. Stance networks have as nodes the comments related to the news and as edges the weight between two documents. Co-occurrence networks evaluate if the users comment on articles that fall in the same category of news. Finally, friendship graphs that are constructed by forming connections between the followers indicate the dissemination of information.

2.3.5 Fake News Datasets

McIntire consists of 6,335 articles divided into fake and real labels. The dataset is constructed by randomly selecting a part (3,164) of Kaggle's fake news dataset[6] enhanced with 3,171 credible news articles. The real news items were gathered from All Sides[7], a website that is devoted to hosting articles from across the political spectrum and is considered to be a reliable source of information.

LIAR [34] The dataset contains 12,836 short news statements from 3,341 speakers over the past 10 years, collected from PolitiFact, a website that is devoted to checking the truthfulness of social media content by professional editors. The annotated labels include 6 different values namely, true, mostly-true, half-true, barely-true, false and pants-on-fire. The distribution of labels is well-balanced for the first 5 categories, varying from 2,063 to 2,638 instances, with the exception of the pants-of-fire label that consists of 1,050 statements. Moreover, the speakers cover a wide range of topics with the most discussed subjects being healthcare, economy, education, elections, immigration, etc. However, it is difficult to automatically distinguish between these 6 types, as the current studies achieved nearly 30% accuracy on this dataset [35].

Mihalcea The authors in [21] constructed two datasets, including fake news covering seven different domains. The first dataset, named FakeNewsAMT, contains 240, fake and real news respectively. The legitimate documents, belonging in six different types of news (sports, business, entertainment, politics, technology, and education) were obtained from a variety of websites such as the ABCNews, CNN, and FoxNews. The fake versions of the legitimate news pieces were achieved with the use of crowdsourcing via Amazon Mechanical Turk (AMT). Every item was manually checked to ensure its credibility.

The records for the second dataset were collected from the web and are focused on celebrities that are frequent targets of rumors and fake reports. The news sources are mainly online magazines and their validity was determined using gossip-checking sites such as "GossipCop.com". The dataset is referred to as Celebrity and is comprised of 250 news items for every label (fake vs real).

ISOT [36] This dataset was composed of real-world sources and the articles are separated into fake and non-fake. The truthful articles were gathered by crawling articles from Reuters[8] while the fake news articles were collected from various unreliable websites that were flagged by PolitiFact and Wikipedia. The dataset covers different topics, however, the majority of articles

[6] https://www.kaggle.com/mrisdal/fake-news
[7] https://www.allsides.com/unbiased-balanced-news
[8] https://www.reuters.com

focus on political and World news topics. The total size is 44.898 with 21.417 real news and 23.481 fake ones.

NEws LAndscape (NELA2017) [37] This dataset is a large collection of U.S. political news with over 136K articles from 92 sources, gathered between April 2017 and October 2017. Real news articles were acquired from well-established sources, according to Wikipedia lists. Opensources[9] was used to select sources that spread articles of deceptive content. Moreover, to ensure that both ends of the political spectrum (left and right) are equally represented Media Bias/Fact Check[10] was used, which is an independent online media outlet that provides occasional fact checks, mostly related to USA politics.

NELA-GT-2018 [38] This dataset is an extension of NELA2017 and includes 713K articles from 194 news and media sources collected between February 2018 and November 2018. The labels were assigned to every article according to 8 different assessment sites that encompass various levels of credibility such as reliability, bias, transparency, adherence to journalistic standards, and consumer trust.

[9] https://github.com/OpenSourcesGroup/opensources
[10] https://mediabiasfactcheck.com

3 Material and Methods

This chapter describes the methods implemented on the thesis for text preprocessing, feature extraction and finally the models for text classification.

3.1 Introduction

The goal of the thesis is to detect fake news by using raw text as input for the machine learning algorithms. As a second step, it is examined whether or not linguistic features can improve the accuracy of the model. We encounter this problem as a binary classification task that categorizes real and non-real news. The reason for this approach is that it is easier to distinguish between reliable and unreliable content than the multi-classification setting as it is precise and needs little interpretation. An overview of the model is illustrated in figure 3.1. First the input text is filtered so as to only keep the essential information from its content. For instance stop-words that exist in large quantities within each text but add little meaning, could act as noise for the model. Therefore, they are removed from the content. Then, the "cleaned" text is transformed into a vector representation that constitutes as 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 training set and evaluated on the unseen test set to check the generalization ability. The metrics used for the evaluation is the classification accuracy. Next, the feature vector is enhanced with linguistic features, the model is trained with the enhanced dataset and we compare the results.

The design and experiments of this model were deployed by using Python programming language, which is a general-purpose programming language that is regularly utilized by the data science research community, due to the active community and the vast selection of libraries and resources.

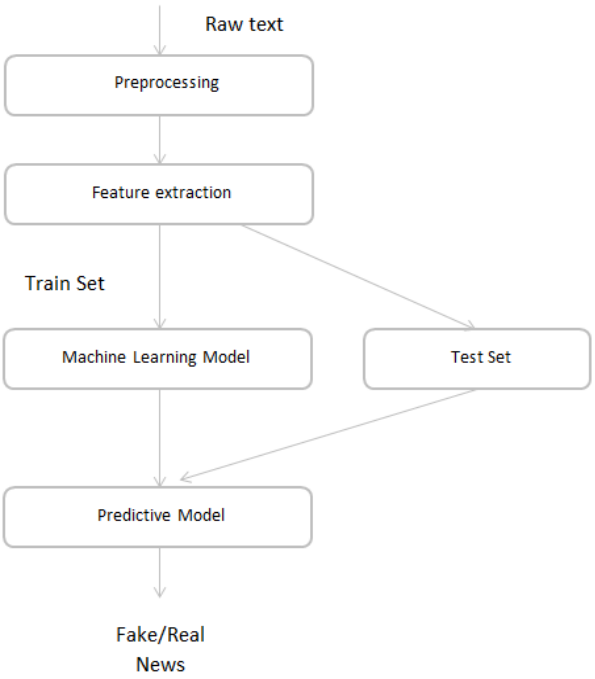

Figure 3.1 Overview of the model. After cleaning the text, it is converted into a feature vector. The training set is used to train the ML model and the test set to evaluate the models performance.

3.2 Dataset

The dataset used in this thesis is McIntire and it is described here. It consists of 6335 articles equally divided into fake and real news. Non-real articles were acquired from a fake news dataset in Kaggle[11] comprising of 13000 articles in 2016 during the US election. In order to determine which fake news articles to use, BS Detector[12] was utilized. BS Detector is a chrome extension tool that uses a curated list of unreliable or otherwise questionable sources annotating the articles with labels such as fake news, satire, extreme bias, conspiracy theory, etc. On the other hand, real news articles were collected from All Sides, a website that is devoted to hosting articles from across the political spectrum.

For every article the title and the main text are available. Both of them are utilized in the feature extraction process.

[11] https://www.kaggle.com
[12] https://gitlab.com/bs-detector/bs-detector

3.3 Preprocessing & Feature extraction

3.3.1 Text cleaning

Data scientists spend most of their time cleaning and preprocessing the data rather than modeling. Text is an unstructured form of data and could contain noisy content that could mislead our prediction model, thus a basic text cleaning is a necessary step to build more robust classification models. We used the Natural Language Toolkit (NLTK) and Regular Expressions (RE) Python libraries for the implementation of data preprocessing. The techniques for text cleaning utilized for this problem are presented as follows:

De-capitalization: Python is a case sensitive language. For example, it interprets "Dog" and "dog" as two different words. Hence every alphabetical character was converted to lowercase.

