UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

"Automatic Fake News Detection in Online Media"

A thesis presented for the degree of:
Maestría en Ingeniería Eléctrica
(Instrumentación y Sistemas Digitales)

By:

Ing. Luis Miguel López Santamaría

Thesis Directors:

Dr. Juan Carlos Gómez Carranza
Dr. Saskia Van Amerongen

Salamanca, Guanajuato January 2022

UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

"Detección Automática de Noticias Falsas en Medios en Línea"

Para obtener el título de:
Maestría en Ingeniería Eléctrica
(Instrumentación y Sistemas Digitales)

Presenta:

Ing. Luis Miguel López Santamaría

Directores de Tesis:

Dr. Juan Carlos Gómez Carranza
Dra. Saskia Van Amerongen

Salamanca, Guanajuato

Enero 2022

Abstract

Nowadays, online media has become one of the principal sources of news consumption. People and news organizations use online media such as news websites and social media to stay informed and distribute information. These sites offer different advantages such as reduced costs, adaptability, easy access, and quick distribution of information. Nevertheless, the extensive dissemination of information on these sites has led to the existence of fake news. Fake news contains intentionally fabricated false information or alterations of real events. This type of news aims generate biased ideas and beliefs in society. To stop the spread of fake news for the benefit of society and new organizations, there is a current trend to develop systems that automatically detect them, since doing a manual fact-checking is practically impossible. In this thesis, we present a study for the problem of automated fake news detection in online media using the textual content from the news. We collected different datasets that contain news extracted from variety of news websites and social networks to solve this task. The datasets we collected are: COVID Fake News, LIAR, FakeNewsNet, ISOT, The Fake News Corpus Spanish and Fake Costa Rica News. In addition, the verification of the veracity of the news in the datasets was in charge of organizations such as PolitiFact, Gossip Cop, Verificado, etc. With these datasets, we conducted a series of experiments with different machine learning and deep learning models using a set of superficial and deep features extracted from the text. To evaluate our models, we use a set of metrics to measure their performance.

Resumen

Hoy en día, los medios digitales se han convertido en una de las principales fuentes de consumo de noticias. Las personas y las organizaciones de noticias utilizan los medios digitales, como los sitios web de noticias y redes sociales para mantenerse informados y distribuir información. Estos sitios ofrecen diferentes ventajas como costos reducidos, adaptabilidad, fácil acceso y rápida distribución de la información. Sin embargo, la amplia difusión de la información en estos sitios ha dado lugar a la existencia de noticias falsas. Las noticias falsas contienen información falsa fabricada intencionalmente o alteraciones de hechos reales. Este tipo de noticias tiene como objetivo generar ideas y creencias sesgadas en la sociedad. Para frenar la difusión de noticias falsas en beneficio de la sociedad y de las nuevas organizaciones, existe una tendencia actual a desarrollar sistemas que las detecten automáticamente, ya que hacer una verificación manual es prácticamente imposible. En esta tesis, se presenta un estudio para el problema de la detección automática de noticias falsas en medios digitales utilizando el contenido textual de las noticias. Se recopilaron diferentes conjuntos de datos que contienen noticias extraídas de varios sitios web de noticias y redes sociales para resolver esta tarea. Los conjuntos de datos que recopilamos son: COVID Fake News, FakeNewsNet, ISOT, The Fake News Corpus Spanish y Fake Costa Rica News. Además, la verificación de la veracidad de las noticias en los cojuntos de datos estuvo a cargo de organizaciones como PolitiFact, Gossip Cop, Verificado, etc. Con estos conjuntos de datos, se realizaron una serie de experimentos con diferentes modelos de aprendizaje de máquina y aprendizaje profundo utilizando un conjunto de características superficiales y profundas extraídas del texto. Para evaluar los modelos, se utilizaron un conjunto de métricas para medir su desempeño.

Dedication

To my mother Ricarda Santamaría Flores and my father Miguel Angel López, for all the love and support, they have given me in my life.

Acknowledgements

First, I want to thank God for always being by my side and guiding me on the right path. I want to thank my Padrino Luisito who has always been with me at all times unconditionally, thanks for always taking care of me and for helping me achieve my goals, without your help I wouldn't have reached where I am. I also want to thank my family for being my support during this journey. I want to thank my mom, Ricarda Santamaría Flores, thank you mom, for always being by my side. Thanks for the advice, the talks, the scolding, but above all, for all the love you give me. I will always be your "cosita hermosa", I love you, mom. I want to thank my dad, Miguel Angel López, thanks for each one of your advice and jokes. Thanks for supporting me with my mom in each one of my dreams, I love you dad. To my brother, Miguel Angel López Santamaría, thank you for always being there and brightening my days with your jokes. I also thank Mercedes Morales Platas who is like a second mother to me, thank you for always taking care of me mamá Meche.

Thanks, Dr. Juan Carlos Gómez Carranza. Doc, thank you for all the things that you teach me, but specially for all the talks, advice and scolding. Even though sometimes I'm a bit foolish, you always motivate me to do things right. You more than a teacher are a friend whom I appreciate very much.

Thanks to life for this achievement and all the people who supported me and believed in the accomplishment of this thesis.

Institutional Acknowledgements

Thanks to the University of Guanajuato and CONACYT.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, the number of people using the internet to stay informed of the latest news has grown exponentially. It is estimated that in 2020, around 86% of adults in the U.S. used the internet to stay informed [1]. Online media such as news websites (e.g., CNN, El Universal, Fox News, etc.) or social media (e.g., Facebook, Twitter, YouTube, etc.) are the main websites people use to consume news. In 2020, news websites or apps were used by 68% of U.S. adults [2].

Recently, news organizations and individuals have preferred to use online media over other traditional media (e.g., T.V. or radio) for different advantages, such as low cost of publication and rapid dissemination of information. Nevertheless, online media is seen as a double-edged sword. Despite the versatility offered by these sites, they have also become a channel for misinformation, which has led to the spread of news denominated *fake news*. Fake news can be defined as news that contains information intentionally fabricated or alterations to real events. The main objective of this type of news is to persuade society for different purposes, such as generating a political influence, damaging the reputation of an organization or a person, financial purposes, among others. Fake news can spread quickly among users thanks to the rapid diffusion of the information offered by news media and social media. In some cases, it can spread faster than real news causing a negative impact on society and news organizations [1].

In this direction, to mitigate the damage caused by fake news to the news ecosystem and society, it would be necessary to develop methods based on machine learning and deep learning algorithms to identify fake news automatically since the manual checking of news is impossible. In recent years, the automatic identification of fake news in online media caught the attention of researchers since it poses several new and interesting research problems. Nevertheless, different issues made this task complex and unique. For example, the content of fake news is diverse in terms of the topics covered in it. Similarly, fake news makes use of different linguistic styles to distort the truth. In addition, some fake news uses factual information cited in the wrong context to mislead consumers [2].

In this work, we present a study that addresses the problem of fake news identification in online media using the textual content from the news. We mainly focus on sites with easier access and a great abundance of fake news, such as news pages and social networks.

[1] https://pewrsr.ch/383BbNz

[2] https://pewrsr.ch/3eBhmQV

For this research, we decided to collect different datasets to conduct several experiments for the task. The datasets we collected are: COVID Fake News [3], LIAR [3], FakeNewsNet [4], ISOT [5], The Spanish Fake News Corpus [6] and Fake Costa Rica News [4]. These datasets are conformed with news from different news sources, such as CNN, El Universal, Fox News, CBS, etc. Likewise, some datasets have news from different social networks, such as Facebook and Twitter. The truthfulness of the news in these datasets was in charge of different organizations, such as PolitiFact, Gossip Cop, Verificado, among others. The majority of the datasets collected are conformed with news in English except for The Spanish Fake News Corpus and Fake Costa Rica News that are in Spanish. We conducted several experiments with these datasets using different superficial and deep features extracted from the textual content from the news that lately were combined with different machine learning and deep learning models for automatic fake news detection.

1.1 Motivation

The internet has become one of the principal communication channels among people and news organizations. It is estimated that in 2020, approximately 4.5 billion people were active internet users [5]. Many of these users use online media such as social networks, microblogs, news pages, etc., as primary sources for news consumption. For example, in 2020, 20% of the adult population who represents active users on the internet in the U.S. consumes news through news sites such as CNN.com or Yahoo News [6]. Twitter, which is one of the most popular social networks in this country, it is used by 17% of the adult population to consume news [7]. In Mexico, it is estimated that around 20% of the active internet users consume news from sites such as Aristegui Noticias or El Universal online [8]. Thus, gradually, online media is becoming one of the leading sources for news spreading.

On the other hand, this widespread information has led to the existence of fake news. Fake news has become a global issue that causes negative repercussions on society and the news industry. This type of news can generate disinformation and false ideas among people. For example, during the 2016 U.S. presidential election, a massive wave of fake news spread in online media, creating biased ideas among voters [7]. Likewise, due to the COVID-19 pandemic, a considerable quantity of fake news was spread among users and news organizations, creating speculation if the pandemic was real or not or if the vaccines were dangerous or not. The credibility of the news industry is also affected by fake news. Since 2017, U.S. news companies such as CBS and NBC have seen reduced credibility from consumers due to the boom of fake news [9].

In this direction, the identification of fake news has become an essential task that benefits society and news organizations but also the governments that have been equally affected by fake news. Moreover, identifying fake news would provide

[3]https://bit.ly/3zB1XZZ
[4]https://bit.ly/3wduNx6
[5]https://bit.ly/3xHJY2r
[6]https://bit.ly/3hALLAO
[7]https://bit.ly/3khSdyk
[8]https://bit.ly/3B4vL1t
[9]https://bit.ly/3ePxDSB

a healthier news ecosystem. For example, consumers could access the news to help them create their own criteria about a specific topic without fear that these are built on false information. In addition, the news industry would be benefited since the content of the news would have a higher level of veracity, and the news companies focused on creating fake news would have less penetration among consumers.

The identification of fake news must be made automatically because the large amount of information and rapid dissemination that it has on the internet makes a manual review impossible. On the other hand, the problem of automatically identifying fake news is far from being solved, despite the investigations that have been done.

1.2 Objectives

The main objective of this thesis is to build different machine learning and deep learning models to automatically detect fake news generated in online media. Using different superficial and deep learning features extracted from the textual content, and measure their performance for the task.

The specific objectives of the automatic fake news detection in online media are:

- Collect different datasets.
- Statistically describe the datasets.
- Extract a series of superficial features such as words, emojis, hashtags, abbreviations, ats and links.
- Extract a series of deep features such as GloVe, Word2Vec and fastText.
- Use the textual content to create machine learning models such as Logistic Regression, Support Vector Machines, k-Nearest Neighbors, etc., and deep learning models such as Long-Short Term-Memory networks, Bidirectional Encoder Representations from Transformers, etc.
- Experiment with the built machine learning and deep learning models, and evaluate them using accuracy, F1 macro and AUC-ROC as performance metrics.

1.3 Literature Review

The quick dissemination of information through the internet has resulted in an exponential growth of fake news. Therefore, automatic fake news identification on online media is a task that has become relevant in recent years. In this task, the main objective is to predict if a piece of news is true or false based on its content features [8].

Researchers have used different approaches to tackle the problem of the identification of fake news. For example, in [9] the authors used the number of grammatical errors found in the fake news and combined it with a Naïve Bayes classifier, obtained results of 0.75 in accuracy. In [10, 11] they used the textual content extracted from the news and compared with different machine learning models, being

Support Vector Machines, the classifier with the highest accuracy of 0.92. In [12] the authors used an n-gram approach for representing different contexts from the news, as well as to generate features to classify it with different machine learning models. The model with the best results was a Support Vector Machines with a linear kernel which obtained 0.92 on accuracy. The authors in [13] used pre-trained GloVe word embeddings as weights in the embedding layer of deep learning models such as Recurrent Neural Networks (RNN) and Covolutional Neural Networks (CNN). A Bi-directional Long Short-Term Memory (Bi-LSTM) archived the best result, obtaining 0.98 for accuracy. Similarly, in [14] the authors used pre-trained GloVe vectors but proposed a deep CNN model that used multiple hidden layers to learn the discriminatory features from the fake news. The best result was obtained by combining the proposed CNN model and GloVe with a 0.98 on accuracy.

On the other hand, some studies have used textual content and visual content extracted to identify the fake news. In [15] the authors used a combination of text and images from the news. They used a CNN to extract informative features from the text and a pre-trained model VGG19 [16] to extract the visual features from the images. The results obtained in this research were 0.82 for accuracy, 0.84 for precision and 0.82 for F1 macro.

Recently, one of the forums where the problem of fake news is studied is the MEX-A3T. This forum is part of IberLEF (Iberian Languages Evaluation Forum). In the last edition, the organizers introduced the task of identifying fake news in Spanish [17]. The participants used different approaches to solve this task. For example, in [18] the authors used a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model with Spanish encodings and combined with words and char n-grams vectors obtaining 0.85 in F1 macro and 0.85 in accuracy [19]. The authors in [20] used a combination of binary text representation in conjunction with a bag-of-words model. These features were combined with a Support Vector Machines and obtained results of 0.81 in accuracy and 0.80 in F1 macro. In [21], the authors used an approach using text representation with *tf-idf* and combined with different machine learning models, being Support Vector Machine the model with the best results of 0.81 in F1 macro and 0.81 in accuracy.

Despite the variety of approaches in the field of fake news identification, further development and research in this area is necessary to create more robust systems to filter out this type of news, considering that this problem is growing exponentially, affecting society and news organizations, and it is far from be solved.

Chapter 2

Theoretical Framework

In this chapter, we describe the techniques and methods used in our research. In the first part, we present the methods we used to process the news's textual content. Thereafter, we describe the operation of the machine learning and deep learning models used in this work. Lastly, we present the metrics used to evaluate the performance of the machine learning and deep learning models selected for this work.

2.1 Pre-processing Data Methods

Data can come from different sources such as sensors, social media, storage records, etc. Nevertheless, because most of the systems that gather data are often poorly controlled, this gives as a result that the data present out-of-range values, missing values, noisy values, etc. In this direction, pre-processing data is one of the essential steps to prevent errors or biased results during the experimentation phase.

