UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF A FAKE NEWS DETECTION SYSTEM USING MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES

BEATRIZ HERNÁNDEZ-FONTA CODESIDO

ENERO 2024

TRABAJO DE FIN DE GRADO

Título:	Desarrollo de un sistema con Aprendizaje Automático y Procesamiento Natural del Lenguaje para detectar noticias falsas
Título (inglés):	Development of a Fake News Detection System using Machine Learning and Natural Language Processing Techniques
Autor:	Beatriz Hernández-Fonta Codesido
Tutor:	Óscar Araque Iborra
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF A FAKE NEWS DETECTION SYSTEM USING MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES

Beatriz Hernández-Fonta Codesido

Enero 2024

Resumen

En los últimos años, con las crecientes innovaciones tecnológicas, el número de fuentes de información ha aumentado considerablemente. En la sociedad actual, en la que todos estamos ampliamente conectados, las noticias se difunden rápidamente y cualquiera puede publicar un artículo. Este hecho ha facilitado el crecimiento de las llamadas "noticias falsas". Hemos llegado a un punto en el que, ante una noticia, dudamos de su autenticidad. La publicación de noticias falsas conduce a la desinformación, a la mala toma de decisiones y puede ser realmente perjudicial, especialmente cuando está relacionado con determinados temas como la salud, la política, la religión o incluso la mala prensa sobre una determinada empresa o persona.

El principal objetivo de este proyecto es explorar el uso de modelos de aprendizaje para detectar "fake news". El desarrollo de estos modelos permitirá identificar las "fake news" y frenar su propagación, mejorando el entorno digital convirtiendolo en una fuente de información más fiable y un espacio más seguro para investigar y adquirir conocimientos.

Para lograr llevar a cabo estos desarrollos empezaremos recopilando distintas fuentes de datos con noticias ya clasificadas como falsas o no. Los datos recopilados se preprocesaron mediante técnicas de procesamiento del lenguaje natural, como la eliminación de palabras vacías y signos de puntuación, la lematización y la tokenización. Para continuar, se vectorizan los datos y por último, utilizando diferentes técnicas de aprendizaje automático se crearon y entrenaron varios modelos con el fin de obtener la mayor precisión posible.

Todo esto se llevará a cabo utilizando Python con varias de sus librerías pero principalmente Scikit-Learn, una librería que proporciona múltiples algoritmos de aprendizaje automático, construida sobre SciPy y que particularmente utiliza NumPy para realizar los arrays y Pandas para el análisis de datos. Además utilizaremos Hugging Face para poder implementar el modelo de Transformers BERT.

Palabras clave: Fake news, Autenticidad, Desinformation, Aprendizaje Automático, NLP, Python, Scikit-Learn, Scipy

Abstract

Over the last few years with the increasing technological innovations the number of sources of information have significantly increased. In today's society, where we are all widely connected, news spread quickly and anybody can publish a news article. Such an ease has facilitated the growth of the so-called "fake news". We are at a point where confronted with a piece of news, we have doubts regarding its authenticity. Publishing fake news leads to misinformation, bad decision making and it could be really harmful, especially when it is related to certain topics such as health, politics, religion or even bad press about a certain company or person. That is also known as disinformation and large-scale campaigns about it have become a major challenge for Europe, as a result the commission has developed numerous initiatives to tackle it.

The main objective of this project is to explore the use of learning models to detect "fake news". With the development of these models "fake news" will be identified and stopped from spreading, leading to an improvement on the digital environment making it a more reliable source of information and a safer place to research and acquire knowledge .

To make this possible, different sources of data with news already classified as fake or not will be collected. The gathered data was firstly preprocessed using natural language processing techniques such as removing stop words and punctuation, lemmatizing and tokenizing. To continue, different data representations were presented, and at last, using different machine learning techniques various models were created and trained in order to get the best possible accuracy.

In order to carry out the procedures mentioned above, Python has been used with several of its libraries but mainly Scikit-Learn. Scikit-Learn is a library which provides multiple Machine Learning algorithms, it's built upon SciPy and particularly uses NumPy for making the arrays and pandas for the data analysis. A part from that we will make use of Hugging Face to enable implementing the Transformers model BERT.

Keywords:

Fake news, Authenticity, Misinformation, Machine Learning, NLP, Python, Scikit-Learn, Scipy

Agradecimientos

Ahora que mis años de estudiante están llegando a su fin no puedo evitar mirar hacia atrás y recordar con cariño todos los momentos que he vivido en la que ha sido la etapa más enriqucedora, bonita y desafiante que he vivido.