URL removal: Many articles contain external links to refer to their sources although they do not have any actual meaning. For that reason, strings containing "http" or "www" are removed from the text.

Contraction removal: The usage of an apostrophe is commonly met in textual data in order to substitute phrases such as "is not" with their shorter form "isn't". A vocabulary is used to replace such words with the original phrases.

Digits removal: Numbers indicate the date, time or are used as a scale to measure quantity and volume. However it is very difficult for machine learning algorithms to understand their meaning. Thus, it is decided to remove any numerical characters from the text.

Punctuation removal: Punctuation marks and non-alphabetic symbols, such as currency, do not add any significant information. Only dots are kept (.) as they act as the breaking point between sentences.

Tokenization: Tokenization is the process of splitting a sentence into a list of words (tokens). It is an essential part of the preprocessing phase as it converts unstructured text to a sequence of features. For example, the string "The food tastes so good" becomes "the", "food", "tastes", "so", "good".

There are two more techniques that are utilized for text cleaning but in our case we applied them only for specific features. Those are:

Stop-words removal: Stop-words are the words that are used very frequently and they somewhat lose their semantic meaning. Words like a, are, the, is, etc. are some examples of stop-words. The approach used to remove the stop-words was to have a predetermined list and filter them out from the tokenized sentences.

Stemming: In natural language processing, there might be a case when we want to recognize that the words “organize” and “organized” are just different forms of the same word. This is the idea of reducing different forms of a word to a core root. Stemming algorithms are a heuristic process that chops off the ends of words in hope of reducing them to their root (stem).

3.3.2 Feature extraction

Feature extraction is the process by which an initial set of raw data is transformed into more manageable groups for processing. It is a method for selecting or combining variables into features that describe accurately the original dataset. The final feature set, also named feature vector, is the input for the algorithm to predict the target variable. In this project there are two feature extraction techniques that are used in order to describe the textual dataset.

Linguistic features

Taking into consideration the related work presented in section 2, a linguistic-based cue of 49 features is extracted for further examination. These features are divided into eight broad categories and namely are:

Table 3.1 The extracted Linguistic Features

Category	Feature	Category	Feature
Standard linguisti feature	# words		% articles
	% dictionary words		% total pronouns
	# syllables		% first person pronouns
	# sentences		% third person pronouns
	# stop-words		% first person sing. pronouns
	% negations		% first person plr. pronouns
	# punctuation marks		% comparison words
	# words with all capitals in header		# conjuctions
Complexity features	# big words (>6 letters)		# modals
	% big words		# Wh-words
	avg word length	Expressivity	emotiveness index
	avg syllables per word		redundancy
	lexical diversity		rate adjectives & adverbs
	content word diversity		# modifiers
	# short sentences	Psychological	Rate of affective or emotional terms
	# long sentences		Rate of positive emotions
	avg sentence length		Rate of negative emotions
	avg clauses per sentence		Rate of cognitive terms
	avg noun phrase length	Relativity	present tense verbs
	pausality		past tense verbs
Syntactic features	# noun phrases	Readability	Flesh reading ease
	# verb phrases	Subjectivity	overall polarity
	% nouns		subjectivity
	% verbs	Reference	# quotes
	% prepositions		

Standard linguist features

These are typical variables that include character, word and sentence-level signals that measure the amount or frequency of items in the text. In order to determine the rate of negations, a predefined vocabulary with words of negative meaning (no, never, nothing) was utilized. Dictionary words are captured by using the Brown University Standard Corpus[13] of Present-Day American English (Brown Corpus) that was compiled by W. Nelson Francis and Henry Kučera and was first released in 1964.

Syntactic features

In the context of syntactic parsing, it is necessary to identify the parts of speech (nouns, verbs, etc.) in the text. The mechanism of annotating words in a sentence is called part-of-

[13] http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM

speech (POS) tagging and the Python's NTLK toolkit POS-tagger was used to keep track of the tag distribution in every article. This process is not straightforward, as a specific word may correspond to different parts of speech. The annotation is based on its context and definition within the sentence.

For instance, in the sentence "*we can go to the park today*" the result of POS tagging is:

[('we', 'PRP'), ('can', 'MD'), ('go', 'VB'), ('to', 'TO'), ('the', 'DT'), ('park', 'NN'), ('today', 'NN')]

The word "can" is assigned to Modal (MD) according to the context of the sentence, while in another case it could act as a Verb (VB).

Noun and verb phrases are formed with chunking. Chunking or shallow parsing is the process of extracting phrases (chunks) from unstructured text. The parsing is implemented based on rules. There are no pre-defined rules, but they are defined according to the requirements. Rules combine parts of speech with regular expressions. The rules for noun (NP) and verb phrases (VP) are determined as:

NP: {<DT>?<JJ.>*<NN.*>+}*

The chunk is any sequence beginning with an optional determiner, followed by zero or more adjectives, followed by at least one noun.

VP: {<MD>?<TO>?<VB.>+}*

Verb phrases are any sequence beginning with an optimal modal, followed with an optional "to" and one or more verbs of any type (i.e. "have to go").

The result is a syntax tree that can be displayed graphically.

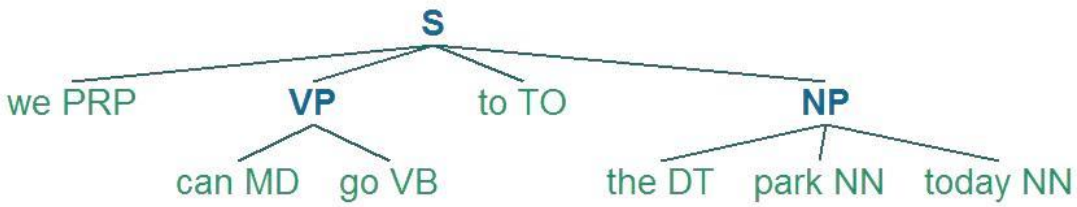

Figure 3.2 Syntax tree of the sentence

Complexity features

The complexity features depend on deeper NLP computations and aim to capture the overall complexity of a document. The articles are examined on two levels of intricacy, the word level, and the sentence level. Word-level features (number of big words, average syllables per words, etc.) intend to capture the average complexity of the words used by the writer. To measure the lexical diversity the following steps were used: after removing the stop-words, the remaining words were stemmed to their base form and then the number of unique words was divided by the total number of words within each document. The content word diversity is calculated in similar way. Content (or lexical) words according to [39] consist of nouns, main verbs, adjectives, and adverbs.

Sentence-level features (number of long sentences, avg clauses per sentence, etc.) are used to describe the average sentence structure. Pausality is a measure to count the number of punctuation marks per sentence.

Expressivity

$$Emotiveness = \frac{total\ \#\ of\ adjectives + total\ \#\ of\ adverbs}{total\ \#\ of\ nouns + total\ \#\ of\ verbs} \quad [24]$$

$$Redundancy = \frac{total\ \#\ function\ words}{total\ \#\ of\ sentences}, \quad [24]$$

where function words are words that express grammatical relationships between other words in a sentence. This includes articles, conjunctions, pronouns and question words that start with "wh" such as who, where, etc.

Modifiers are words that are used to modify, clarify or quantify another element in the structure. There are two parts of speech that are modifiers: adjectives and adverbs. This is a feature that is highly connected to the detection of different writing styles as many use modifiers to add more emphasis or detail to their sayings.