2.1.1 Cleaning Process

Nowadays, we can gather data from different internet sources such as social media, news web pages, microblogs, etc. The data found on these sites is in various forms, for example, images, text, video, audio, etc. Nevertheless, one of the problems presented during the gathering process is that the data come with different inconsistencies that may affect another process. For example, missing or out-of-range values can cause longer execution times or inaccurate results. Additionally, some organizations and companies may be affected by unclean data in the decision-making process.

In this direction, the data cleaning process aims to smooth noisy data, identifying or removing *dirty* data that can cause confusion in any model that requires correct data pre-processing. Different tools can help standardize data properly in this process, such as scripts, programs, specialized libraries, etc. In Figure 2.1 we present a general idea of the concept of the data cleaning process. We can observe on the left side we have two pieces of news in which we find words in English and Spanish combined with other elements such as links, punctuation marks, and upper case letters. On the right side of the figure, we see the result of this process where we observe only the words kept and converted to lowercase.

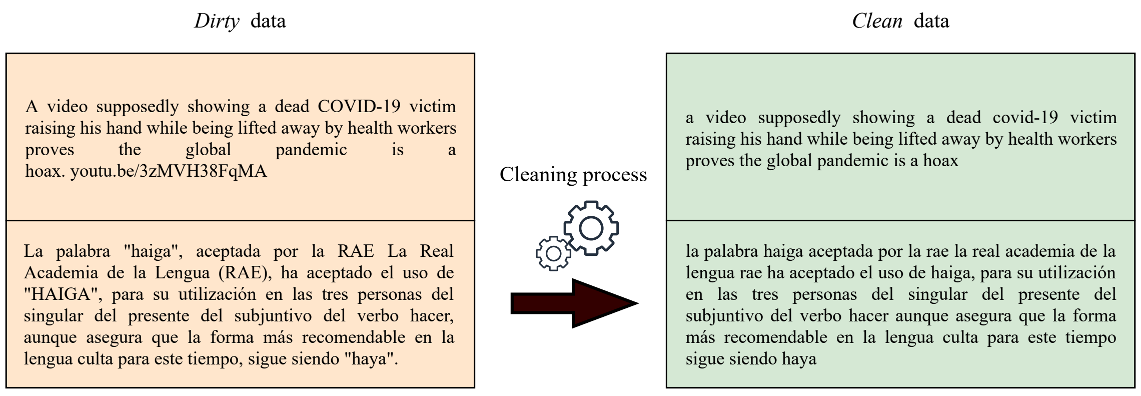

Dirty data
A video supposedly showing a dead COVID-19 victim raising his hand while being lifted away by health workers proves the global pandemic is a hoax. youtu.be/3zMVH38FqMA
La palabra "haiga", aceptada por la RAE La Real Academia de la Lengua (RAE), ha aceptado el uso de "HAIGA", para su utilización en las tres personas del singular del presente del subjuntivo del verbo hacer, aunque asegura que la forma más recomendable en la lengua culta para este tiempo, sigue siendo "haya".
Cleaning process
Clean data
a video supposedly showing a dead covid-19 victim raising his hand while being lifted away by health workers proves the global pandemic is a hoax
la palabra haiga aceptada por la rae la real academia de la lengua rae ha aceptado el uso de haiga, para su utilización en las tres personas del singular del presente del subjuntivo del verbo hacer aunque asegura que la forma más recomendable en la lengua culta para este tiempo sigue siendo haya

Figure 2.1: Representation of the cleaning process of text data.

In this work, we focused on the textual content in which we can apply several techniques. One of these techniques is the *tokenization* that consists in breaking a piece of text into *tokens*. Tokens could be phrases, words, symbols or another character sequence that has a meaningful structure. For example, we can perform this technique separating the sentences in the text by white space. The result of this process will be the extraction of only the words from the sentences. Likewise, we can use different separators such as punctuation marks, line breaks, tabs, among others. Regular expressions, also known as RegEx, are another way to perform the process of tokenization. Regular expressions are a set of encoded strings used to match patterns in the text strings [22]. There is a variety of regular expressions that can help us to extract different features from the text. For example, we can create a regular expression to match only the links in a text. Another technique used in the cleaning process consists of convert all the letters to lowercase. Nevertheless, this process can cause problems as some words have different meanings when they start with capital letters or capitals.

In the cleaning process of the textual content, we may face other problems where tokenization or regular expressions are insufficient to clean the data. In most languages, there is a set of words more commonly used than others. These words are known as *stop words* [23]. In English, for example, words such as “the” or “an” appear more frequently in sentences. In Spanish, we can also find examples of stop words such as “de” or “que”. Thus, removing these types of words may not affect the meaning of phrases or sentences. To perform this technique, it is required to have a list of stop words available based on the language in question. Some tools facilitate the task of removing stop words from the text. For example, the NTLK (Natural Language Toolkit) library [24] contains lists of stop words for different languages and makes this task easier.

2.1.2 Transformation Process

The data transformation is an important step when working with machine learning and deep learning models. In these models, the algorithms work with numbers, specifically with matrices of numbers. For this reason, data in a raw format cannot be used as input. In this work, we focus on textual content that is a sequence of characters. Therefore, it is necessary to convert the data into a suitable form.

tf-idf

In the transformation data process, one of the most popular techniques is *term-frequency-inverse-document-frequency* (*tf-idf*). This method measures the importance of a word for a document in a collection of documents. This technique is defined by Equation 2.1.

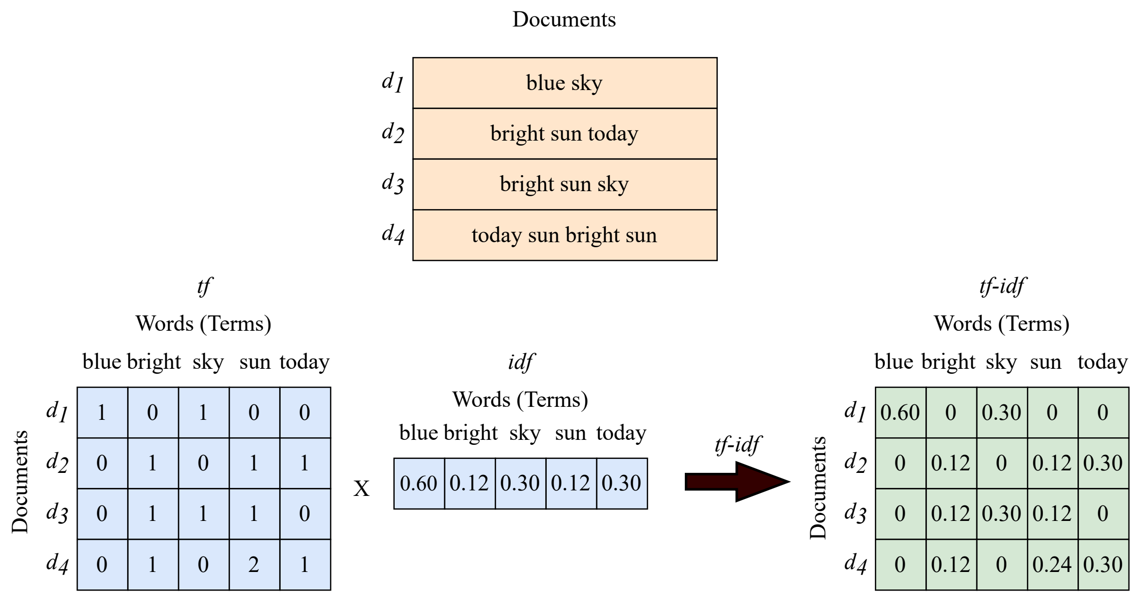
$$\textit{tf-idf}_{t,d} = tf_{t,d} \times idf_t \tag{2.1}$$

Equation 2.1 consists of two steps, the first one, $tf_{t,d}$ (*term-frequency*), corresponds to the frequency of a term t in a document d. An easy way to perform this step is to count the number of times a word occurs in a document. Nevertheless, using a raw count can present some problems, since some documents vary considerably in length. In this case, an adjustment used is a term weighting that is proportional to the term frequency [25].

The second step consists of idf_t (*inverse-document-frequency*) which is defined by Equation 2.2 where N corresponds to the total number of documents and df_t is the number of documents where the term t appears.

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

The result of this process is a term-document matrix. In Figure 2.2 we can see a schematic representation of the *tf-idf* process. In this figure, we calculate the tf and the *idf* (both in blue) for a set of documents (in orange). After multiplying these terms, we obtain the term-document matrix (in green) where each cell has a *tf-idf* value for each term of each document. The x-axis represents each document, while the y-axis represents each term.



	Documents
d_1	blue sky
d_2	bright sun today
d_3	bright sun sky
d_4	today sun bright sun

tf	blue	bright	sky	sun	today
d_1	1	0	1	0	0
d_2	0	1	0	1	1
d_3	0	1	1	1	0
d_4	0	1	0	2	1

blue	bright	sky	sun	today
0.60	0.12	0.30	0.12	0.30

tf-idf	blue	bright	sky	sun	today
d_1	0.60	0	0.30	0	0
d_2	0	0.12	0	0.12	0.30
d_3	0	0.12	0.30	0.12	0
d_4	0	0.12	0	0.24	0.30

Figure 2.2: Representation of the *tf-idf* method.

GloVe

GloVe (Global Vectors) is an unsupervised learning method in which words are represented through vectors. The way GloVe represents a word with a vector is by using a large set of documents. It calculates the ratio of probabilities obtained from a word-word co-occurrence matrix. To calculate these probabilities, the model used Equation 2.3. This equation calculates the probability of word j appearing in the context of word i [26].

$$P_{ij} = P(j|i) \tag{2.3}$$

In Figure 2.3, there is a schematic representation of how GloVe generates a semantic relation between words. For example, we can see that the words “King” and “Queen” are in the same context. At the end of the learning phase, we would have each word represented as a vector.

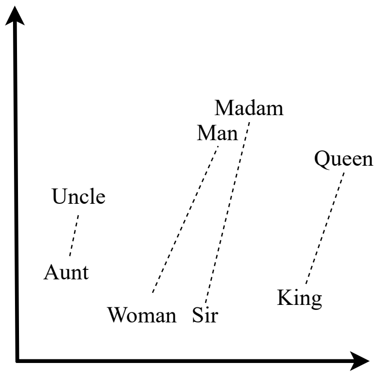


Figure 2.3: Representation of the GloVe semantic search.

Word2Vec

Word2Vec is a method that helps us to represent words as feature vectors. This method employs a feed-forward neural network with a single hidden layer that uses text data as an input to generate the output vectors [27].

In Figure 2.4, we can see a schematic representation of how the neural network of Word2Vec works. First, we create a vocabulary with unique words from our set of documents. Then, we create a vector according to the size of our vocabulary, filled with zeros except for the position of the word we want to represent. This vector will be the input of the neural network, the weights of the hidden layer will help represent a word as a feature vector. Finally, in the output layer, we have the context word probability in relation to the other words in the vocabulary.

fastText

Introduced by the Facebook Research AI team [28], fastText is another method to represent words as features vectors. This method is an extension of the Word2Vec model. The main difference between Word2Vec and fastText is that fastText represents a word as an n-gram of characters. An n-gram is a contiguous sequence of n number of characters from a given instance of text.

In fastText, the n-grams of a word are used as input for a feed-forward neural network. For example, instead of using the word “cold” as an input, fastText obtain the n-grams that will be: $<$*co, col, old, ld*$>$, with an *n=3*. The angular brackets represent the beginning and the end of a word. The process continues using a

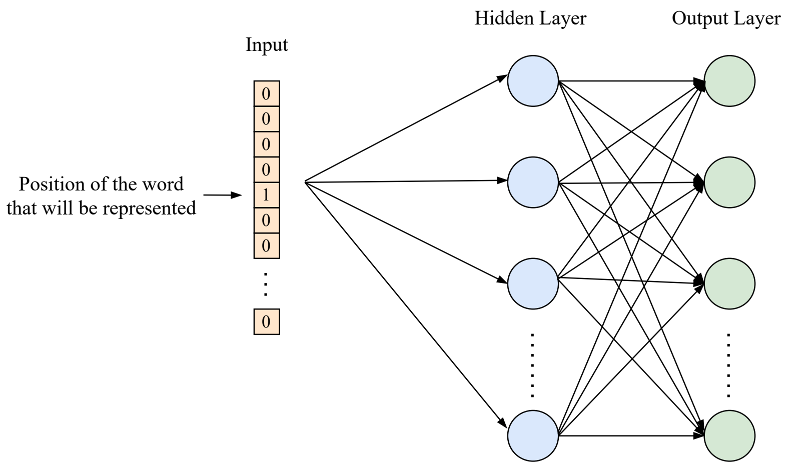


Figure 2.4: Feed-forward neural network that Word2Vec uses.

neural network with an architecture similar to the one we see in Figure 2.4, where the weights of the network are used as feature vectors.

2.1.3 Normalization Process

Data normalization is a common step in the pre-processing phase of a data mining project. This step aims to transform the data into a standard range. For example, convert the measurements units from meters to inches or kilograms to pounds.

For text content, there are a variety of techniques for normalizing data, such as l^2-norm, l^1-norm, and max norm. The l^2-norm, also known as Euclidean normalization, is one of the most popular normalization technique. This is defined by Equation 2.4 for a vector $\mathbf{x}$, which is an n-dimensional vector with values $x_1, x_2, \cdots, x_n$.

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{2.4}$$

The normalization process of the vector $\mathbf{x}$ is given by the Equation 2.5, where $\mathbf{x}_N$ is the vector $\mathbf{x}$ normalized.

$$\mathbf{x}_N = \frac{\mathbf{x}}{\|\mathbf{x}\|_2} \tag{2.5}$$

2.2 Machine Learning and Deep Learning Models

Machine learning (ML) is a subarea of artificial intelligence (AI) where, through the use of algorithms, computers learn to recognize complex patterns based on data. With this learning, computers can autonomously perform tasks such as facial recognition, self-driving cars, speech recognition, etc.