No puedo decir que el camino haya sido fácil, pero si que ha merecido la pena. Estoy muy agradecida a la Universidad Politecnica de Madrid y en especial a la Escuela Técnica Superior de Ingenieros de Telecomunicación y a todas las personas que forman parte de ella, en especial a todo el personal docente por el apoyo brindado y los conocimientos que me han transmitido a lo largo de los años de carrera.

Agradecer también a mi tutor, Óscar Araque, por guiar mi Trabajo de Fin de Grado y darme la oportunidad de profundizar sobre un tema tan ineresante y actual, ayudandome en todo momento para sacar el máximo potencial.

Durante estos años ha sido tremendamente importante el apoyo de mis amigos, siempre dispuestos a ayudarnos, sacado lo mejor los unos de los otros y acompañandonos durante esta etapa de la que tanto hemos disfutado.

Aunque sin duda a quien mas debo dar las gracias es a mi familia. Gracas por animarme a tomar esta decisión y creer en mi en todo momento, hasta cuando yo misma dudaba. Gracias por haber estado siempre dispuestos a ayudarme en todo lo posible y animarme a no renderime nunca, creyendo en mi siempre de manera incondicional.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

The concept of "fake news" is not something new, on the contrary, it has been around since the times of the Roman Empire [1] when Octavian in order to have the public on his side spread false rumors about Anthony, thanks to that he got the victory on the elections. Nowadays the Information and Communication Technologies (ICTs) [2] have transformed the way we interact in profound ways, particularly the communications field has been deeply affected by digital media and technology. This new environment facilitates persuasive ways for advertisers and media organisations to communicate with audiences. In this new digital era social media sites are now the recurring sources of information and that has led to a competition between professional journalists and amateur publishers for readers attention. This new environment has played a very important role in the proliferation of "fake news".

As for the definition of the so called "fake news", it has not changed much since the first time it was used. "Fake news" consist on false or misleading information presented as news usually to influence people on such topics as politics or health or even as a joke.

When referring to "fake news" we must take into consideration the difference between misinformation and disinformation [3] . Misinformation occurs when the content of the new

is false but the author is not aware of it, therefore the harm that information may cause is not intended. On the other hand, when we refer to disinformation the author has generated a piece of information based on untruthful information in a deliberate manner, usually as a motivation to influence on the public opinion about certain topics. Although in both cases a "fake new" is created and the impact it has can be the same, the difference exists on the intentions of the author to cause harm or not.

"Fake news" are key to influence the public decision in politics, health or the way we perceive public figures. The impact a news article has depends heavily on the source from which it comes from, the more recognized it is, the more credibility it will have. The effect of a news item in a major newspaper is very different from perhaps an anonymous profile on any social media platform. Nowadays, in the digital era we not only face the problem of being exposed to fake news but we are also confronted with the speed at which they spread due to the highly connected society we live in. Most times when a fake new is created it is strategically design to go viral, the more it spreads, the bigger the repercussion it causes. Clickbait and eye-catching headlines containing misinformation are key to attract the public attention, that generates multiple clicks on advertisements and the more clicks it gets more money they make.

The consequences [1] of fake news are far-reaching, an example of that could be the 2016 US presidential elections during which the term "fake news" gained a lot of popularity because of the tremendous amount of them that were generated to influence the citizens. Back in 2020 during the COVID-19 pandemic, "fake news" also increased significantly as social media was full of theories about how the virus started, how it affected us or how to avoid contagion. Both cases illustrate how quickly those news went viral and how they affected the decision making of people, to the point of putting at risk their own health.

The dimensions of the problem are such that Europe has launched numerous campaigns to give visibility to this problem, appealing to critical thinking. The problem is even bigger when it comes to the younger generations, as they are those who make a more intensive use of social media. Although raising awareness about the existing problem is fundamental, due to the tremendous amount of information circulating on the internet, the sophisticated designs of fake news to make them apparently real and the speed at which they go viral that is not enough. Therefore the latest techniques are being used to try fighting this problem.

Currently a lot of work is being done in this field in order to make the internet a safer place and provide users an environment in which to find quality content and reduce the amount of false information circulating.

This project aims to help with this problem that has such a big impact in our society.

By using the latest technologies in Machine Learning and Natural Language Processing different news will be analyzed in order to determine if they are fake or not. This work starts by gathering enough datasets to get multiple results to work with looking for a more accurate conclusion when analyzing the obtained values. Those datasets will firstly be train by different models and then tested to obtain the results that will help to elaborate a conclusion about this development.