Psychological

The psychological features are word counts that are related to different psychological processes. In order to extract the features the Empath tool was used. Empath is a text analysis tool that covers a broad, pre-validated set of 200 emotional and topical categories [40]. It also allows the user to construct new categories by using a few seed terms. The process followed to build Empath was to train a deep learning skip-gram network on 1.8 billion words of fiction and extend it to word embeddings. The 200 pre-defined categories were seeded with words and cosine similarity was deployed on the vector space to find related terms for each category.

Finally, the categories' contents were filtered by humans. Figure 3.3 depicts the whole procedure:

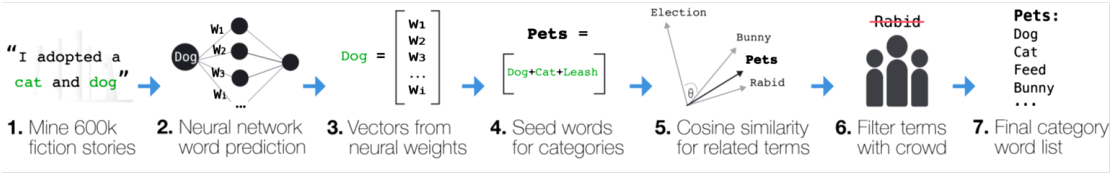

Figure 3.3 [40] Empath learns word embeddings from 1.8 billion words of fiction, uses seed terms and cosine similarity to define and discover new words for each of its categories, and finally filters its categories using crowds.

Relativity

This time-oriented category allows the writer to add validity to the information presented in his work. The use of multiple verb tenses allows him to jump around the timeline of the story, present facts and support his opinion.

Readability

Readability features indicate how difficult a passage in English is to understand. The authors in [21] have used several readability metrics in their effort to detect fake news. In this work, the Flesch reading ease (FRE) is applied. The formula for the FRE score is:

$$206.835 - 1.015\left(\frac{total\ words}{total\ sentences}\right) - 84.6\left(\frac{total\ syllables}{total\ words}\right)$$

The formula returns a score between 0 and 100 although, negative scores can be produced. Different scores represent different grade levels. A higher score indicates a low level of difficulty in understanding the content of the document, whereas a low score a high level [41].

Subjectivity

As explored in previous efforts [42], articles that intend to spread false information usually appose personal opinions and strong emotions. The subjectivity and the overall polarity (sentiment) of each article are tested with the TextBlob's API[14].

Reference

Fake news article's goal is to make absurd claims and usually do not have sound arguments to support their opinion. On the other hand, real articles provide quotes and facts to back up the information shared.

[14] https://textblob.readthedocs.io/en/dev/

Word embeddings - Word2Vec

An approach for word representation, which gained popularity in the recent years, is word embeddings. Word embeddings are a feature learning technique that assigns a set of words or phrases of a vocabulary to a vector space. A widely known approach for word embeddings representation is Word2Vec, which is included in the Gensim Python library[15]. It is an unsupervised method to construct embeddings by using contextual information integrated into a shallow, two-layer neural network. The input of the model is a large corpus, produced from the collection of documents, with every unique word in the corpus being mapped to a vector of several hundred dimensions. The main two architectures to produce embeddings are Continuous bag-of-words (CBOW) [43] and Skip-Gram [44].

The CBOW model architecture tries to predict a word (the center word) based on its context words. In other words, the neural network uses the word's context within a fixed-sliding window as input and it is trained to predict the center word as output. The simplest case of CBOW is if we consider a context window of one as demonstrated in figure 3.4. Given a corpus of size V the input and the output layer are both one-hot encoded of size [1 X V], with zero values except the cell that corresponds to the examined word. There are two sets of weights, one is between the input and the hidden layer and the other is between the hidden and the output layer. The input-hidden and the hidden-output layer matrices are respectively of size [V X N] and [N X V], where N is the arbitrary hyper-parameter denoting the units in the hidden layer.

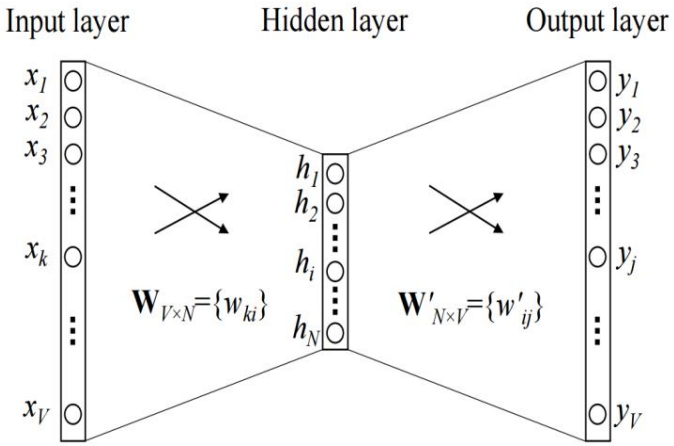

Figure 3.4: A simple CBOW model with only one word in the context [45]

[15] https://radimrehurek.com/gensim/models/word2vec.html

Skip-Gram follows the same topology as of CBOW but with the exact opposite direction. This model reverses the use of the target and context words. The input for the neural network is the current word and it is trained to predict the surrounding window of context words. Figure 3.5 depicts the two model architectures.

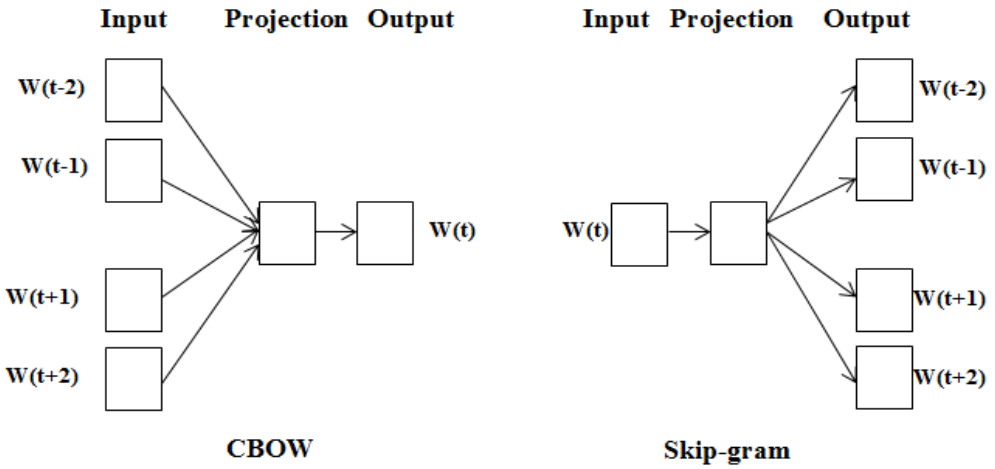

Figure 3.5 The CBOW architecture predicts the current word based on the context, while Skip-gram predicts surrounding words given the current word [43].

Apart from training word embeddings on a given dataset, pre-trained word embeddings can also be utilized by exploiting the concept of transfer learning explained in section 2.1.2. This technique is very powerful when training on small datasets as it reduces the number of learning parameters and avoids over-fitting issues. Moreover, it improves the overall model performance, as it speeds up the training phase without requiring heavy computational resources. In text classification tasks, pre-trained word embeddings form a language model that can generalize well. They are trained in large corpus with billions of words and can be used as a vector representation of text that captures general language aspects. A well-known, pre-trained embedding model that was also used in this project was trained on Google News corpus and it contains word vectors for a vocabulary of 3 million words[16]. This model was implemented on Word2Vec framework and includes word vectors with 300 hidden units.