Deep learning (DL) is a subdiscipline of machine learning. Deep learning uses the concept of artificial neural networks where three layers or more are used to identify complex patterns and extract various features from the input data.

There are different approaches that machine learning and deep learning models use to learn. Supervised learning is a widespread technique among machine learning and deep learning models. Models that use supervised learning have a set of variables X, which corresponds to the training data; a set of Y variables, which are labels of the corresponding provided class. The main objective is to learn a mapping function or weights to predict the output variables of a new set of variables never seen by the model.

In this work, we use supervised learning in all the models we implemented. We provide the labeled instances to their corresponding class. In this section, we describe the machine learning and deep learning models used in this thesis.

2.2.1 Support Vector Machines

This model was developed by Vladimir Vapnik at the AT&T Bell laboratories [29]. The objective of this model is to find a *hyperplane* that represents a boundary to separate the instances from one class to another.

The methodology followed by SVM to find the hyperplane is that given a set of n points of the form $[(\mathbf{x}_1, y_1), \cdots (\mathbf{x}_n, y_n)]$, where y_n represents to which class $\mathbf{x}_n$ belongs, tries to find the *maximum-margin hyperplane* that divides the points $\mathbf{x}_n$ with its corresponding class. SVM uses Equation 2.6 to find the hyperplane, where $\mathbf{x}$ is a set of points, $\mathbf{w}$ is the normal vector and b is the distance from the hyperplane to the origin.

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.6}$$

2.2.2 Logistic Regression

Logistic regression is a classification model that uses probability to model a categorical output of a given number of input features. This model uses Equation 2.7 to calculate the probability that a feature set represented by the vector $\mathbf{x}$ belongs to a class (e.g., true or false). In Equation 2.7, β_0 is the bias and β_1 is the weight of the input vector of features $\mathbf{x}$.

$$P(x) = \frac{e^{(\beta_0 + \beta_1 \mathbf{x})}}{1 + e^{(\beta_0 + \beta_1 \mathbf{x})}} \tag{2.7}$$

Applying natural logarithm (ln) two both sides of the Equation 2.7 and solving it, we obtain Equation 2.8. In Equation 2.8, the *odds* of a class are calculated. The odds are the probability of an event divided by the probability of a no-event.

$$ln(odds) = \beta_0 + \beta_1 \mathbf{x}_1 \tag{2.8}$$

2.2.3 Random Forest Classifier

Random Forest is an ensemble model that combines a series of methods for an identification task. In particular, this ensemble uses Decision Trees and calls it *forest* [30]. Random Forest uses a sampled random vector with the same distribution for all the trees in the forest. Then, each tree votes and the class with the majority of votes is chosen.

In Figure 2.5, we can see a schematic representation of this technique. We have four trees that vote for a class. In the final part, the algorithm chooses the class with more votes.

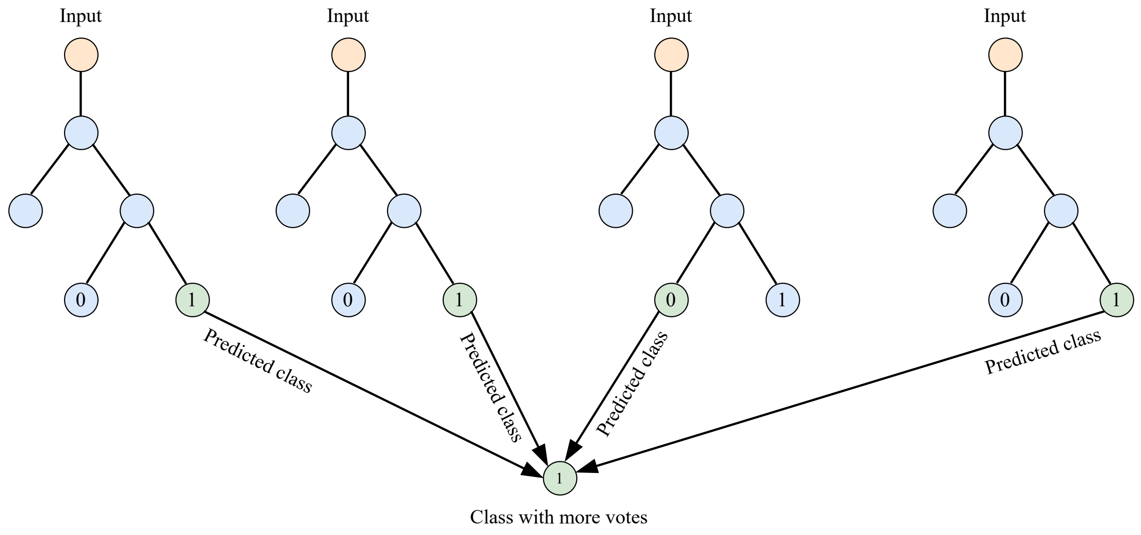


Figure 2.5: Representation of the process in a Random Forest Calssifier.

2.2.4 Naïve Bayes Classifier

Naïve Bayes Classifier is a model based on the Bayes' probabilistic theorem. This theorem developed by Thomas Bayes is defined by Equation 2.9 which shows the probability of an event occurring given the knowledge before that event. In Equation 2.9, $P(A)$ is the probability that A occurs; $P(B)$ is the probability that B occurs; $P(A|B)$ the probability that A occurs given B; $P(B|A)$ the probability that B occurs given A.

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{2.9}$$

With Equation 2.9 we can define a Multimodal Naïve Bayes that predicts the conditional probability to identify if a document belongs to a class or another. In Equation 2.10, $P(c_i|d_j)$ is the probability that document d_j belongs to the class c_i; $P(d_j|c_i)$ is the probability of the documents d_j given a class c_i; $P(c_i)$ is prior probability of a class c_i; $P(d_j)$ is the prior probability of d_j.

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)} \tag{2.10}$$

2.2.5 *k*-Nearest Neighbors

k-Nearest Neighbors (KNN) is an instance-based classifier. This classifier determines the class of a instance by its nearest neighbors [31]. The algorithm stores instances that contain a set of feature vectors and the corresponding class labels from a training set. Then, using a test feature vector, the model classifies this vector by the most frequent class among the k training instances nearest to that test vector. The variable k represents the number of neighbors and is a constant defined by the user.

The nearest neighbors are found using different distance functions such as the *cosine similarity* which is defined by Equation 2.11, where $\mathbf{x}$ and $\mathbf{y}$ are vectors. Cosine similarity ranges from 0 to 1, where 0 is not similar and 1 is similar.

$$SIM(\mathbf{x}, \mathbf{y}) = cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} \tag{2.11}$$

2.2.6 Reconstructive Models

Reconstructive models use statistical properties from diverse techniques to classify unseen documents. Based on the idea of reconstructing new documents using a transformation matrix computed from similar documents. These models have been used in the classification of emails [32] and categorization of general documents [33].

At the end of the processing phase, we have a features matrix X with dimensions $m \times n$ where m is the number of instances and n is the size of the extracted vocabulary. We have a Y matrix of dimensions $n \times 1$ that contains the classes that correspond to the instances of the matrix X. Using the labels, we can represent the matrix X as $X = [X_1, X_2, \cdots, X_i]$ where X_i is the matrix that corresponds to the i^{th} class.

During the training phase, the algorithm computes a set of projection matrices W_i for each matrix of the class X_i using Equation 2.12.

$$X_i = Z_i W_i^T \tag{2.12}$$

Afterward, using Equation 2.13 we obtain a matrix Z_i which is a low-rank matrix of the class X_i.

$$Z_i = X_i W_i \tag{2.13}$$

In Equation 2.13, W_i is a projection matrix with dimensions $m \times r$. The objective is to find an optimal matrix W that represents a minimum loss of information. At the end of the training process, we obtain a set of projection matrices for each class $\{W_1, W_2, \cdots, W_i\}$.

In the test phase, the algorithm takes an unseen instance vector $\mathbf{q}$ and projects it over the low-rank space with the corresponding projection matrix W_i for each class. The result is a reduced vector $\mathbf{p}_i$ for each class as we see in Equation 2.14.

$$\mathbf{p}_i = \mathbf{q} W_i \tag{2.14}$$

Subsequently, we obtain a reconstructed vector $\tilde{\mathbf{q}}_\mathbf{i}$ for each class and every reduced vector $\mathbf{p}_i$ will be built in the same space using the projection matrices W_i as seen on Equation 2.15.

$$\tilde{\mathbf{q}}_\mathbf{i} = \mathbf{p}_i W_i^T \tag{2.15}$$

The process continues using Equation 2.11 to calculate the similarity between each vector $\tilde{\mathbf{q}}_\mathbf{i}$ and the original vector $\mathbf{q}$ as $SIM(\tilde{\mathbf{q}}_\mathbf{i}, \mathbf{q})$, where the index of the maximum similarity will be the class of the instance vector $\mathbf{q}$.

There are different ways to compute the W matrix. However, the most common way to obtain this matrix is through the use of methods such as Principal Component Analysis (PCA), Singular Vector Decomposition Reconstructive (SVDR) and Non-Negative Matrix Factorization Reconstructive (NMFR).

2.2.7 LSTM

Long Short-Term Memory (LSTM) neural networks are a class of Recurrent Neural Network (RNN) [34]. RNN presents a short-term memory issue called *vanishing gradient* [35], where the information may fade through the layers during the training phase. This problem prevents the neural network's weights from being updated since the gradient values are in charge of updating these weights. As a result, if these values become too small, they do not contribute to the neural network learning.

To solve the vanishing gradient problem, an LSTM uses a mechanism of *gates* to regulate the flow of information through the layers. For example, in Figure 2.6, we can see a representation of an LSTM unit. We can see that we have three different gates: forget gate, update gate and output gate. The forget gate receives an input X_t and a previous hide state h_{t-1}, combining these two inputs and with a *sigmoid* function, and decides if it keeps the information or not.

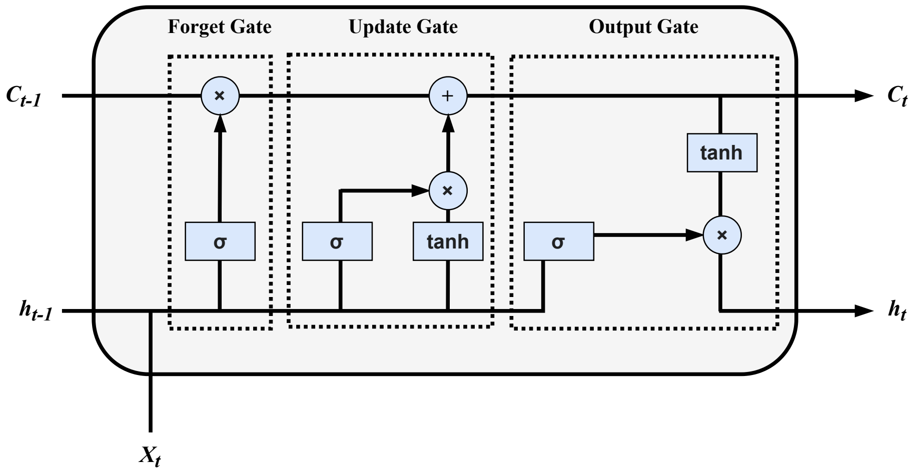


Figure 2.6: Representation of an LSTM unit.

The update gate consists of two steps. First, we take the previous hidden state h_{t-1} and the current input X_t and combine these two inputs with another sigmoid function where values close to one will be kept. Second, we take the same two inputs, but instead of using a sigmoid function, we use a *hyperbolic tangent* function (*tanh*). Finally, we multiply ($\times$) these two outputs and get the update gate's output.

To calculate the new cell state C_t, the network takes the previous cell state C_{t-1} and multiplies it with the output of the forget gate. Then, using the output of the update gate and adding ($+$) it with the result of the multiplication of the forget gate and the previous state, we get the new cell state C_t.

The new hide state h_t is the result of the multiplication of combining two elements. The first one is the combination of h_{t-1} and X_t, passing it through a sigmoid function. In the second one, we take the new cell state C_t and filter it through a *tanh* function. At the end of this process, we get the new h_t.

2.2.8 Bi-LSTM

Bi-directional Long Short-Term Memory (Bi-LSTM) are neural networks that use two LSTM units together in their architecture [36]. In this model, one LSTM unit is used to process information in a forward direction, and the other unit is used in a backward direction. The objective of Bi-LSTM is to predict an output more efficiently, having more information from the input. For example, in the sentence: “it may be quite simple, but now it is done”, Bi-LSTM networks process the information from left to right and from right to left. Thus, the network can increase the amount of knowledge of which word can precede or follow the word *but*.

In Figure 2.7, we can see a representation of a Bi-LSTM neural network. We observe that it has two blocks conformed by n number of LSTM units. The first block, forward LSTM, uses the information in a forward way. Similarly, the backward LSTM block takes the input but it is operated from a right to left direction.

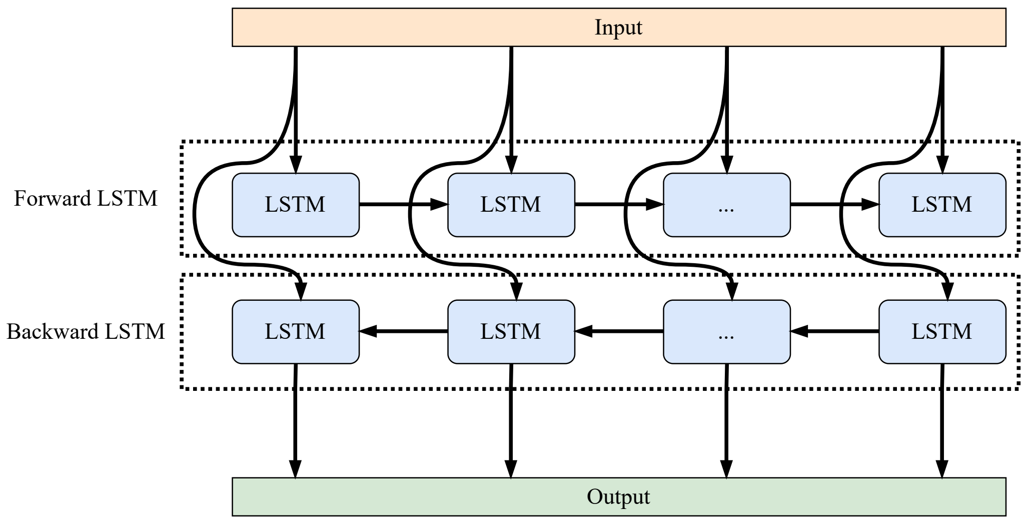


Figure 2.7: Representation of a Bi-LSTM network.