1.2 Project goals

This project has the following goals:

1. To gather multiple datasets in order to properly model the challenge of fake news detection.

2. Comparing different classification models by training and testing a few of them with various level of complexity by using the sources of information previously found.

3. Evaluate the different results obtained from the developed models and analyze their capabilities in order to collect information on the task at stake.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is as follows:

Chapter 1 Introduction: It presents the topic of the project, explaining its context and giving and overview the goals willing to archive and how it is structured.

Chapter 2 Background: The technologies and the different environments used to carry out the project are explained, also related work in the field is presented.

Chapter 3 Models: The different models used to train and test the datasets are explained in depth.

Chapter 4 Evaluation: This section gathers the results of the models previously described explaining the materials, the metrics, and the design used.

Chapter 5 Conclusion: To conclude, in this chapter we evaluate the objectives we first presented and how they have been archived. As well as that, the results will be analyzed

reaching a conclusion on which one is the most suitable for the problem we aim to tackle. At last future work in this topic is displayed.

CHAPTER 2

Background

2.1 Enabling Technologies

2.1.1 Introduction

In this chapter we will introduce the different technologies and tools used in the development of this project. Firstly we will dedicate a section to describe in detail each of the technologies used. Following that, we will specify the various tools that have contributed to the development of the project explaining them all. At last there is a section in which information has been compiled on different works in this same field, in this section we comment on the techniques used and their relationship with our work.

2.1.2 Python

Python [4] is a high-level programming language distinguished by its interpreted nature, object-oriented design and dynamic semantics. Its appeal lies in its combination of high-level built-in data structures, dynamic typing and dynamic linking. It works well as a scripting or glue language, facilitating the connection of existing components. The language's focus on simplicity and readability of its syntax reduces the cost of maintaining programs. Python

supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and an extensive standard library are freely available in source and binary formats for all major platforms, allowing unrestricted distribution.

In this first subsection we are going to explain the various Python libraries that were used to create this project. Each library was used with a different purpose such as representing, loading, organizing, pre-processing or applying different models to the data. Those libraries were Pandas, NLTK, SciKit, MatplotLib and Pytorch.

Pandas [5] is a powerful Python library entirely designed for data manipulation and analysis, it stands out for being flexible and easy to use. The development of the library began on n 2008 at AQR Capital Management and since the end of 2009 it has been open sourced.

In besides its role in data analysis, Pandas [6] is widely used for data manipulation, a process that involves techniques for converting disorganized data into a more usable format. It is capable of handling well-organized and structured data in the form of tables, arrays or time series. In addition, it integrates seamlessly with several Python libraries, increasing its versatility and potential for collaboration with the Python ecosystem.

The library is based on "DataFrames" which are two-dimensional arrays of data consisting of columns with the value of the variable and rows containing a set of values for those variables. When importing or exporting data it allows both CSV and JSON formats.

Natural Language Toolkit (NLTK) library [7] is one of the most widely used libraries for natural language processing in Python. It provides an easy-to-use interface for text preprocessing through a wide range of tasks, including tokenization, stemming and lemmatization. Text preprocessing is a crucial step, as it helps to clean and normalize the text data, making it easier to analyze.

- Tokenization involves breaking down the text into individual words or tokens, essential to separate individual words from the raw text.
- Stopwords involves removing common and irrelevant words which do not carry much meaning. NLTK provides a built-in list of stop words for several languages.
- Stemming means removing the suffixes from words reducing them to their base form.
- Lemmatization involves reducing words to their base form based on their part of speech.

Scikit-Learn [8] is a Python library that provides multiple Machine Learning algorithms

including regression, classification, dimensionality reduction, and clustering. The library is built upon SciPy (Scientific Python) that should be installed before using Scikit-learn. It particularly uses NumPy, a package for managing n-dimensional arrays, and Pandas which has been explained above.