In order to extend word embeddings to document-level embeddings, two approaches were applied: The first one was to aggregate the word vectors in each document by either calculating the average word vector or by firstly multiplying every word with its IDF score and

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

then averaging the whole document (weighted Word2Vec). The second approach is to merge the word vectors forming a matrix and feed 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.

3.4 Machine Learning Models

In this thesis there are two machine learning approaches employed for document classification, neural and traditional models. The traditional machine learning models examined are Support-Vector machine, Logistic Regression (LR), Random Forest (RF) and eXtreme Gradient Boosting (XGB) classifiers. Neural network models include Deep Neural Networks, Convolutional Neural Networks (CNN) and Gated Recurrent Unit (GRU) which is another type of Recurrent Neural Networks.

3.4.1 Traditional Models

SVM, LR and RF according to the literature are widely used for text classification and are further examined in section 2.1. XGB classifier is a scalable, tree boosting algorithm that is based on function approximation by optimizing specific loss functions [46]. It is an iterative technique that combines a set of weak learners (decision trees) and delivers improved prediction accuracy. At any time t, the model's weights are updated based on the outcomes of previous instant t-1. The outcomes predicted correctly from the first weak learner are given a lower weight and the ones miss-classified a higher one. Models are added sequentially until no more improvements can be made to the entire model.

3.4.2 Deep Neural Network Models

Neural networks are a specific set of algorithms that are inspired by biological neural networks and have shown great success in various applications such as text categorization. In this thesis, there are three deep learning architectures examined, deep neural networks (DNN), convolutional networks (CNN), and gated recurrent units (GRU). More details about the architecture of DNN and CNN can be found in the literature section.

GRUs, introduced by the authors in [47], is a variation of the RNN architecture that uses gating mechanisms to modulate the flow of information inside the cell. At each time step the cell is fed with the new input data along with the memory, or otherwise known as the hidden state. This structure allows the model to adaptively capture dependencies from sequential data. This is achieved through its gating units. Every cell contains two gates: the update gate and the

reset gate. The update gate decides whether the hidden state needs to be updated while the reset gate decides whether to ignore or not the previous hidden state. The diagram of a GRU cell is presented in figure 3.6.

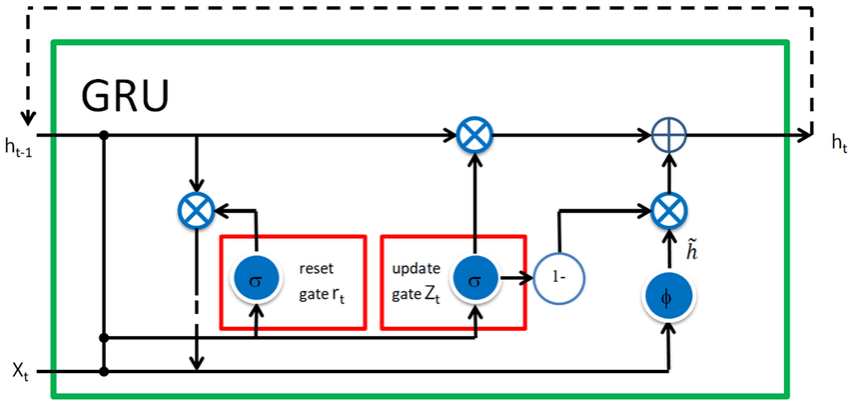

Figure 3.6 [48] internal structure of a GRU cell.

Where x_t is the input, h_t t and h_{t-1} are the hidden states and z_t and r_t denote the update and reset gates respectively.

The deep learning frameworks were implemented on Keras[17] Python library. Keras is a high-level neural networks API, written in Python language. More specifically, for the purpose of this thesis, the Keras functional API was utilized which allows us to define more complex models such as multi-input and multi-output models. In order for GRU and Convolutional models to be functional, Keras requires an embedding layer that transforms the input text to numerical values. Each word within the corpus is assigned to a unique integer and then each integer is mapped to the embedding vector space. However, the length of the documents in the dataset may differ from each other. To avoid this problem a special technique called padding is utilized. Firstly, we specify the maximum length of the documents that were previously converted into a vector of integers. The ones that have a smaller size are padded with zeros so as to persist the fixed length of the feature array. Then the numerical sequence is mapped to the values in the embedding space. The embedding vector is the input for the model. The output is fully connected to a dense layer to calculate the probability distribution of the classification task. Since it is a binary classification, the activation function is the sigmoid. The general structure of the model is displayed in figure 3.7.

[17] https://keras.io/

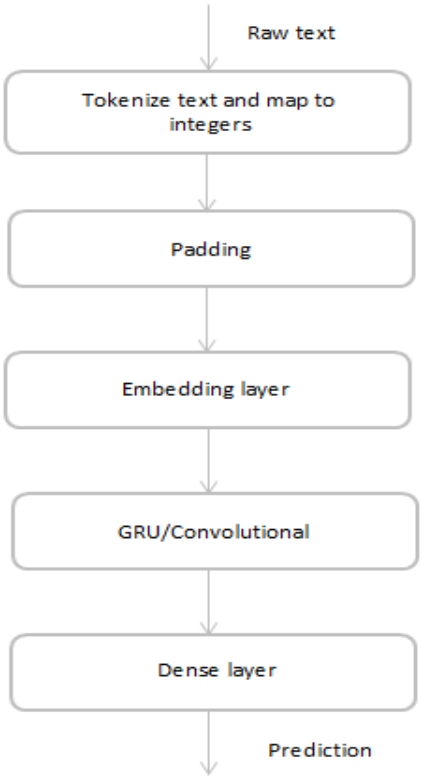

Figure 3.7 architecture of GRU/Convolutional models. Text is tokenized and the tokens are mapped to an integer. Each document is converted to a fixed size vector with zeros filling missing positions. Then through the embedding layer the text is passed to the model that is fully connected to a dense layer for the label prediction.

4 Experimental Results

The purpose of the thesis is to detect fake news, out of a large collection of documents from a content-based view. In this section, we describe the different experiments conducted in order to achieve the goal of the thesis. We compare different feature extraction techniques (Word2Vec, weighted Word2Vec, linguistic features) that convert each document into a vector representation, in combination with multiple machine learning algorithms (RF, XGB, CNN, etc.). For all the experiments the same data preprocessing technique was used. The title and the main body, before being mapped to a feature vector, were cleaned as described in section 3.3. After the preprocessing phase, the cleaned documents with less than 3 words were excluded from the dataset leaving a total of 6296 instances. Then, the experimentation process is divided into two main approaches according to the input required by the examined model.

Traditional models (Random Forest, Support Vector Machine, Logistic Regression and XGBoost) and Deep Neural Networks are trained separately with the datasets that derive from the different feature extraction techniques. Later, the Word2Vec (weighted or not) representations are enhanced with the linguistic features to test the claim that linguistic cues improve the performance of the model. On every trial, we randomly split the examined dataset into a training set (70%, 4408 instances) and test set (30%, 1888 instances). The metric to evaluate the distinct model is classification accuracy, which is the percentage of correctly classified documents. For the robust comparison of the results, we use a specific random seed which is a number that initializes a pseudorandom generator and subsequently the constructed dataset will be the same for each experiment.

CNN and GRU models are more complicated structures that take as input each word in the text. However, when dealing with large documents it will lead to high dimensionality. To resolve this problem we take into account the 20000 most frequent words in the corpus and keep up to 1000 words for each document. The ones with less than 1000 words are padded with zeros. Moreover, in order to train faster the models we used GPUs though this may prevent the exact reproducibility of our results. To overcome this, we repeated the experiments multiple times and the mean accuracy is returned. For the training set we used 5000 instances and for the test set the 1296 instances left.