2.2.9 GRU

Gated Recurrent Unit (GRU) is a class of RNN. It takes the idea of gates from LSTM networks to control the flow of information during the learning phase [37]. Nevertheless, unlike LSTM networks, where we have three gates that are in charge of the flow of information, in GRU networks, we only have two gates: reset and update gates. Having two gates is an advantage since the number of operations is fewer compared to the LSTM network. In Figure 2.8, we can observe a schematic representation of a GRU unit and the gates that conform it.

The reset gate receives the input X_t and the previous hide state h_{t-1} and, using a sigmoid function, multiplies the result of that operation by h_{t-1} and generates the output of the reset gate.

The update gate will take X_t and h_{t-1} and generate two outputs. The first output of the update gate takes two elements. The first one combines the input X_t and the output of the reset gate and pass through a tanh function; the second one is conformed by the operation of taking X_t, h_{t-1} and passing through a sigmoid function. Then, using a multiplication operator, we combined these two elements. The second output of the update gate is the subtraction of the result after passing X_t and h_{t-1} through a sigmoid function. Next, we multiply the result of this subtraction with the previous hide state h_{t-1}.

Finally, the new hide state h_t will be the sum of the two outputs obtained in the update gate.

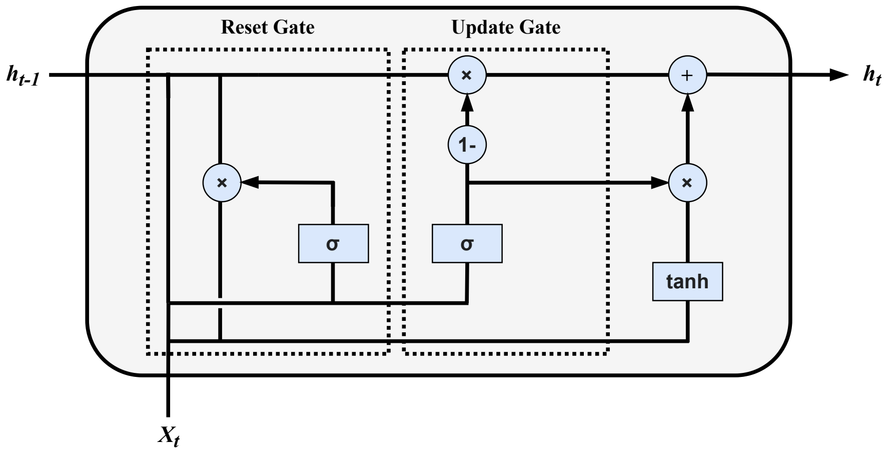


Figure 2.8: Representation of a GRU unit.

2.2.10 BERT

BERT (Bi-directional Encoder Representations for Transformers) is a recent deep learning model introduced by the Google AI team [19]. BERT starts from the idea of the Transformers. Transformers are models that learn contextual similarities among words. These models use two essential mechanisms an *encoder* and a *decoder*. In Figure 2.9, we see a representation of a Transformer. We can observe that the model has the two mechanisms mentioned before.

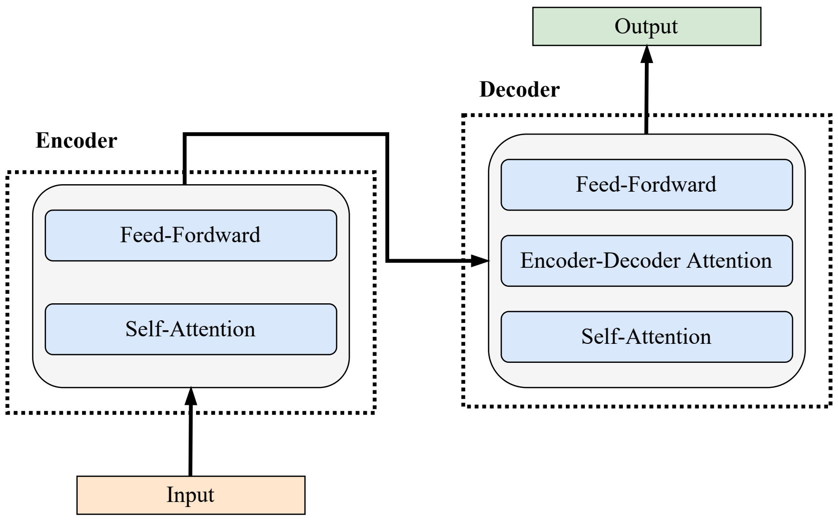


Figure 2.9: Representation of a Transformer.

The encoder mechanism has two elements. First, we have a *self-attention* layer that helps the encoder to see which other words are in the input sentence while encoding a specific word. The second element is a feed-forward neural network that receives the output of the self-attention layer.

The decoder mechanism has the same two elements that the encoder with the difference that between these two elements there is an attention layer that helps the decoder to see essential parts of the input.

BERT uses the encoder part of Transformers, stacking an n number of encoders, one on top of the other. BERT models are trained with extensive data sets made up of different texts in various languages. The objective of this process is that BERT models can comprise patterns in language and later be used in other supervised tasks such as fake news identification, sentiment analysis, sentence translation, among others.

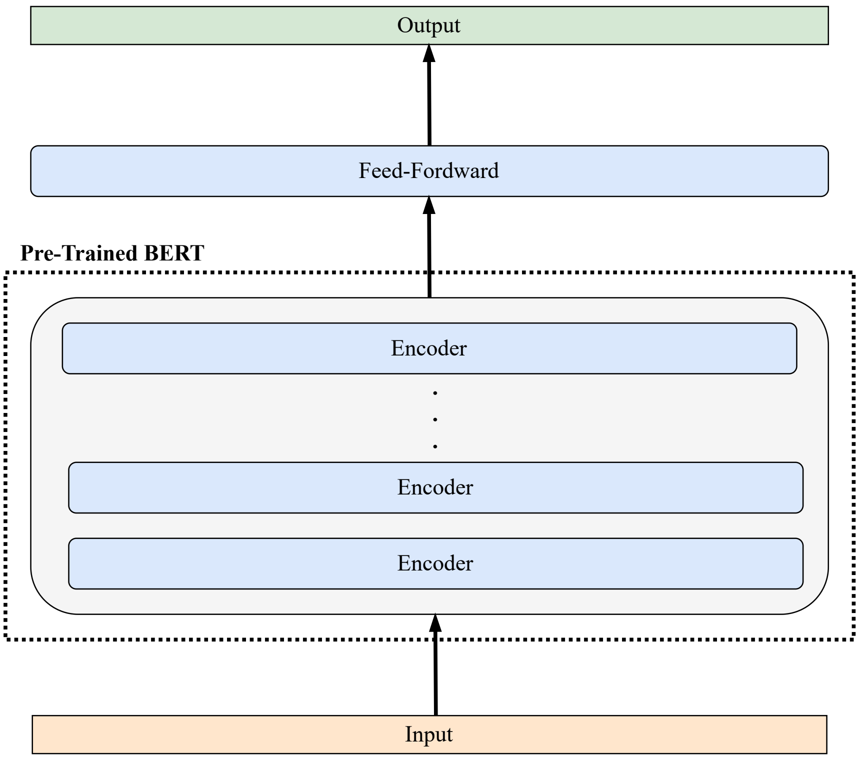


Figure 2.10: Representation of BERT.

In Figure 2.10, we can observe a schematic representation of a BERT model.

2.3 Evaluation Setup

In this section, we present the different performance metrics and evaluation techniques used to evaluate our machine learning and deep learning models.

2.3.1 Performance Metrics

Evaluating the performance of machine learning and deep learning models is essential for knowing the quality of a model. There are different methods to evaluate the models, and below we describe the ones used in this work.

All classification models produce an output for each instance. Thus, each instance could be classified as belonging to a class (e.g., *positive*) or not belonging to a class (e.g., *negative*). Taking the predictions by a model, we can construct a table called *confusion matrix*. The confusion matrix relates the real class of the instances (columns) with the classes predicted by the model (rows). We can see a schematic representation of the confusion matrix in Figure 2.11. The cells in the matrix contain the number of predictions that result to be *true positive* (TP), *true negative* (TN), *false positive* (FP) and *false negative* (FN). TP is a result where the model predicts the positive class correctly. Similarly, TN is a result where the model predicts the negative class correctly. On the other hand, FP is when the model predicts the positive class incorrectly. Finally, FN is when the model predicts the negative class incorrectly. From the confusion matrix, we can define a variety of performance metrics.

	Model's output: Positive	Model's output: Negative
Real Value: Positive	TP	FN
Real Value: Negative	FP	TN

Figure 2.11: Representation of a confusion matrix.

Accuracy is one of the famous metrics to evaluate machine learning and deep learning models. It measures the proportion of correct predictions made by a model, and it is defined by Equation 2.16. Nevertheless, accuracy is a pretty sensitive metric when we have unbalanced datasets. Unbalanced datasets contain an unequal class distribution. When we work with this type of dataset, we can reach higher results with this metric since one o more classes tend to dominate the data. However, this is a problem since the minority class is ignored and the results for this metric are biased.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.16}$$

To compensate for inconsistencies of accuracy, another set of metrics are a good option when we have unbalanced datasets such as *Precision*, *Recall* and *F1*. We can define these metrics in Equations 2.17, 2.18 and 2.19, respectively [38]. In Equation 2.19, we can see the definition of F1 that is the harmonic mean between precision and recall.

$$Precision = \frac{TP}{TP + FP} \tag{2.17}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.18}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.19}$$

Another metric used to evaluate the performance of models is the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC). The AUC-ROC plots the relationship between the rate of true positives (TPR) and the rate of false positives (FPR). The TPR and FPR are defined by Equations 2.20 and 2.21, respectively.

$$TPR = \frac{TP}{TP + FN} \tag{2.20}$$

$$FPR = \frac{FP}{TN + FP} \tag{2.21}$$

AUC-ROC evaluates the degree of separability, measuring the probability that a model classifies an instance in a random way in one class rather than choosing an instance in another class.

In problems where we have several classes, averaging the evaluation measures can help us see if a model is performing well despite the number of classes. *Macro-average* and *micro-average* are two techniques used to obtain the average of metrics such as precision, recall, F1 and AUC-ROC. Macro-average computes each metric individually for each class to calculate the average at the end. Micro-average computes the contributions of each class to obtain the average [39].

2.3.2 Dataset Split

There are several techniques to estimate the generalization of a predictive model; an example of these techniques is *k-fold cross-validation.* In cross-validation, we split the data one or several times to use one part to train the model and the other for validation. In k-fold, k is a constant used to know the number of subsets obtained from a dataset and the number of times the method is repeated, and the user defines this constant. There are different variants of this technique such as 3-fold cross-validation, 5-fold cross-validation, 20-fold cross-validation, etc. One of the most popular variants is stratified 10-fold cross-validation. In this variant, the dataset is split into ten parts, where each class is represented with about the same portion of instances.

Another popular technique is *train-test* split. The objective of this method is to separate the data into two subsets. The first one is used to train the model, and the other one is used to make predictions with the model. The amount of data found in each subset will depend on each problem. There are common divisions we can do in a dataset such as train: 70% test: 30%, train: 80% test: 20%, etc. [40]

Chapter 3

Methodology

In this chapter, we describe the methodology used in this work, as presented schematically in Figure 3.1. We divide the methodology in four phases: data gathering, data processing, experimental setup and results. The first three phases are described in this chapter, whereas the last one is described in Chapter 4.

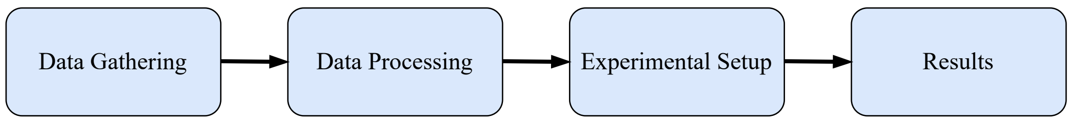


Figure 3.1: Representation of the methodology process.

3.1 Data Gathering

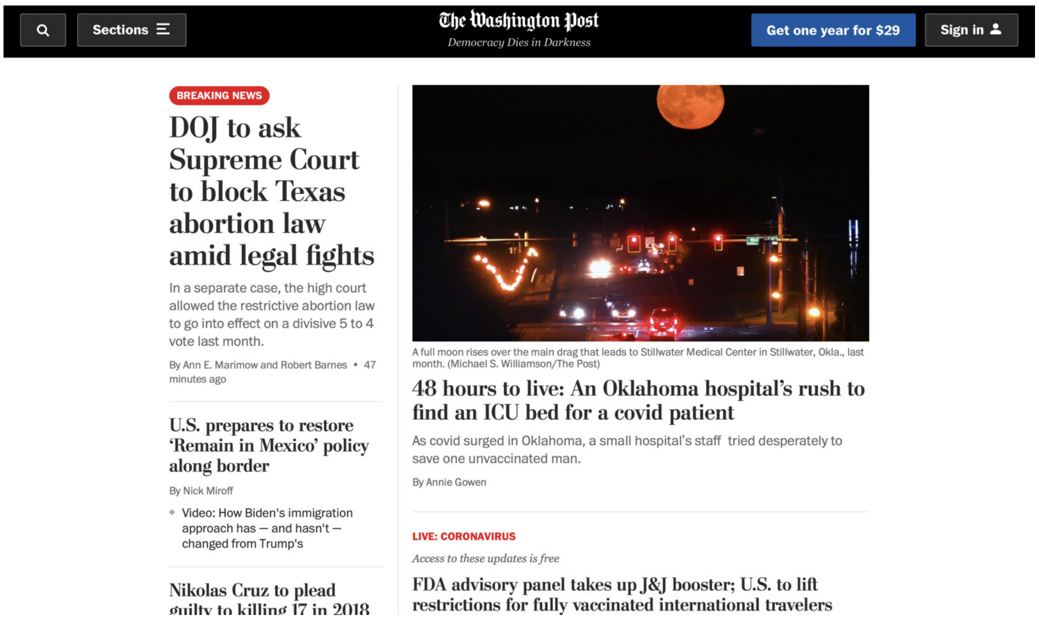
The first phase in our methodology was the data gathering process. First, we did an extensive search on the internet and the state-of-the-art, looking for pre-collected datasets that contain fake news from different sources such as news web pages, social networks, microblogs, etc.