The algorithms mentioned above are described below:

- Regression [9]: Typically employed to recognize cause-effect connections, predict trends, perform time series prediction analysis and evaluate the strength of predictors, regression produces results in the form of continuous data. By focusing on features, this method allows you to anticipate patterns within the training data, producing a numerical result. Some of the available regression algorithms are linear regression,logistic regression, Bayesian Linear Regression and Decision Tree Regression.
- Classification: It categorises data based on certain features. The output is discrete data. Some the algorithms used for this purpose are decision trees, SVM, Random forest and Perceptron.
- Dimensionality reduction: This works by reducing the amount of random variables to consider. SVD and PCA are examples of this sort of algorithms.
- Clustering: Applied in situations where the target or outcome variable is not specified in the datasets, this method gathers similar data points, which allows us to uncover hidden patterns and associations in our data. As an example of that we have the K-means algorithm [10]

Matplotlib [11] is a library designed for making 2D plots of arrays in Python. It was created en 2008 by John D. Hunter because of his need for a packet of Python with certain requirements. In spite of being mainly written in Python, Matplotlib intensively uses NumPy and other extension code in order provide good performance. Its characteristics are based on the features established by its creator:

- Support users of the Scientific Python ecosystem.
- Facilitate interactive data exploration.
- Produce high-quality raster and vector format outputs suitable for publication.
- Provide a simple graphical user interface and support embedding in applications.
- Be understandable and extensible by people familiar with data processing in Python.

- Make common plots easy, and novel or complex visualizations possible.

Some of the plot types Matlab provides are Pairwise data, Statistical distributions and Gridded data.

PyTorch [12] is a comprehensive framework designed to build deep learning models, a category of machine learning widely applied in tasks such as image recognition and language processing. It is noted for its robust GPU support and use of inverse-mode self-differentiation, which enables dynamic modifications to computational graphs. This contributes to its broad acceptance for both rapid experimentation and prototyping.

The core components of Pytorch are tensors and graphs:

- Tensors are similar to a multidimensional array, used to store and manipulate the inputs and outputs of a model, as well as the model's parameters.

- Graphs are structures of information comprising interconnected nodes, known as vertices, and edges. PyTorch relies on dynamic computation graphs, where the computation graph is constructed in real-time using the same code that executes computations during the forward pass, simultaneously creating the necessary data structure for backpropagation.

2.1.3 Natural Language Processing

Natural Language Processing (NLP) [13], is a subfield of Artificial Intelligence and linguistic, dedicated to make computers understand the statements or words written in human languages. Natural languages are those languages that are spoken by people for communicating, it came into existence because it was a way for everyone to be able to communicate with the computer without learning machine specific language.

With the advancement of computing technologies and the increased availability of data, the way natural language is being processed has changed. At first, a traditional rule-based system was used for computations, currently they are being done using Machine Learning and Deep Learning techniques. [14] It was during 1980s when the major work on machine learning-based NLP started. NLP arises from the combination of various disciplines such as artificial intelligence, linguistics, formal languages, and computations.

Contrary to what one might think NLP is not just restricted to text data but also to voice recognition. It can broadly be categorized into two types:

- Natural Language Understanding (NLU), refers to a process by which an inanimate object with computing power is able to comprehend spoken language.
- Natural Language Generation (NLG), refers to a process by which an inanimate object with computing power is able to manifest its thoughts in a language that humans are able to understand.

Because of the presence of various unknown symbols or links, most times text data cannot be used as it is. The process called data cleaning is the one through which unnecessary details are eliminated and only the meaningful portions from data are extracted, this is made for focusing on the actual content and also to reduce computation to archive this different methods are followed.

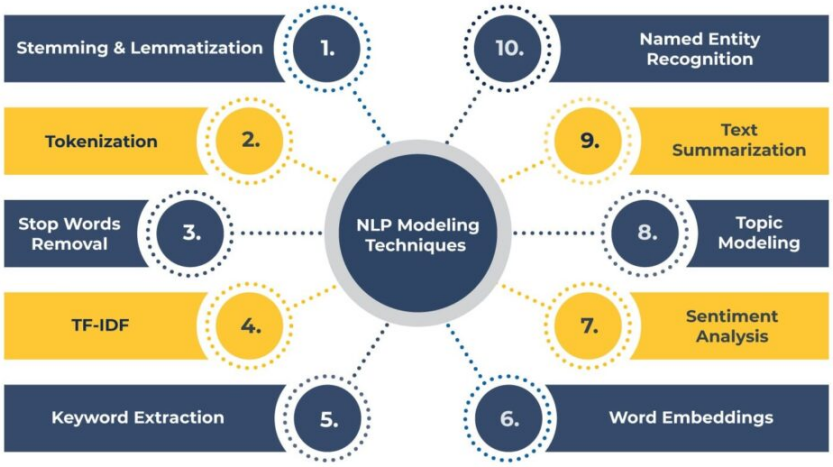

Figure 2.1: Natural Language Processing Techniques

The methods adopted for pre-processing are the ones described below:

- Tokenization refers to the procedure of splitting a sentence into its constituent words.
- PoS Tagging, PoS refers to parts of speech and PoS tagging refers to the process of tagging words within sentences into their respective parts of speech and then finally labeling them.
- Stopwords are common words that are removed from the analysis as they are just used to support the construction of sentences.
- Spelling correction is one of the most important tasks in any NLP project. In spite of being time consuming is key because without it there are high chances of losing out on required information.