4.1 Implementation

The experiments were implemented on the Kaggle Kernels which are offered by the Kaggle platform, a free of charge service. The kernels run in the browser and the processing power comes from servers in the cloud. That means less time for setting up a local environment. The user can choose between two displays, a Jupiter notebook or a script and also the desired language: Python, R or RMarkdown. The resources offered are up to 4 CPUs for multithread workloads, 16 GB of RAM and 1 GB of disk space. The execution time per session is 60 minutes. Kaggle grants access to GPUs as well (30 hours per week) and allows sharing publicly the content of the kernel for feedback and help, comprising a great environment to run machine learning code.

4.2 Experiments on traditional models

As mentioned above, traditional machine learning algorithms are firstly trained on the datasets constructed by using the different feature extraction techniques. Those namely are:

1. Word2Vec representations
2. weighted Word2Vec representation
3. linguistic cues

In order to map each word to the vector space we used the pre-trained embedding model which was trained on Google News corpus and the vector size was set to 300. To extend it from word to document-level representation the aggregated value of the vectors was calculated either by taking the mean value or by multiplying each vector by the IDF score of the corresponding word and then calculating the average. Linguistic cues include the 49 linguistic features from section 3.3.2.

In addition, the document representations are enhanced with the linguistic features to check whether they improve the accuracy of the models.

The experiments conducted using different combinations of datasets and algorithms are presented in table 4.1. SVM classifier is implemented with a linear kernel since it is observed to have very good results on text classification tasks.

Table 4.1 Accuracy of models in respect to the different datasets

Features	Model Accuracy (%)			
	XGB	XGB	SVM	LR
Word2Vec	86.55	81.15	86.82	84.44
Word2Vec weighted	85.12	80.62	86.29	87.08
Linguistic	80.94	78.14	72.15	72.00
Word2Vec + linguistic	**88.56**	**84.17**	84.70	86.02
Word2Vec weighted + linguistic	87.71	82.69	**87.08**	**88.30**

The results indicate that the combination of word representations and linguistic features achieved a higher accuracy with all classifiers. Moreover, it is observed that the linguistic dataset had the poorest performance when used alone, but the hypothesis that it would improve the overall accuracy of the model when used as an enhancement is confirmed. XGBoost classifier outperformed the rest of the methods with an accuracy score of 88.56% with the dataset created from the concatenation of linguistic cues and word2vec representations.

4.2.1 Linguistic Feature Importance Evaluation

Apart from using the linguistic feature set, we calculate the importance of each feature for every classifier to estimate the discriminatory power they have. It is clear that different features have different discriminatory powers depending on the model. Yet, we calculate the Mutual Information (MI) of each feature and we juxtapose the results. Mutual information is the measure that quantifies the dependence between two random variables. More specifically, it measures how much information of a random variable (target variable) is obtained by observing the other random variable. Mutual information has widely used in filter feature-selection methods to assess both the dependency and the redundancy of variables with respect to the target [49]. Table 4.2 summarizes the top 10 features based on their importance on each classifier and the results extracted from MI.

Table 4.2 Most important features for each classifier, and top 10 features according to Mutual Information technique

Order of importance	Most important features				
	RF	**XGB**	**SVM**	**LR**	**Mutual Information**
1	# sentences	# sentences	% 3rd person pr	% 3rd person pr	# big words
2	# quotes	# words	affective/emotional terms	affective/emotional terms	# words
3	# words	# quotes	% dict. words	% articles	# syllables
4	Flesh reading ease	# punctuation	Avg NP length	Avg NP length	# noun phrases
5	% 3rd person pr	# big words	subjectivity	subjectivity	# stop-words
6	# punctuation	% 3rd person pr	% articles	% dict. words	# modifiers
7	past tense verbs	Avg sent. length	lexical diversity	% comparison words	# punctuation
8	Avg sent. length	pausality	% comparison words	Rate of neg. emotions	# sentences
9	pausality	Flesh reading ease	content word diversity	avg word length	# verb phrases
10	# noun phrases	# syllables	avg word length	redundancy	past tense verbs

From the table we can clearly see that the different Linguistic features come from all the categories set in section 3.3.2. Moreover, the main features for Random Forest and XGBoost classifiers are almost the same. This happens because the models have similar architectures since both are based on decision trees. If we observe, the same thing appears for Logistic Regression and Support Vector Machine.

4.3 Experiments with neural networks

In addition to the previous experiments, we examine neural networks (NN) implemented on Keras library with the different combinations of feature sets. The linguistic features are only used as an enhancement as it did not show any promising results. In order to reduce memory usage, Keras provides a function that fits the machine learning model by iterating through batches over the training dataset. The weights and the parameters of the network are updated at the end of each iteration. A pass over all the training examples is called an epoch. We also use a technique called early stopping, which is a regularization method in neural networks that monitors the performance of the classifier in a validation set for each training epoch and interrupts the training process when the accuracy does not increase in the validation data for a pre-defined window. In other words, this method learns in which epoch to stop training in order to make the model more generalizable, because in further epochs the model starts overfitting. The results of the experiments are presented in table 4.3.

Table 4.3 Accuracy results of neural network classifier

Features	Accuracy (%)
Word2Vec	89.00
Word2Vec weighted	86.86
Word2Vec + Linguistic	**89.35**
Word2Vec weighted + Linguistic	88.84

The results show that linguistic features improve the performance of the model. The architecture used for all the experiments is the same including 3 hidden layers, each having 512 hidden units. The last hidden layer is connected to the output layer which calculates the probability distribution of the classification task. For the enhanced datasets, before undergoing the activation layer, the output of the third hidden layer is concatenated with the linguistic feature vectors. The two structures are shown in the figures below:

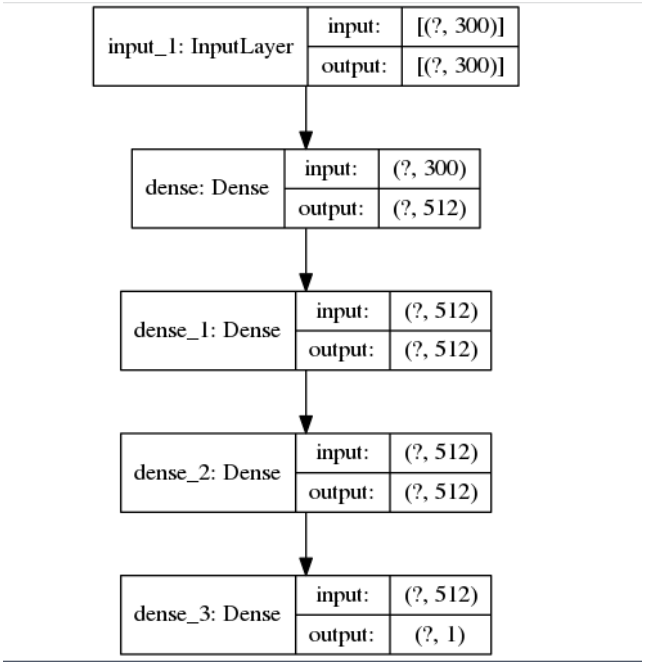

Figure 4.1 Architecture of Neural network model without linguistic features

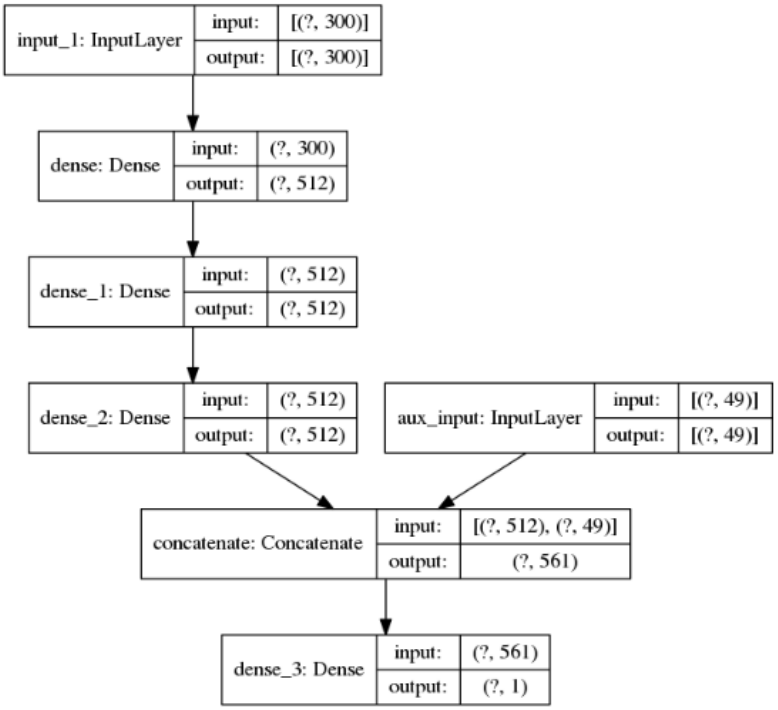

Figure 4.2 Architecture of Neural network model with linguistic features

4.4 Experiments with Convolutional Neural Networks and Gated Recurrent Unit

CNN and GRU take as input each word vector in the document as described in the beginning of chapter 4. The main idea behind that is that they "read" the documents as a sequence of words instead of vector representations of the whole document. We used the GRU model architecture based on the figure # with 100 units and relu as activation function. CNN was constructed accordingly with the following parameters:

1. filters = 100
2. kernel size = 2 (CNN considers the input as bigrams)
3. activation function = ReLu

However, when training the GRU and the CNN models, we observed that it was overfitting on the training set. To avoid this problem we add a dropout layer after the GRU and the max pooling layers respectively. This technique is a regularization method that randomly ignores (hence the name drop out) a set of neurons during the training phase. This way it prevents the network layers to co-adapt to correct mistakes from prior layers, in turn making the

model more robust. For the prediction on the enhanced datasets the drop out layer is placed after the concatenation of the two datasets. Table 4.4 summarizes the results.

Table 4.4 Accuracy results for GRU and CNN classifiers

Model	**Model accuracy (%)**	
	GRU	**CNN**
Word2Vec	91.38	93.62
Word2Vec + Linguistic	**92.70**	**94.39**

It is indicated that linguistic features improve the accuracy for both models. In order for the results to be comparable when using the two datasets, a basic architecture was maintained for the two models and any alternations on the layers structure were made only for the linguistic data.

Below (Figure 4.3) we demonstrate the architectures of the GRU models. Two dense layers with 100 units each are connected with the linguistic data before the concatenation.

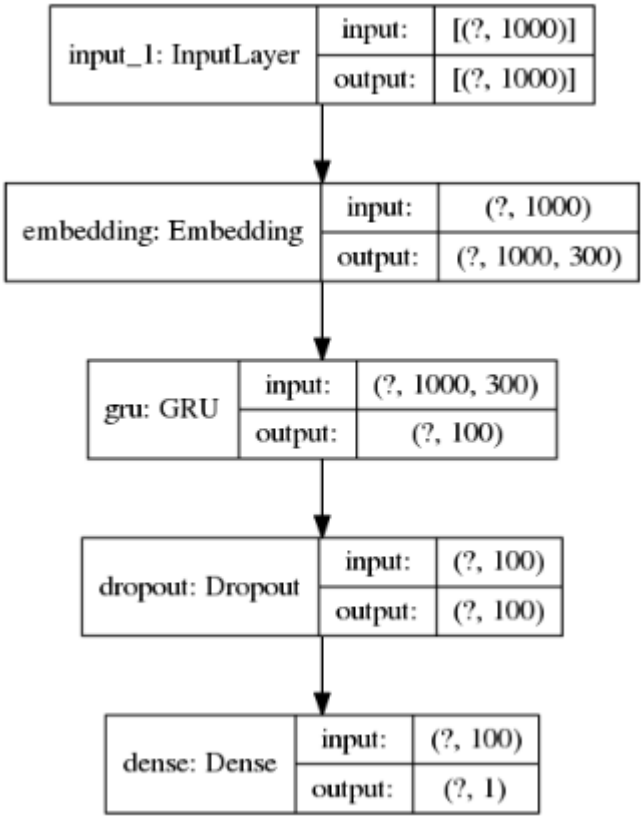

Figure 4.3 Architecture of GRU model without linguistic features

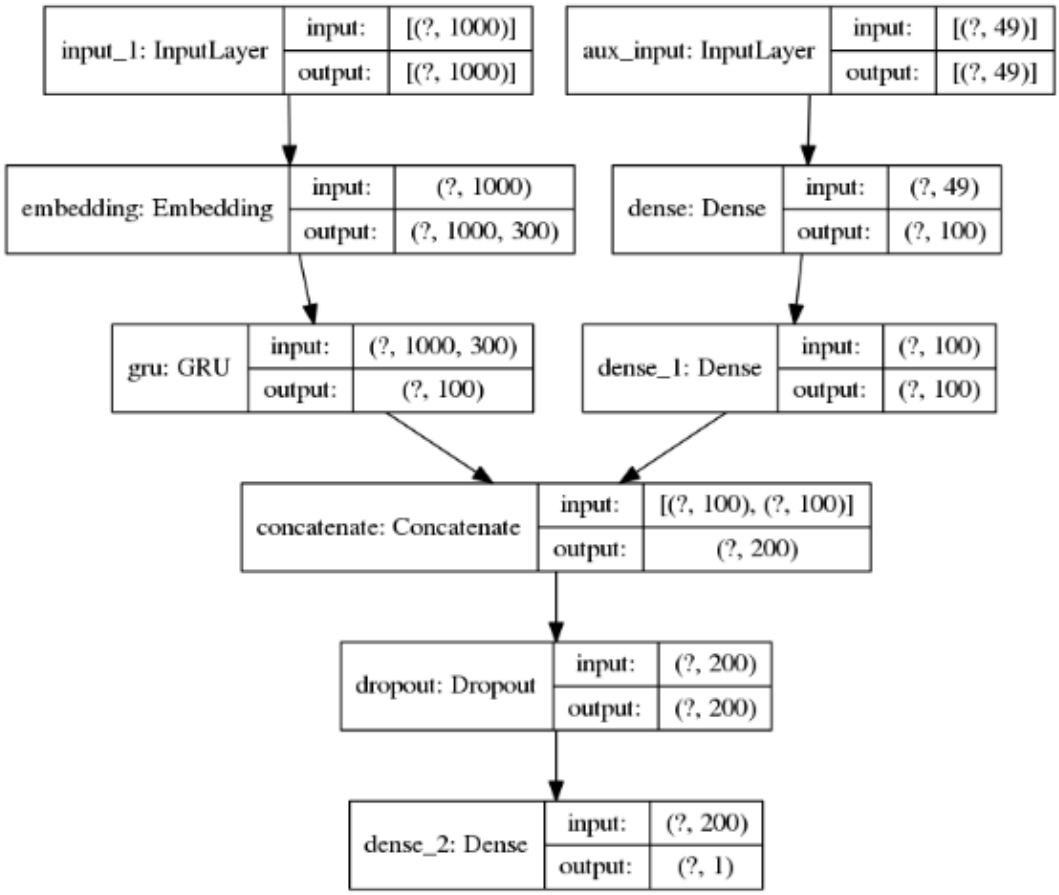

Figure 4.4 Architecture of GRU model enhanced with linguistic features

The results from CNN (94.39 %) with the enhanced feature set was obtained by adding 3 hidden layers of 100 units each to the linguistic input. The last layer is fully connected to the concatenation layer. The structures of the CNN models are illustrated in the following figures (4.5 and 4.6).