At the end of this search, we found different datasets containing news from various sources such as CNN, El Universal, Facebook, Twitter, etc. The pre-collected datasets we gathered for this work are COVID Fake News (COVID) [1], Spanish Fake News Corpus (SFNC) [6], LIAR (LIAR) [3], FakeNewsNet (FNN) [4], ISOT (ISOT) [5] and Fake Costa Rica News (FCRN) [2]. These datasets were previously collected by different researchers and organizations that scraped sites to extract real and fake news. In Figure 3.2, we can see an example of a site from where the information was extracted.

3.1.1 Dataset Description

After the gathering process, we ended with a total of six datasets. These datasets contain information from a variety of sources. The COVID dataset contains mainly news from the COVID-19 pandemic. LIAR, ISOT and FNN contain news from popular American news media such as CNN, The Washington Post, CBS and NBC.

[1]https://bit.ly/3zB1XZZ
[2]https://bit.ly/3wduNx6


Sections
The Washington Post
Democracy Dies in Darkness
Get one year for $29
Sign in

BREAKING NEWS

DOJ to ask Supreme Court to block Texas abortion law amid legal fights

In a separate case, the high court allowed the restrictive abortion law to go into effect on a divisive 5 to 4 vote last month.

By Ann E. Marimow and Robert Barnes • 47 minutes ago

U.S. prepares to restore 'Remain in Mexico' policy along border

By Nick Miroff

Video: How Biden's immigration approach has — and hasn't — changed from Trump's

Nikolas Cruz to plead guilty to killing 17 in 2018

A full moon rises over the main drag that leads to Stillwater Medical Center in Stillwater, Okla., last month. (Michael S. Williamson/The Post)

48 hours to live: An Oklahoma hospital's rush to find an ICU bed for a covid patient

As covid surged in Oklahoma, a small hospital's staff tried desperately to save one unvaccinated man.

By Annie Gowen

LIVE: CORONAVIRUS

Access to these updates is free

FDA advisory panel takes up J&J booster; U.S. to lift restrictions for fully vaccinated international travelers

Figure 3.2: Example of news web page.

The topics covered in these datasets were political and gossip news. Additionally, in FNN and LIAR datasets, we also found news that came from social media. In SFNC and FCRN, the information came from sources where the primary language is Spanish. In SFNC, we found news from Mexican sources such as Aristegui Noticias, El Universal, El Financiero, among others. Finally, in FCRN, the news came from Costa Rican sources and the topics covered talk about local and international news.

The data included in the datasets is primarily textual content from the news. Nevertheless, in some of these datasets, we can find other variables that were extracted from the news, such as news text, headline, speaker, source, speaker's job, among others. In this work, we focused on the text content; therefore, we only took the news text or the headline since this type of information could be found in all datasets.

Each of the news in most of the datasets is labeled with one class, *true* or *fake*, except for the LIAR dataset, where the news were labeled using six labels: *true*, *mostly-true*, *half-true*, *barely-true*, *false* and *pants-fire*. In this case, we took the instances labeled with the first three classes as true and the rest as fake. The task of labeling the news was in charge of different organizations and persons depending on the dataset. In LIAR, ISOT and FNN, organizations such as Gossip Cop [3] or PolitiFact [4] were in charge of verifying the veracity of the news. In these organizations, they have a board of experts who check different sources to find the information that ensures the credibility of a news item; if a piece of news is found fake or true, it is put in a database to consultancy later. In SFNC, the organization called Verificado [5] was in charge of verifying if the news was true or fake. In the case of the COVID and FCRN datasets, the labeling work was a task that was performed manually by the people who collected them and verified the veracity of the news.

In the case of the SFNC and LIAR datasets, they are divided into subsets.

[3]https://bit.ly/3AL56Wi
[4]https://bit.ly/2YQmnR7
[5]https://bit.ly/3p5C5SY

SFNC has a train and test dataset, whereas the LIAR dataset is divided into train, validation and test datasets. We use the training part of SFNC and the training and validation parts to describe SFNC and LIAR datasets, respectively.

For each dataset, we computed series of statistics to understand the distribution of the data among the classes and in the whole dataset. We present these statistics in Table 3.1.

Dataset	**True**	**Fake**	**Total**
COVID	474	9,727	10,201
SFNC	338	338	676
FNN	15,569	5,347	20,916
ISOT	21,416	22,857	44,273
LIAR	7,276	13,204	20,480
FCRN	2,531	3,584	6,115

Table 3.1: News distribution among the classes.

In Table 3.1, we can see the distribution among the classes and the total number of news in each dataset. We can observe that in some datasets, we have an unbalanced distribution. For example, in the COVID dataset, we see that the number of instances of the fake class is superior compared with the true class. On the other hand, in the FNN dataset, we have more news corresponding to the true class than the fake class. In the LIAR dataset, we have a slight difference between the number of instances in each class. We also have datasets with almost the same number of news per class, such as ISOT, FCRN and SFNC datasets; the last one with exactly the same number of instances for each class.

We extracted a series of statistics of some relevant features from the textual content found in the datasets. These features are words, hashtags (#), ats (@), links, abbreviations and *emojis/emoticons*. The last ones can be used to express emotions with letters (e.g., :}D). Recently, emojis are becoming popular since they take the idea from emoticons but now are color icons to represent different emotions or situations [41].

In this direction, we computed the average number of words, emojis (emojis/emoticons), hashtags, ats, links and abvs. (abbreviations) per each dataset as we observe in Table 3.2. This table shows that the words are the feature with a major presence in the news. In the case of COVID and LIAR datasets, the number of words is less than the other datasets since only the news headlines was collected. The rest of the features are almost null in most datasets; although we can observe a slight presence of abbreviations in the FNN and ISOT datasets. Thus, we consider the words as the main feature set.

In Table 3.3, we present series of statistics of each dataset among the classes using words. We computed the average number of words, the standard deviation (STD), the median and the maximum length divided by the minimum length (Max/Min) among the classes. For example, in COVID, ISOT, LIAR and FCRN datasets, we can see there is a slight difference between the average number of words in the fake class and the true class. On the other hand, in SFNC and FNN datasets, we observe that the average number of words in the true class is more significant than the news

	Features					
Dataset	**Words**	**Emojis**	**Hashtags**	**Ats**	**Links**	**Abvs.**
COVID	9.53	0.06	0.00	0.00	0.00	0.32
SFNC	194.90	0.13	0.04	0.04	0.11	0.06
FNN	320.09	1.31	0.24	0.13	0.53	16.69
ISOT	225.50	0.31	0.16	0.14	2.84	9.01
LIAR	10.37	0.00	0.00	0.00	0.00	0.41
FCRN	185.97	0.22	0.05	0.08	0.20	0.06

Table 3.2: Average number of features per dataset.

with the fake class. This may happen because real news has more information since it must show that what is written is real, while fake news only uses trending words to persuade readers.

	True				**Fake**			
Dataset	**Average**	**Median**	**STD**	**Max/Min**	**Average**	**Median**	**STD**	**Max/Min**
COVID	9.2	9.0	3.5	16.0	9.8	9.0	5.1	31.0
SFNC	239.9	201.0	155.8	30.7	149.8	149.8	77.0	93.3
FNN	353.4	190.0	698.5	4,800.5	286.7	181.0	556.1	9,661.0
ISOT	222.6	207.0	156.2	178.9	228.3	192.0	222.5	4,696.0
LIAR	10.3	10.0	6.2	270.0	10.4	10.0	4.7	185.0
FCRN	167.1	118.0	141.2	76.0	204.8	169.0	146.1	725.0

Table 3.3: Statistics of words in each dataset per class.

With the words, we extracted a vocabulary represented by the unique words in each dataset. Table 3.4 presents the size of the vocabularies of each dataset. We see that FNN and ISOT datasets have a much bigger vocabulary compared with the other datasets.

Dataset	**Vocabulary**
COVID	11,446
SFNC	24,260
FNN	123,604
ISOT	122,514
LIAR	12,792
FCRN	66,644

Table 3.4: Vocabulary size of words in each dataset.

3.2 Data Processing

Once we finished the gathering process, we processed the data. All the data that we collected in the gathering step are considered *raw data*; we extracted and processed different superficial and deep features from the datasets.

In the case of superficial features, we extracted words, hashtags, ats, links, abbreviations and emojis, as we mentioned before. To retrieve these features, we wrote a Python code to extract them from the text content. We used different regular expressions to extract words, hashtags, ats and links. In the case of hashtags and ats, we only kept the words and removed the tags (#) and ats (@) found at the beginning of these features. To extract emoticons and emojis, we used the *EMOTICON_RE* [6] module of the NLTK library and the *emoji* [7] library, respectively. To extract the abbreviations, we used a list of common abbreviations in Spanish and English. The result of this process allowed the creation of six files, each containing one feature.

After the splitting process, we continued with a cleaning process, especially for the words; we performed it to remove stop words. Stop words are words commonly used in a language. For example, in English, words such as "for", "the", "in", among others, are stop words. In Spanish, words such as "la", "aquí", "los", etc. We used a list provided in the NTLK library for English and Spanish to remove the stop words. We also performed an additional cleaning process to remove those words with fewer characters (length < 3), words with a larger number of characters (length > 35) and HTML tags. Additionally, we converted all the letters to lowercase.

After the splitting and cleaning processes, we used the *tf-idf* method to vectorize our superficial features sets. The result of this process is a matrix that shows the relevance of each feature regarding news. For some models, we performed a tokenization process to convert the textual content into a matrix of integers, where each integer is the index of a token in a dictionary.

To extract the deep features from the textual content, we used pre-trained models such as GloVe, fastText and Word2Vec. In the case of GloVe, for Spanish, we used a model trained over the *Spanish Billion Word Corpus* [8], which contained a dictionary of 800,000 words, and each one is represented by a vector of size 300. For English, we used a trained model over a Twitter dataset [9], which contained a dictionary of 1.2 million words, each one represented by a vector of size 200. With fastText, we used the models trained over a Wikipedia dataset [42]. For English and Spanish, the words are represented with vectors of size 300 and with dictionaries of 2 million words. In the case of Word2Vec models, they are trained over a Wikipedia dataset [43]. Both English and Spanish models represent words with a vector of size 300. The dictionaries are conformed by 1 million words for the Spanish model and 4.5 million words for the English model. For the features GloVe, fastText and Wrod2Vec, we calculated the average vector of all the words found in the news. Therefore, each news is represented by an average vector of 300 or 200 dense features.

Finally, we normalized the matrices obtained from the superficial and deep features using the l^2-norm.

In this work, we used the words as the main superficial features since the other

[6]https://bit.ly/3mYdrkt
[7]https://bit.ly/3lLnGJF
[8]https://bit.ly/3n68hmN
[9]https://stanford.io/3BTgXmt

features, as we mentioned in Section 3.1.1, do not have enough presence in the news. Nevertheless, we merged these features with words to create a new feature set that includes all the features (AF). We also use the text content and do not perform the cleaning process to create a feature set called raw text features (RT). Different from the AF feature set, this feature set keeps uppercase words and special characters. We ended with a total of six feature sets: words, AF, RT, GloVe, fastText and Word2Vec.

We implemented all the codes [10] in this step using Python and different libraries such as *scikit-learn* [11], *fasttext* [12] and *gensim* [13].

3.3 Experimental Setup

With the textual data transformed, we conducted a series of experiments using a variety of machine learning and deep learning models. We used eight machine learning models: Support Vector Machines with a linear kernel (SVM), Logistic Regression (LR), Multinomial Naïve Bayes (MNB), Support Vector Machines with a Stochastic Gradient Descent optimizer (SGDC), k-Nearest Neighbors (KNN), Random Forest Classifier (RFC), Non-Negative Matrix Reconstructive (NMFR) and Singular Vector Decomposition Reconstructive (SVDR). Additionally, we used four deep learning models: Long Short-Term Memory Network (LSTM), Bi-directional Long Short-Term Memory Network (Bi-LSTM), Gated Recurrent Unit (GRU) and Bi-directional Encoder Representations for Transformers (BERT).

With COVID, FNN, ISOT and FCRN datasets, we performed a 10-fold cross-validation with each model. This is because we aim to obtain robust results regarding the independence between the training and the test sets. In addition, in some machine learning models, it was necessary to find optimal values for their *hyper-parameters*. A hyper-parameter is a constant value set by the user before the training process. Thus, we performed an inner 3-fold cross-validation with the datasets mentioned before. The models where we needed to find the best hyper-parameters were SVM, LR, SGDC, KNN, RFC, NMFR and SVDR. In Table 3.5 there are the hyper-parameters we tested. In the fourth column, there is a list of different values that we used to find the best hyper-parameter for the corresponding classifier.

With deep learning models, we used a setup for its hyper-parameters. In Table 3.6, we can observe the number of epochs, batch size and number of neurons. In the case of BERT, we cannot define the number of neurons since the approach used is different and does not use the number of neurons as a hyper-parameter. The number of epochs and batch size is minor compared to the other deep learning models since BERT's computational cost is higher. Similarly, with BERT, we used pre-trained models for each language. For Spanish, we used a pre-trained BERT cased model called BETO [44] and a base-cased pre-trained model for English [14].

In the case of the SFNC and LIAR datasets, we did not perform a 10-fold cross-validation since datasets are divided into subsets. Instead, for SFNC, we performed a 5-fold cross-validation using the values of Table 3.5 on the training set to find the best

[10]Code available at: https://bit.ly/3wXNuWY
[11]https://bit.ly/3pcg1q2
[12]https://bit.ly/3vp3KPR
[13]https://bit.ly/3BSiLMH
[14]https://bit.ly/2XlEqxL

Model	HP	Description	Values
SVM	c	Regularization parameter	[0.01, 1, 10, 100]
LR	c	Regularization parameter	[0.01, 1, 10, 100]
SGDC	c	Regularization parameter	[0.01, 1, 10, 100]
KNN	k	Number of neighbors	[1, 2, 3, 5, 10]
RFC	n	Number of trees	[10, 50, 100, 200, 500]
NMFR	r	Number of components	[1, 2, 4, 8, 16, 32]
SVDR	r	Number of components	[1, 2, 4, 8, 16, 32]

Table 3.5: Values tested for the hyper-parameters of each classical machine learning model

Model	Epochs	Batch size	Number of neurons
LSTM	100	64	128
Bi-LSTM	100	64	128
GRU	100	64	128
BERT	7	6	-

Table 3.6: Values tested for the hyper-parameters of each deep learning model.

hyper-parameters of the corresponding machine learning models. In addition, we used the same setup shown in Table 3.6 for the deep learning models in combination with the training set. In the case of the LIAR dataset, we used the training set to train the models and the validation set for the hyper-parameter optimization of the machine learning and deep learning models using Table 3.5 and Table 3.6, respectively.

To evaluate the performance of each model, we used the next metrics: Accuracy, F1 (macro) and AUC-ROC (macro). The metrics can take values between 0 and 1, where 0 is the worst and 1 is the best.

We implemented the machine learning and deep learning models using Python and libraries such as *scikit-learn*, *NumPy* [15], *Tensorflow* [16] and *ktrain* [45]. We ran the experiments in a PC with an Intel Xeon Silver processor with 8 cores at 2.1 GHz and 128 GB of RAM.

[15]https://bit.ly/3lXzTeu
[16]https://bit.ly/3DWXQZu

Chapter 4

Results

In this chapter, we present the results of this work. In Tables 4.1 to 4.18 we can observe the results of the methodology described in Chapter 3. In Tables 4.1 to 4.9, we present the results using superficial features, whereas in Tables 4.10 to 4.18 we show the results using deep features. In the tables, columns represent the dataset and rows represent the machine and deep learning models used. Each table shows the results for a different evaluation metric. In the case of COVID, FNN, ISOT and FCRN datasets, each table shows the median of the 10-fold cross-validation of each model. Additionally, in the tables, we present the median per model and dataset. The values in bold represent the best values per dataset for each metric.

As a baseline, we consider a totalitarian model for each dataset. This model would classify the majority class for the test data in terms of accuracy. In this direction, news identification using this baseline would be the following: 0.95 for COVID, 0.50 for SFNC, 0.74 for FNN, 0.51 for ISOT, 0.64 for FCRN and 0.59 for FCRN.

In Tables 4.1 to 4.3, we can observe the results for accuracy, F1 and AUC-ROC using words as feature. For COVID and FNN datasets, SVM is the model with the best results for both models in the three tables. On the other hand, BERT presents good results in SFNC and FCRN datasets. Additionally, Bi-LSTM presents good results for accuracy, F1 and AUC-ROC for the ISOT dataset, although SVM and LR present results close to the ones obtained by this model for this dataset in all the evaluation metrics. In the LIAR dataset, the result obtained by LSTM in Table 4.1 is the best in terms of accuracy, whereas in Tables 4.2 and 4.3 models such as RFC, SVDR and NMFR obtain good results in terms of F1 and AUC-ROC. If we see the medians per model, SVM presents the best performance for the three metrics combined with words. In the medians per dataset, the models have consistent results for all the metrics with the ISOT dataset.

From Tables 4.4 to 4.6 we can see the results for accuracy, F1 and AUC-ROC using AF. For this feature, we can see that the BERT model obtained good results with the SNFC, FCRN and COVID datasets for all metrics. Additionally, in the COVID dataset, models such as SVM and LR obtained acceptable results, as we see in Tables 4.4 to 4.6. With the FNN dataset, SVM was the model with the best results. Nevertheless, LR also obtained the best results in terms of F1 and AUC-ROC, as we can observe in Table 4.5 and 4.6, respectively. For the ISOT dataset, we can see that Bi-LSTM obtained the best results for all the metrics. SVM presents the best results in the medians per model followed by LR and BERT for the three metrics. For the medians per dataset, the models obtained the best results for the ISOT dataset for the three performance metrics, which means this dataset is easy to classify disregarding the classification approach.

The results of the last set of superficial features, RT, are presented in Tables 4.7 to 4.9. Again, BERT obtains good results in the majority of the datasets in the three metrics. In the case of the COVID and FNN datasets, SVM and LR also obtained good results in terms of F1 and AUC-ROC. For the ISOT dataset, Bi-LSTM, SVM and LR present the best results in all the metrics. Additionally, models such as GRU and NMFR present good results in terms of accuracy and AUC-ROC, as we see in Table 4.7 and Table 4.9, respectively. In the medians per model, BERT presents the best performance, although SVM has similar results. The models have consistent results for all the metrics with the ISOT dataset, especially in terms of F1 and AUC-ROC, which means that it is easier to classify.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	**0.98**	0.72	**0.86**	0.99	0.58	0.93	**0.90**
LR	**0.98**	0.73	0.85	0.99	0.60	0.92	0.89
MNB	0.95	0.72	0.77	0.93	0.65	0.80	0.79
SGDC	0.95	0.72	0.74	0.93	0.65	0.59	0.73
KNN	0.96	0.59	0.81	0.75	0.60	0.80	0.78
RFC	0.97	0.80	0.84	0.99	0.64	0.92	0.88
SVDR	0.95	0.73	0.80	0.92	0.57	0.85	0.83
NMFR	0.95	0.73	0.80	0.92	0.58	0.85	0.83
GRU	0.97	0.61	0.73	0.98	0.65	0.80	0.77
BERT	**0.98**	**0.84**	0.84	0.52	0.65	**0.96**	0.84
Bi-LSTM	0.97	0.72	0.82	**1.00**	0.64	0.89	0.86
LSTM	0.97	0.69	0.74	0.98	**0.66**	0.88	0.81
Median	**0.97**	0.72	0.80	0.95	0.64	0.86	-

Table 4.1: Results for accuracy using words.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	**0.85**	0.72	**0.79**	0.99	0.54	0.92	**0.82**
LR	0.83	0.73	**0.79**	0.99	0.55	0.92	0.81
MNB	0.51	0.71	0.54	0.93	0.44	0.78	0.63
SGDC	0.49	0.72	0.43	0.93	0.39	0.37	0.46
KNN	0.77	0.57	0.73	0.73	**0.57**	0.80	0.73
RFC	0.81	0.80	0.73	0.99	**0.57**	0.91	0.81
SVDR	0.73	0.73	0.75	0.92	0.56	0.85	0.74
NMFR	0.73	0.73	0.73	0.92	**0.57**	0.85	0.73
GRU	0.77	0.60	0.46	0.98	0.51	0.80	0.69
BERT	0.85	**0.84**	0.77	0.34	0.39	**0.96**	0.81
Bi-LSTM	0.72	0.72	0.75	**1.00**	0.41	0.89	0.74
LSTM	0.80	0.68	0.47	0.98	0.56	0.87	0.74
Median	0.77	0.72	0.73	**0.95**	0.54	0.86	-

Table 4.2: Results for F1 using words.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	**0.81**	0.72	0.76	0.99	0.54	0.92	**0.79**
LR	0.78	0.73	**0.77**	0.99	0.55	0.92	0.78
MNB	0.51	0.71	0.56	0.93	0.52	0.77	0.64
SGDC	0.50	0.72	0.50	0.93	0.50	0.50	0.50
KNN	0.74	0.58	0.71	0.74	0.57	0.81	0.73
RFC	0.74	0.81	0.70	0.99	0.57	0.91	0.78
SVDR	0.74	0.73	0.75	0.92	0.58	0.85	0.75
NMFR	0.74	0.73	0.73	0.92	**0.59**	0.85	0.74
GRU	0.70	0.61	0.51	0.98	0.54	0.80	0.66
BERT	**0.81**	**0.84**	**0.77**	0.50	0.50	**0.96**	**0.79**
Bi-LSTM	0.72	0.72	0.74	**1.00**	0.50	0.89	0.73
LSTM	0.73	0.69	0.51	0.98	0.57	0.87	0.71
Median	0.74	0.72	0.72	**0.95**	0.54	0.86	-

Table 4.3: Results for AUC-ROC using words.

Model	**COVID**	**SFNC**	**FNN**	**ISOT**	**LIAR**	**FCRN**	**Median**
SVM	**0.98**	0.73	**0.86**	0.99	0.57	0.93	**0.90**
LR	**0.98**	0.73	0.85	0.99	0.60	0.93	0.89
MNB	0.95	0.72	0.77	0.94	0.65	0.81	0.79
SGDC	0.95	0.73	0.74	0.93	0.65	0.59	0.74
KNN	0.97	0.59	0.81	0.90	0.60	0.80	0.81
RFC	0.97	0.81	0.84	0.99	0.62	0.92	0.88
SVDR	0.95	0.72	0.80	0.93	0.57	0.85	0.83
NMFR	0.95	0.72	0.80	0.93	0.59	0.85	0.83
GRU	0.97	0.61	0.74	0.98	0.67	0.82	0.78
BERT	**0.98**	**0.85**	0.84	0.52	0.65	**0.96**	0.85
Bi-LSTM	0.97	0.73	0.83	**1.00**	0.65	0.89	0.86
LSTM	0.97	0.76	0.74	0.98	**0.66**	0.89	0.83
Median	**0.97**	0.73	0.80	0.96	0.63	0.87	-

Table 4.4: Results for accuracy using AF.

Model	**COVID**	**SFNC**	**FNN**	**ISOT**	**LIAR**	**FCRN**	**Median**
SVM	0.84	0.73	**0.79**	0.99	0.53	0.93	**0.82**
LR	0.84	0.73	**0.79**	0.99	0.54	0.92	**0.82**
MNB	0.51	0.71	0.54	0.94	0.44	0.79	0.63
SGDC	0.49	0.73	0.43	0.93	0.39	0.37	0.46
KNN	0.78	0.57	0.74	0.90	**0.57**	0.80	0.76
RFC	0.81	0.81	0.73	0.99	0.55	0.92	0.81
SVDR	0.73	0.72	0.74	0.93	0.56	0.85	0.74
NMFR	0.73	0.72	0.73	0.93	**0.57**	0.85	0.73
GRU	0.77	0.60	0.46	0.98	0.55	0.81	0.69
BERT	**0.86**	**0.85**	0.78	0.34	0.39	**0.96**	**0.82**
Bi-LSTM	0.79	0.73	0.76	**1.00**	0.44	0.89	0.78
LSTM	0.80	0.75	0.46	0.98	0.55	0.88	0.78
Median	0.78	0.73	0.73	**0.96**	0.54	0.86	-

Table 4.5: Results for F1 using AF.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	**0.80**	0.73	**0.77**	0.99	0.53	0.92	**0.79**
LR	0.78	0.73	**0.77**	0.99	0.54	0.92	0.78
MNB	0.51	0.42	0.56	0.94	0.52	0.78	0.54
SGDC	0.50	0.73	0.50	0.93	0.50	0.50	0.50
KNN	0.74	0.58	0.73	0.90	0.57	0.81	0.74
RFC	0.73	0.81	0.70	0.99	0.55	0.92	0.77
SVDR	0.73	0.72	0.75	0.93	0.58	0.85	0.74
NMFR	0.73	0.72	0.73	0.93	**0.59**	0.85	0.73
GRU	0.70	0.61	0.51	0.98	0.57	0.81	0.66
BERT	**0.80**	**0.85**	0.76	0.50	0.50	**0.96**	0.78
Bi-LSTM	0.75	0.73	0.74	**1.00**	0.52	0.89	0.75
LSTM	0.74	0.76	0.51	0.98	0.56	0.88	0.75
Median	0.73	0.73	0.73	**0.96**	0.54	0.86	-

Table 4.6: Results for AUC-ROC using AF.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	**0.98**	0.76	**0.86**	**1.00**	0.58	0.95	**0.91**
LR	**0.98**	0.76	**0.86**	**1.00**	0.59	0.94	0.90
MNB	0.95	0.74	0.77	0.97	0.65	0.81	0.79
SGDC	0.95	0.76	0.74	0.94	0.65	0.59	0.75
KNN	0.97	0.63	0.81	0.65	0.62	0.83	0.73
RFC	0.97	0.80	0.83	**1.00**	0.64	0.93	0.88
SVDR	0.95	0.74	0.80	0.94	0.59	0.86	0.83
NMFR	0.95	0.74	0.80	0.94	0.58	0.86	0.83
GRU	0.97	0.60	0.74	0.99	**0.66**	0.82	0.78
BERT	**0.98**	**0.87**	0.85	0.77	0.64	**0.98**	0.86
Bi-LSTM	0.97	0.73	0.83	**1.00**	0.65	0.91	0.87
LSTM	0.97	0.73	0.74	0.99	0.65	0.89	0.81
Median	0.97	0.74	0.81	**0.98**	0.64	0.88	-

Table 4.7: Results for accuracy using RT.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	0.85	0.76	**0.79**	**1.00**	0.54	0.95	0.82
LR	0.84	0.76	**0.79**	**1.00**	0.55	0.94	0.82
MNB	0.49	0.73	0.52	0.97	0.42	0.79	0.63
SGDC	0.49	0.75	0.43	0.94	0.39	0.37	0.46
KNN	0.80	0.62	0.73	0.59	0.56	0.82	0.68
RFC	0.80	0.80	0.72	**1.00**	0.57	0.93	0.80
SVDR	0.74	0.74	0.75	0.94	0.57	0.85	0.75
NMFR	0.74	0.74	0.74	0.94	0.57	0.86	0.74
GRU	0.77	0.59	0.46	0.99	0.52	0.81	0.68
BERT	**0.88**	**0.87**	**0.79**	0.75	**0.59**	**0.98**	**0.83**
Bi-LSTM	0.79	0.73	0.75	**1.00**	0.57	0.90	0.77
LSTM	0.77	0.72	0.46	0.99	0.55	0.88	0.75
Median	0.78	0.74	0.74	**0.98**	0.55	0.87	-

Table 4.8: Results for F1 using RT.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	0.79	0.76	0.77	**1.00**	0.54	0.94	0.78
LR	0.77	0.76	0.77	**1.00**	0.55	0.94	0.77
MNB	0.50	0.73	0.55	0.97	0.51	0.78	0.64
SGDC	0.50	0.75	0.50	0.94	0.50	0.50	0.50
KNN	0.77	0.63	0.71	0.64	0.56	0.83	0.68
RFC	0.72	0.80	0.69	**1.00**	0.57	0.92	0.76
SVDR	0.76	0.74	0.76	0.94	0.58	0.86	0.76
NMFR	0.76	0.74	0.75	0.94	**0.59**	0.86	0.76
GRU	0.70	0.61	0.51	0.99	0.55	0.81	0.66
BERT	**0.84**	**0.87**	**0.79**	0.76	**0.59**	**0.98**	**0.82**
Bi-LSTM	0.72	0.73	0.74	**1.00**	0.57	0.90	0.74
LSTM	0.70	0.72	0.51	0.99	0.56	0.88	0.71
Median	0.74	0.74	0.72	**0.98**	0.56	0.87	-

Table 4.9: Results for AUC-ROC using RT.