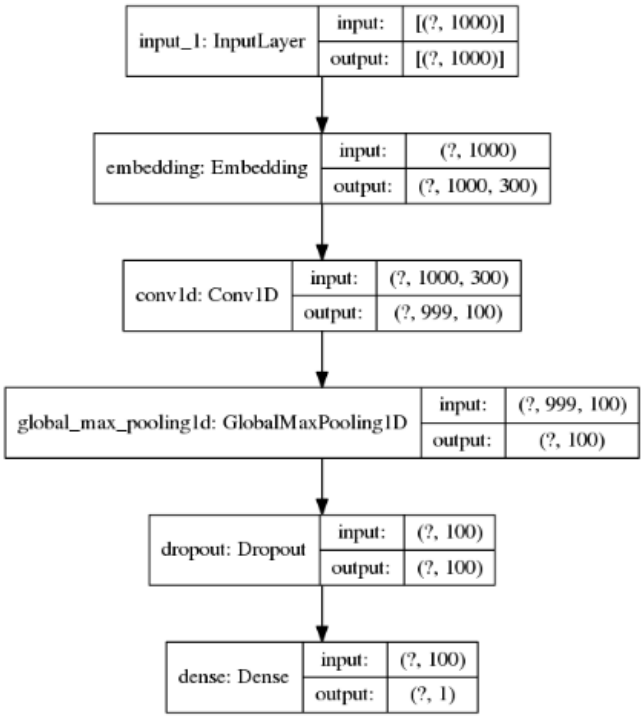

Figure 4.5 Architecture of CNN model enhanced with linguistic features

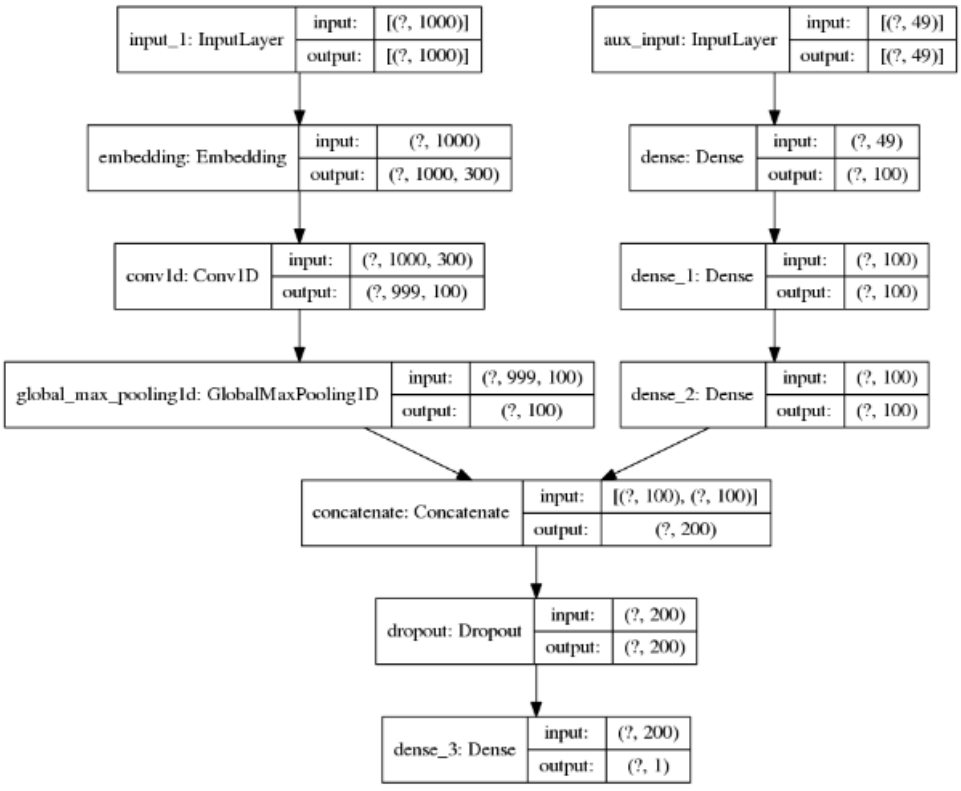

Figure 4.6 Architecture of CNN model enhanced with linguistic features

5 Discussion

The outcome presented in the previous section supports the findings in the literature about improving the performance of a fake news detection model by introducing linguistic features [6]. As anticipated, our experiments proved that linguistic cues, when used as an enhancement to word embeddings, can improve the accuracy of the model. In other words, extracting features from different levels of a document's organization (syllables, clauses, sentences), other than words, can boost the predictive ability.

Following the algorithm evaluation experiments, it is worth noticing that the assumption was confirmed using a variety of models with different architectures. Testing on various frameworks adds more validity to the results as it proves that they are not depending on the model examined. As expected, deep learning models that are used in many state-of-the-art approaches, outperformed the rest of the classifiers. Although, it should be mentioned that traditional learning algorithms did not undergo any specific tuning. Table 5.1 summarizes the results for each machine learning method on every dataset.

Table 5.1 Overview of the different models' accuracy

Features	**Model accuracy (%)**						
	XGB	**RF**	**SVM**	**LR**	**NN**	**GRU**	**CNN**
Word2Vec	86.55	81.15	86.82	84.44	89.00	91.38	93.62
Word2Vec weighted	85.12	80.62	86.29	87.08	86.86	-	-
Linguistic	80.94	78.14	72.15	72.00	-	-	-
Word2Vec + Linguistic	88.57	84.17	84.70	86.02	89.35	92.70	**94.39**
Word2Vec weighted + Linguistic	87.71	82.69	87.08	88.30	88.84	-	-

The experimental results indicate that detecting deception in written narratives by using only linguistic characteristics does not achieve a high accuracy score. However, when used in combination with word embeddings features the predictive performance increased over the seven models. It needs to be highlighted that Logistic Regression, which is a simple linear model, performed equally well (88.30 %) with other ensemble methods (XGBoost classifier). The results of the deep learning models were promising, as the accuracy scores improved significantly with the combination of the two feature extraction techniques. CNN performed the highest score of 94.39% which we believe happens because these models are trained on sequential data and they capture more complex patterns.

Regarding the importance of linguistic features, it varies depending on the classification model. As illustrated in table # the number of words, the number of sentences and the amount of punctuation marks are highly important for ensemble methods. This is also deduced from mutual information technique. For Support Vector Machine and Logistic Regression the rate of articles and the number of affective and emotional words are greatly associated with fake news detection. With a closer look we observe that the percentage of 3rd person plural pronouns is ranked in the top ten features for all models.

While the results were encouraging, we are aware that our research may have some limitations. First, in our experiments we used a relatively small dataset including articles only from the political spectrum. Second, there are some psycholinguistic features such as causation, certainty and social processes that were not included in the linguistic dataset. Finally, we only faced the problem from a content-based view without including metadata, network data from social media or multimedia data like video and images that can play a vital role in detecting fake news.

6 Conclusions and Future Work

In this thesis we encountered the problem of disinformation detection by implementing various text classification algorithms using supervised approaches. The main purpose of the study was to examine the prediction ability of the models by exploiting linguistic features. In addition, linguistic cues were used as an enhancement for word embeddings and the results prove that there was an increase to the accuracy for all tested models. The dataset used for our research was McIntire, a balanced dataset containing articles annotated as reliable and unreliable. We ran experiments on three different features sets derived from the different feature extraction techniques: w2v, weighted w2v and linguistic.

Deep learning models outperformed the rest of the classifiers which are based on less complex structures. The proposed framework that combines an enhanced dataset of word embeddings with linguistic cues trained on Convolutional Neural Networks resulted in 94,39% prediction accuracy while when using only word embeddings, it was 93.62%. Such results suggest that the use of linguistic features can improve existing deep learning-based fake news detection models and also provide some transparency on the most important linguistic characteristics.

Since the dataset used for our experiments is relatively small (6335 news articles), a possible future direction is to test the generalization ability of our models with larger datasets. Also, it would be preferable to contain information from multiple sources other than just politics. A potential improvement is to enlarge the linguistic dataset with psycholinguistic features that are correlated with various psychological processes and basic sentiment analysis. As a next step, we aim at using more transfer learning approaches for feature extraction, that could improve the accuracy of model decisions. Finally, we believe that employing several meta-data about the source and the author, user responses from social network platforms and multimedia data (images and video) could also support the diffusion of disinformation.

Bibliography

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

[2]. Newman, N., 2018. Journalism, media and technology trends and predictions 2018.

[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]. Zhou, X. and Zafarani, R., 2018. Fake news: A survey of research, detection methods, and opportunities. *arXiv preprint arXiv:1812.00315*.

[5]. Konstantinidis, I., 2019. Disinformation Detection with Model Explanations.

[6]. Gravanis, G., Vakali, A., Diamantaras, K. and Karadais, P., 2019. Behind the cues: A benchmarking study for fake news detection. *Expert Systems with Applications*, *128*, pp.201-213.

[7]. Sowmya, B.J. and Srinivasa, K.G., 2016, October. Large scale multi-label text classification of a hierarchical dataset using Rocchio algorithm. In *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)* (pp. 291-296). IEEE.

[8]. Korde, V. and Mahender, C.N., 2012. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, *3*(2), p.85.

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

[10]. Qu, Z., Song, X., Zheng, S., Wang, X., Song, X. and Li, Z., 2018, January. Improved Bayes method based on TF-IDF feature and grade factor feature for chinese information classification. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)* (pp. 677-680). IEEE.

[11]. McCallum, A. and Nigam, K., 1998, July. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization* (Vol. 752, No. 1, pp. 41-48).

[12]. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. and Watkins, C., 2002. Text classification using string kernels. *Journal of Machine Learning Research*, *2*(Feb), pp.419-444.

[13]. Weston, J. and Watkins, C., 1998. *Multi-class support vector machines* (pp. 98-04). Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, May.

[14]. Onan, A., Korukoğlu, S. and Bulut, H., 2016. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, *57*, pp.232-247.

[15]. Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D., 2019. Text classification algorithms: A survey. *Information*, *10*(4), p.150.

[16]. Kowsari, K., Brown, D.E., Heidarysafa, M., Meimandi, K.J., Gerber, M.S. and Barnes, L.E., 2017, December. Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 364-371). IEEE.

[17]. Sutskever, I., Martens, J. and Hinton, G.E., 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 1017-1024).

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

[19]. Pan, S.J. and Yang, Q., 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), pp.1345-1359.

[20]. Howard, J. and Ruder, S., 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

[21]. Pérez-Rosas, V., Kleinberg, B., Lefevre, A. and Mihalcea, R., 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

[22]. Burgoon, J.K., Blair, J.P., Qin, T. and Nunamaker, J.F., 2003, June. Detecting deception through linguistic analysis. In *International Conference on Intelligence and Security Informatics* (pp. 91-101). Springer, Berlin, Heidelberg.

[23]. Newman, M.L., Pennebaker, J.W., Berry, D.S. and Richards, J.M., 2003. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin*, *29*(5), pp.665-675.

[24]. Zhou, L., Burgoon, J.K., Nunamaker, J.F. and Twitchell, D., 2004. Automating linguistics-based cues for detecting deception in text-based asynchronous computer-mediated communications. *Group decision and negotiation*, *13*(1), pp.81-106.

[25]. Horne, B.D. and Adali, S., 2017, May. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Eleventh International AAAI Conference on Web and Social Media*.

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

[27]. Golbeck, J., Mauriello, M., Auxier, B., Bhanushali, K.H., Bonk, C., Bouzaghrane, M.A., Buntain, C., Chanduka, R., Cheakalos, P., Everett, J.B. and Falak, W., 2018, May. Fake news vs satire: A dataset and analysis. In *Proceedings of the 10th ACM Conference on Web Science* (pp. 17-21). ACM.

[28]. Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M. and Procter, R., 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, *51*(2), p.32.

[29]. Sharma, K., Qian, F., Jiang, H., Ruchansky, N., Zhang, M. and Liu, Y., 2019. Combating fake news: A survey on identification and mitigation techniques. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(3), p.21.

[30]. Zannettou, S., Sirivianos, M., Blackburn, J. and Kourtellis, N., 2019. The web of false information: Rumors, fake news, hoaxes, clickbait, and various other shenanigans. *Journal of Data and Information Quality (JDIQ)*, *11*(3), p.10.

[31]. 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.

[32]. 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.

[33]. 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.

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

[35]. Oshikawa, R., Qian, J. and Wang, W.Y., 2018. A survey on natural language processing for fake news detection. *arXiv preprint arXiv:1811.00770*.

[36]. Ahmed, H., Traore, I. and Saad, S., 2018. Detecting opinion spams and fake news using text classification. *Security and Privacy*, *1*(1), p.e9.

[37]. Horne, B.D., Khedr, S. and Adali, S., 2018, June. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *Twelfth International AAAI Conference on Web and Social Media*.

[38]. Nørregaard, J., Horne, B.D. and Adalı, S., 2019, July. NELA-GT-2018: A Large Multi-Labelled News Dataset for the Study of Misinformation in News Articles. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 13, No. 01, pp. 630-638).

[39]. Nerbonne, J., 2014. The secret life of pronouns. What our words say about us. *Literary and Linguistic Computing*, *29*(1), pp.139-142.

[40]. Fast, E., Chen, B. and Bernstein, M.S., 2016, May. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 4647-4657). ACM.

[41]. Why, O.M., 2017. *DUST-COVERED OPERATIONS AND MAINTENANCE MANUALS* (Doctoral dissertation, Worcester Polytechnic Institute).

[42]. Volkova, S., Shaffer, K., Jang, J.Y. and Hodas, N., 2017, July. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 647-653).

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

[44]. 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).

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

[46]. Chen, T. and Guestrin, C., 2016, August. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). ACM.

[47]. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[48]. Su, Y. and Kuo, C.C.J., 2019. On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*, *356*, pp.151-161.

[49]. Beraha, M., Metelli, A.M., Papini, M., Tirinzoni, A. and Restelli, M., 2019, July. Feature Selection via Mutual Information: New Theoretical Insights. In *2019 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-9). IEEE.