In Tables 4.10 to 4.12, we observe the results for accuracy, F1 and AUC-ROC using GloVe as feature. For the COVID dataset, neural networks models such as GRU and LSTM obtained good results for the three metrics. In the same direction, for the FNN, ISOT and FCRN datasets, the Bi-LSTM model obtained the best results for accuracy, F1 and AUC-ROC. For the SNFC dataset, RFC obtained acceptable results in Tables 4.10 to 4.12 followed by the LSTM model. With the LIAR dataset, Bi-LSTM obtained good results in terms of accuracy, as we see in Table 4.10. On the other hand, the NMFR model obtained acceptable results for F1 and AUC-ROC for this same dataset. In the medians per model, Bi-LSTM obtained the best result, followed by SVDR. In the medians per dataset, the models obtained the best results for the ISOT dataset for all the metrics, which means this dataset is easy to classify.

From Tables 4.13 to 4.15 we show results for accuracy, F1 and AUC-ROC using fastText. For the COVID dataset, SVM, LR and Bi-LSTM models have the best results in terms of accuracy and F1. In this dataset, the reconstructive models, SVDR and NMFR, obtained good results for AUC-ROC. Bi-LSTM model obtained good results for the SNFC, ISOT and FCRN datasets in the three metrics. In the FNN dataset, Bi-LSTM has the best results in the three metrics, although RFC and NMFR have good results in accuracy and AUC-ROC, respectively. The best results using fastText for the LIAR datasets were obtained by LR for accuracy, KNN for F1 and NMFR for AUC-ROC. If we see the general medians per model, Bi-LSTM presents the best performance for all the metrics followed by NMFR. In the medians per dataset, the models have consistent results with the ISOT dataset for the three performance metrics. Nevertheless, the models also obtained good results with the FCRN and COVID datasets that are also easier to classify compared with the other datasets.

In Tables 4.16 to 4.18 we present the results for the three evaluation metrics using Word2Vec as feature. For the COVID dataset, models SVM, LR, KNN, GRU and Bi-LSTM obtained good results for accuracy. On the other hand, the best models for F1 and AUC-ROC were KNN and SVDR. SVM obtained good results in the three metrics, especially for the SFNC dataset. In the FNN dataset, models Bi-LSTM, SVDR and RFC obtained acceptable results in the three evaluation metrics, being the last one the best for accuracy. Bi-LSTM, again, has the best results with the ISOT dataset. For the LIAR dataset, a reconstructive model such as NMFR has the best results in F1 and AUC-ROC. Additionally, models SVM, LR, MNB, SGDC, GRU and LSTM have the best results in accuracy. For the FCRN dataset, the model with the best results in all the metrics was Bi-LSTM. In the medians per model, we see that RFC obtained the best result for accuracy. Meanwhile, for F1 and AUC-ROC, SVM and NMFR obtained the best performance, respectively. For the medians per dataset, the models obtained good results for the ISOT dataset especially for F1 and AUC-ROC, which means that disregarding of the classification approach this dataset is easier to classify.

Model	**COVID**	**SFNC**	**FNN**	**ISOT**	**LIAR**	**FCRN**	**Median**
SVM	0.96	0.68	0.80	0.94	0.65	0.79	0.80
LR	0.96	0.67	0.80	0.93	0.65	0.78	0.79
MNB	0.95	0.67	0.74	0.52	0.65	0.59	0.66
SGDC	0.95	0.52	0.74	0.52	0.65	0.59	0.62
KNN	**0.97**	0.63	0.80	0.93	0.59	0.77	0.79
RFC	0.96	**0.73**	0.82	0.95	0.61	0.80	0.81
SVDR	0.89	0.66	0.76	0.87	0.57	0.77	0.77
NMFR	0.91	0.67	0.75	0.86	0.60	0.76	0.76
GRU	**0.97**	0.60	0.74	0.93	0.65	0.84	0.79
BERT	-	-	-	-	-	-	
Bi-LSTM	**0.97**	0.67	**0.83**	**1.00**	**0.66**	**0.91**	**0.87**
LSTM	**0.97**	0.72	0.77	0.92	0.65	0.84	0.81
Median	**0.96**	0.67	0.77	0.93	0.65	0.78	-

Table 4.10: Results for accuracy using GloVe.

Model	**COVID**	**SFNC**	**FNN**	**ISOT**	**LIAR**	**FCRN**	**Median**
SVM	0.62	0.68	0.63	0.94	0.39	0.78	0.66
LR	0.62	0.67	0.66	0.93	0.39	0.77	0.67
MNB	0.49	0.67	0.43	0.34	0.39	0.37	0.41
SGDC	0.49	0.34	0.43	0.34	0.39	0.37	0.38
KNN	0.78	0.63	0.73	0.92	0.56	0.76	0.75
RFC	0.63	**0.73**	0.71	0.95	0.55	0.79	0.72
SVDR	0.66	0.66	0.70	0.87	0.55	0.77	0.68
NMFR	0.68	0.67	0.69	0.86	**0.59**	0.75	0.69
GRU	**0.79**	0.59	0.47	0.93	0.51	0.84	0.69
BERT	-	-	-	-	-	-	-
Bi-LSTM	0.77	0.67	**0.76**	**1.00**	0.50	**0.90**	**0.77**
LSTM	0.76	0.71	0.64	0.92	0.52	0.83	0.74
Median	0.66	0.67	0.66	**0.92**	0.51	0.77	-

Table 4.11: Results for F1 using GloVe.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	0.57	0.69	0.62	0.94	0.50	0.77	0.66
LR	0.57	0.67	0.64	0.93	0.50	0.77	0.66
MNB	0.50	0.67	0.50	0.50	0.50	0.50	0.50
SGDC	0.50	0.50	0.50	0.50	0.50	0.50	0.50
KNN	0.75	0.63	0.71	0.93	0.56	0.76	0.73
RFC	0.59	**0.73**	0.68	0.95	0.55	0.79	0.71
SVDR	**0.81**	0.66	0.71	0.87	0.56	0.78	**0.75**
NMFR	0.78	0.67	0.70	0.86	**0.61**	0.76	0.73
GRU	0.72	0.61	0.51	0.93	0.54	0.86	0.67
BERT	-	-	-	-	-	-	-
Bi-LSTM	0.70	0.68	**0.73**	**1.00**	0.54	**0.90**	0.72
LSTM	0.70	0.72	0.63	0.92	0.55	0.83	0.71
Median	0.66	0.67	0.66	**0.92**	0.51	0.77	-

Table 4.12: Results for AUC-ROC using GloVe.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	**0.97**	0.73	0.82	0.97	0.65	0.84	0.83
LR	**0.97**	0.71	0.82	0.96	**0.66**	0.83	0.83
MNB	0.95	0.65	0.74	0.52	0.65	0.59	0.65
SGDC	0.95	0.48	0.74	0.52	0.35	0.59	0.56
KNN	0.96	0.69	0.80	0.94	0.63	0.78	0.79
RFC	0.96	0.70	**0.83**	0.96	0.59	0.81	0.82
SVDR	0.90	0.66	0.74	0.89	0.59	0.79	0.77
NMFR	0.89	0.66	0.76	0.89	0.59	0.79	0.78
GRU	**0.97**	0.65	0.76	0.95	0.66	0.83	0.80
BERT	-	-	-	-	-	-	
Bi-LSTM	**0.97**	**0.75**	**0.83**	**1.00**	0.65	**0.90**	**0.87**
LSTM	0.96	0.69	0.77	0.92	0.65	0.84	0.81
Median	**0.96**	0.69	0.77	0.94	0.65	0.81	-

Table 4.13: Results for accuracy using fastText.

Model	**COVID**	**SFNC**	**FNN**	**ISOT**	**LIAR**	**FCRN**	**Median**
SVM	**0.74**	0.73	0.72	0.97	0.39	0.83	0.74
LR	**0.74**	0.71	0.73	0.96	0.54	0.82	0.74
MNB	0.49	0.65	0.43	0.34	0.39	0.37	0.41
SGDC	0.49	0.32	0.43	0.34	0.26	0.37	0.36
KNN	**0.74**	0.68	0.72	0.94	**0.58**	0.77	0.73
RFC	0.61	0.70	0.71	0.96	0.54	0.80	0.71
SVDR	0.68	0.66	0.70	0.89	0.57	0.78	0.69
NMFR	0.67	0.66	0.71	0.89	0.57	0.79	0.69
GRU	0.71	0.64	0.55	0.95	0.52	0.83	0.68
BERT	-	-	-	-	-	-	-
Bi-LSTM	**0.74**	**0.75**	**0.76**	**1.00**	0.55	**0.89**	**0.76**
LSTM	0.72	0.68	0.63	0.92	0.52	0.83	0.70
Median	0.71	0.68	0.71	**0.94**	0.54	0.80	-

Table 4.14: Results for F1 using fastText.

Model	**COVID**	**SFNC**	**FNN**	**ISOT**	**LIAR**	**FCRN**	**Median**
SVM	0.67	0.73	0.69	0.97	0.50	0.82	0.71
LR	0.67	0.71	0.70	0.96	0.56	0.82	0.71
MNB	0.50	0.65	0.50	0.50	0.50	0.50	0.50
SGDC	0.50	0.50	0.50	0.50	0.50	0.50	0.50
KNN	0.71	0.69	0.71	0.94	0.57	0.77	0.71
RFC	0.57	0.70	0.68	0.96	0.54	0.80	0.69
SVDR	**0.82**	0.67	0.71	0.89	0.58	0.79	0.75
NMFR	**0.82**	0.67	**0.73**	0.89	**0.59**	0.79	**0.76**
GRU	0.64	0.66	0.57	0.95	0.55	0.83	0.65
BERT	-	-	-	-	-	-	
Bi-LSTM	0.68	**0.75**	**0.73**	**1.00**	0.56	**0.89**	0.74
LSTM	0.67	0.70	0.62	0.92	0.54	0.83	0.69
Median	0.67	0.69	0.69	**0.94**	0.55	0.80	-

Table 4.15: Results for AUC-ROC using fastText.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	**0.97**	**0.77**	0.83	0.96	**0.65**	0.82	0.83
LR	**0.97**	0.75	0.82	0.95	**0.65**	0.81	0.82
MNB	0.95	0.67	0.71	0.52	**0.65**	0.59	0.66
SGDC	0.95	0.52	0.77	0.52	**0.65**	0.59	0.62
KNN	**0.97**	0.69	0.93	0.94	0.61	0.78	0.86
RFC	0.96	0.75	**0.95**	0.96	0.61	0.80	**0.88**
SVDR	0.92	0.67	0.88	0.89	0.55	0.79	0.84
NMFR	0.92	0.67	0.88	0.89	0.59	0.78	0.83
GRU	**0.97**	0.60	0.92	0.95	**0.65**	0.81	0.87
BERT	-	-	-	-	-	-	-
Bi-LSTM	**0.97**	0.72	0.82	**1.00**	0.65	**0.88**	0.85
LSTM	**0.97**	0.68	0.89	0.92	**0.65**	0.82	0.86
Median	**0.97**	0.68	0.88	0.94	0.65	0.80	-

Table 4.16: Results for accuracy using Word2Vec.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	0.76	**0.77**	0.72	0.96	0.39	0.81	**0.77**
LR	0.75	0.75	0.73	0.95	0.52	0.81	0.75
MNB	0.49	0.67	0.43	0.68	0.39	0.37	0.46
SGDC	0.49	0.34	0.43	0.76	0.39	0.37	0.41
KNN	**0.80**	0.68	0.71	0.93	0.55	0.77	0.74
RFC	0.66	0.75	0.70	0.95	0.55	0.79	0.73
SVDR	0.70	0.67	0.71	0.88	0.54	0.78	0.71
NMFR	0.72	0.67	0.69	0.88	**0.58**	0.77	0.71
GRU	0.75	0.64	0.46	0.92	0.50	0.80	0.70
BERT	-	-	-	-	-	-	-
Bi-LSTM	0.75	0.72	**0.75**	**1.00**	0.53	**0.87**	0.75
LSTM	0.74	0.68	0.52	0.89	0.50	0.81	0.71
Median	0.74	0.68	0.70	**0.92**	0.52	0.79	-

Table 4.17: Results for F1 using Word2Vec.

Model	COVID	SFNC	FNN	ISOT	LIAR	FCRN	Median
SVM	0.69	**0.77**	0.70	0.97	0.50	0.81	0.74
LR	0.69	0.75	0.70	0.96	0.55	0.80	0.73
MNB	0.50	0.68	0.50	0.50	0.50	0.50	0.50
SGDC	0.50	0.50	0.50	0.50	0.50	0.50	0.50
KNN	0.76	0.69	0.70	0.94	0.55	0.77	0.73
RFC	0.60	0.75	0.67	0.96	0.55	0.78	0.71
SVDR	**0.82**	0.67	**0.73**	0.89	0.56	0.79	0.76
NMFR	0.84	0.67	0.72	0.89	**0.59**	0.78	**0.75**
GRU	0.68	0.61	0.51	0.95	0.54	0.80	0.65
BERT	-	-	-	-	-	-	-
Bi-LSTM	0.68	0.72	**0.73**	**1.00**	0.55	**0.88**	0.73
LSTM	0.67	0.69	0.54	0.92	0.54	0.80	0.68
Median	0.68	0.69	0.70	**0.94**	0.55	0.79	-

Table 4.18: Results for AUC-ROC using Word2Vec.

In Table 4.19, we present a summary of the best combination of models and features for each dataset for all the metrics. For the COVID dataset, we observe that the combination SVM+Words obtained good results for accuracy. On the other hand, BERT+RT and NMFR+Word2Vec obtained acceptable results for F1 and AUC-ROC, respectively. For SNFC and FCRN datsets, the combination of BERT+RT obtained good results for all the metrics, which means that these datasets are easier to classify using this combination. For the FNN dataset, RFC+Word2Vec obtained good results for accuracy. On the other hand, SVM+Words and BERT+RT obtained good results for F1 and AUC-ROC for this dataset. In the ISOT dataset, a combination of Bi-LSTM+Words obtained the best results for this dataset in all the metrics. For the LIAR dataset, GRU+AF obtained good restults in terms of accuracy, meanwhile a combination of NMFR+GloVe obtained good results for F1 and AUC-ROC.

Dataset+Model+Feature	Accuracy	Dataset+Model+Feature	F1	Dataset+Model+Feature	AUC-ROC
COVID+SVM+Words	0.98	**COVID+BERT+RT**	0.88	**COVID+NMFR+Word2Vec**	0.84
SFNC+BERT+RT	0.87	**SNFC+BERT+RT**	0.87	**SFNC+BERT+RT**	0.87
FNN+RFC+Word2Vec	0.95	**FNN+SVM+Words**	0.79	**FNN+BERT+RT**	0.79
ISOT+Bi-LSTM+Words	1.00	**ISOT+Bi-LSTM+Words**	1.00	**ISOT+Bi-LSTM+Words**	1.00
LIAR+GRU+AF	0.67	**LIAR+NMFR+GloVe**	0.59	**LIAR+NMFR+GloVe**	0.61
FCRN+BERT+RT	0.98	**FCRN+BERT+RT**	0.98	**FCRN+BERT+RT**	0.98

Table 4.19: Results of the best combinations of dataset+model+feature.

In Table 4.20, we see a summary of the combination of models and features sets that obtained the best results per metric. In this table, we see that BERT, in combination with RT, obtained the best results for all the metrics. On the other hand, SVM and the combination of words or AF also presented good results.

In general, models present better results when they are combined with superficial feature sets, such as words. For example, the combination between SVM and LR with words has good results. On the other hand, when BERT and RT are combined, we obtain good results for the SFNC and FCRN datasets. For the deep features sets, the combination between SVM+Word2Vec and NMFR+Word2Vec obtained acceptable results in most of the datasets.

Model+Feature	**Accuracy**	**Model+Feature**	**F1**	**Model+Feature**	**AUC-ROC**
SVM+Words	0.90	**SVM+Words**	0.82	**SVM+Words**	0.79
SVM+AF	0.90	**LR+AF**	0.82	**SVM+AF**	0.79
SVM+RT	**0.91**	**BERT+RT**	**0.83**	**BERT+RT**	**0.82**
Bi-LSTM+Glove	0.87	**Bi-LSTM+GloVe**	0.77	**SVDR+GloVe**	0.75
Bi-LSTM+fastText	0.87	**Bi-LSTM+fastText**	0.76	**NMFR+fastText**	0.76
RFC+Word2Vec	0.88	**SVM+Word2Vec**	0.77	**NMFR+Word2Vec**	0.75

Table 4.20: Results of the best combinations of model+feature.

Finally, in Table 4.21 we present a comparison between the results obtained in the state-of-the-art using the corresponding dataset and the models+feature we constructed. For the SNFC dataset, we used a combination of BERT+RT that obtained better results compared with [18] for accuracy and F1. For the ISOT dataset, we obtained better results in accuracy when we used Bi-LSTM+words; this combination achieved better results than the results presented in [12]. On the other hand, the results obtained in [11] for the LIAR dataset are slightly higher that the ones we obtained in our experiments.

	State-of-the-art			**Our approaches**		
Dataset	**Method**	**Accuracy**	**F1**	**Method**	**Accuracy**	**F1**
SNFC	BERT+word n-grams+char n-grams [18]	0.85	0.85	BERT+RT	0.87	0.87
ISOT	Linear SVM+unigrams [12]	0.92	-	Bi-LSTM+Words	1.00	-
LIAR	SVM+n-grams [11]	-	0.61	NMFR+GloVe	-	0.59

Table 4.21: Comparison of results from the state-of-the-art and our approaches.

Chapter 5

Conclusions

In this thesis, we presented an analysis of different machine learning and deep learning models for the task of fake news detection in online media. Identifying fake news would help to benefit the news organizations and society to have a better news ecosystem. For organizations, identifying fake news would help them have a higher level of quality on the news these organizations post. In the case of society, it would help them formulate arguments based on real information and do not create biased ideas. For the experimentation phase, we collected six datasets: COVID, SFNC, FNN, ISOT, LIAR and FCRN. These datasets were formed by the textual content of news from different news sources such as CNN, El Universal, Fox News, among others. Similarly, some of these datasets contain information collected from popular social networks such as Facebook and Twitter. We extracted a series of features sets that lately we combined with different machine learning and deep learning models.

From the results that we obtained in our experimentation, we can conclude the following:

- In terms of F1 macro and AUC-ROC, classical machine learning models such as SVM and LR present good results when combined with words and RT, especially for the COVID and FNN datasets.
- Although features such as emojis, hashtags, links, ats and abbreviations, are not frequently used in the news. We can see that feature sets such as AF and RT performed well in terms of F1 when these features were combined with BERT, SVM and LR.
- Bi-LSTM, in combination with the majority of feature sets, obtains the highest results for all the metrics in the ISOT dataset. Similarly, SVM, LR and RFC combined with words and RT obtain good results for this same dataset for F1. Nevertheless, it is important to mention that this dataset contains a greater number of instances than the other datasets, which helps during the training process to build more robust models.
- In the datasets SNFC and FCRN, the BERT model obtained the best results in combination with RT, although words and AF also obtained good results. In the case of SNFC, RFC also obtained good results when is combined with words and AF. For FCRN, SVM and LR obtained acceptable results when are combined with AF and RT features.

- Reconstructive models such as SVDR and NMFR present good results with the combination of Word2Vec and GloVe for F1 and AUC-ROC in the LIAR dataset. In addition, in this dataset, BERT obtained good results when combined with the RT features set.
- In general, the use of superficial features such as words, AF and RT, has a better performance than the deep features for the three performance metrics.

We can conclude that the detection of fake news is a task that has gained relevance in recent years. Nevertheless, fake news identification is a hard task to solve since there are a variety of topics that can be covered in the news.

For future research we can include the identification of fake news about specific topics covered in the news. Similarly, the combination of different models where deep learning models can be used to extract features from the text and machine learning models can be employed in the classification phase or vice versa. Additionally, the probabilities produced by the models could be combined to produce more precise results.

Bibliography

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

[2] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake News Detection on Social Media: A Data Mining Perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 9 2017.

[3] William Yang Wang. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 422–426, Stroudsburg, PA, USA, 2 2017. Association for Computational Linguistics.

[4] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. FakeNewsNet: A Data Repository with News Content, Social Context, and Spatiotemporal Information for Studying Fake News on Social Media. *Big Data*, 8(3):171–188, 9 2020.

[5] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detecting opinion spams and fake news using text classification. *Security and Privacy*, 1(1):e9, 2018.

[6] Juan-Pablo Posadas-Durán, Helena Gómez-Adorno, Grigori Sidorov, and Jesús Jaime Moreno Escobar. Detection of fake news in a new corpus for the Spanish language. *Journal of Intelligent & Fuzzy Systems*, 36(5):4869–4876, 5 2019.

[7] Hunt Allcott and Matthew Gentzkow. Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives*, 31(2):211–236, 5 2017.

[8] Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. Combating Fake News: A Survey on Identification and Mitigation Techniques. *ACM Transactions on Intelligent Systems and Technology*, 10(3):1–42, 5 2019.

[9] Mykhailo Granik and Volodymyr Mesyura. Fake news detection using naive Bayes classifier. *2017 IEEE 1st Ukraine Conference on Electrical and Computer Engineering, UKRCON 2017 - Proceedings*, pages 900–903, 2017.

[10] Karishnu Poddar, Geraldine Bessie Amali D., and K.S. Umadevi. Comparison of Various Machine Learning Models for Accurate Detection of Fake News. In *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–5. IEEE, 3 2019.

[11] Vasu Agarwal, H. Parveen Sultana, Srijan Malhotra, and Amitrajit Sarkar. Analysis of Classifiers for Fake News Detection. *Procedia Computer Science*, 165:377–383, 2019.

[12] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In Issa Traore, Isaac Woungang, and Ahmed Awad, editors, *First International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, volume 10618 of *Lecture Notes in Computer Science*, pages 127–138. Springer International Publishing, Cham, 2017.

[13] Pritika Bahad, Preeti Saxena, and Raj Kamal. Fake News Detection using Bi-directional LSTM-Recurrent Neural Network. *Procedia Computer Science*, 165:74–82, 2019.

[14] Rohit Kumar Kaliyar, Anurag Goswami, Pratik Narang, and Soumendu Sinha. FNDNet – A deep convolutional neural network for fake news detection. *Cognitive Systems Research*, 61:32–44, 6 2020.

[15] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 849–857, New York, NY, USA, 7 2018. ACM.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.

[17] Mario Ezra Aragón, Horacio Jarquín-Vasqueza, Manuel Montes-Y-Gómez, Hugo Jair Escalante, Luis Villasenõr-Pineda, Helena Gómez-Adorno, Juan Pablo Posadas-Durán, and Gemma Bel-Enguix. Overview of mex-a3t at iberlef 2020: Fake news and aggressiveness analysis in mexican Spanish. *CEUR Workshop Proceedings*, 2664(September):222–235, 2020.

[18] Esaú Villatoro-Tello, Gabriela Ramírez-De-La-Rosa, Sajit Kumar, Shantipriya Parida, and Petr Motlicek. Idiap and UAM Participation at MEX-A3T Evaluation Campaign. *CEUR Workshop Proceedings*, 2664(January):252–257, 2020.

[19] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1(Mlm):4171–4186, 2019.

[20] Diego Zaizar-Gutiérrez, Daniel Fajardo-Delgado, and Miguel A. Álvarez-Carmona. ITCG's participation at MEX-A3t 2020: Aggressive identification and fake news detection based on textual features for mexican Spanish. *CEUR Workshop Proceedings*, 2664:258–264, 2020.

[21] Samuel Arce-Cardenas, Daniel Fajardo-Delgado, and Miguel Álvarez-Carmona. TecNM at MEX-A3t 2020: Fake news and aggressiveness analysis in mexican Spanish. *CEUR Workshop Proceedings*, 2664:265–272, 2020.

[22] Jeffrey E. F. Friedl and Andy Oram. *Mastering Regular Expressions.* O'Reilly & Associates, Inc., USA, 2 edition, 2002.

[23] W. John Wilbur and Karl Sirotkin. The automatic identification of stop words. *Journal of Information Science*, 18(1):45–55, 2 1992.

[24] Edward Loper and Steven Bird. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions -*, pages 69–72, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[25] H. P. Luhn. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*, 1(4):309–317, 10 1957.

[26] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 31, pages 1532–1543. Association for Computational Linguistics, 2014.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pages 1–12, 2013.

[28] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[29] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 9 1995.

[30] Tin Kam Ho. Random decision forests. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1:278–282, 1995.

[31] Evelyn Fix and J. L. Hodges. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238, 1989.

[32] Juan Carlos Gomez and Marie Francine Moens. PCA document reconstruction for email classification. *Computational Statistics and Data Analysis*, 56(3):741–751, 2012.

[33] Juan Carlos Gomez and Marie Francine Moens. Document categorization based on minimum loss of reconstruction information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7630 LNAI(PART 2):91–103, 2013.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[35] John F. Kolen and Stefan C. Kremer. Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. *A Field Guide to Dynamical Recurrent Networks*, 2010.

[36] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[37] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1724–1734, 2014.

[38] Nitin Indurkhya and Fred J. Damerau. *Handbook of Natural Language Processing.* Chapman and Hall/CRC, New York, NY, USA, 2nd edition, 2 2010.

[39] Cathy O'Neil and Rachel Schutt. *Doing Data Science.* O'Reilly Media, Inc., 1 edition, 2013.

[40] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques.* Elsevier, 3rd edition, 2011.

[41] Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. What does this Emoji Mean? A Vector Space Skip-Gram Model for Twitter Emojis. *Proceedings of the Tenth International Conference on Language Resources and Evaluation ({LREC}'16)*, 36(7):3967–3972, 2016.

[42] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pages 3483–3487, 2019.

[43] Ikuya Yamada and Hiroyuki Shindo. Neural attentive bag-of-entities model for text classification. *CoNLL 2019 - 23rd Conference on Computational Natural Language Learning, Proceedings of the Conference*, pages 563–573, 2019.

[44] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Perez. Spanish Pre-Trained BERT Model. *Workshop paper at PML4DC, ICLR*, pages 1–10, 2020.

[45] Arun S. Maiya. ktrain: A Low-Code Library for Augmented Machine Learning. *arXiv preprint arXiv:2004.10703*, 10703, 2020.