- Stemming is a process that convert words into their base forms.

- Lemmatization is used to overcame possible problems with the stemming process. In this process, an additional check is being made, by looking through the dictionary to extract the base form of a word.

2.1.4 Machine Learning

Machine learning (ML) [15] is a field of computer science that studies algorithms and techniques for automating solutions to complex problems that are difficult to program using conventional methods. The goal of an ML algorithm is to learn a model or a set of rules from a labeled data set in order to correctly predict the labels of data points that are not in the data set. ML algorithms do not require an explicit detailed design, but learn the detailed design from a labeled data set. The larger the data set, the more accurate they are.

There are three different styles of learning [16]:

1. Supervised learning involves the use of labeled input data, known as training data, where each data point is associated with a known label or outcome. The model undergoes a training process in which it learns to make predictions and adjusts when its predictions are incorrect. This training iteration continues until the model reaches a certain level of accuracy on the training data.

   Based on the dependent attribute, supervised learning can be categorised into Regression if the attribute is numerical or Classification if it is categorical.[17]

   - A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. The simplest model among many that can be used for this purpose is linear regression which tries to fit data with the best hyper-plane which goes through the points.
   - A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”, attempting to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes.here are a number of classification models. Some of the models used for Classification include decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs-rest, and Naive Bayes.

2. Unsupervised learning works with input data that lacks known labels or outcomes. The model is built by identifying the structures that are inherent in the input data. This process may involve extracting rules of thumb, employing mathematical techniques to systematically reduce redundancy, or organizing the data based on similarity.

   Unsupervised learning can be dived in two [18]:

   - Clustering: The aim is to find homogeneous subgroups within the data, the grouping is based on distance between observations.
   - Dimensionality reduction: The goal is identifying patterns in the features of the data, is often used to facilitate visualisation of the data.

3. Semi-supervised learning works with input data comprising a combination of labeled and unlabeled examples. Although there is a defined prediction problem, the model must simultaneously learn the underlying structures to organize the data and make predictions.

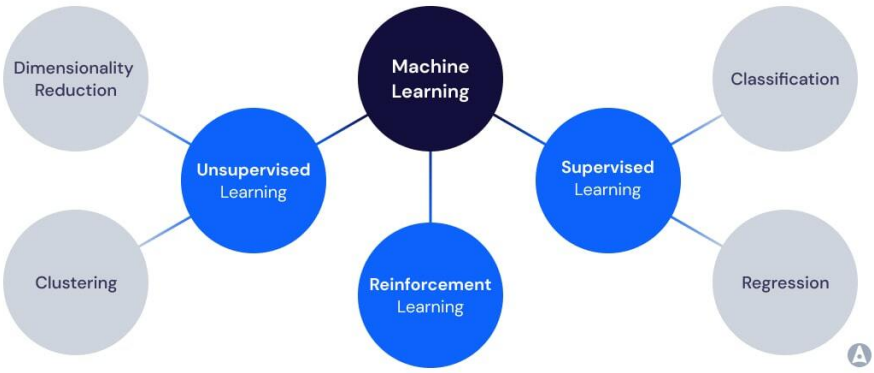

Figure 2.2: Machines Learning styles

2.1.5 Transformers

The Transformer [19] is today an important deep learning model using the TensorFlow and PyTorch frameworks. It has gained wide acceptance in various domains, such as natural language processing (NLP), computer vision (CV) and speech processing. Initially designed as a sequence-to-sequence model for machine translation, later, subsequent studies revealed that Transformer-based pre-trained models (PTMs) offer state-of-the-art performance on a wide range of tasks. Consequently, the Transformer architecture has become the preferred choice in NLP, especially for pretrained models.

Distinguished by its distinctive architectural features [20], the Transformers model is most notable for its departure from conventional reliance on recurrent units. In doing so, it presents a notable advantage in terms of training efficiency, as it requires less time compared to previous recurrent neural architectures, such as short-term memory (LSTM). In the preprocessing phase, the input text is divided into n-grams, which are subsequently encoded as tokens. Each of these tokens undergoes a transformation, becoming a vector through a meticulous process facilitated by a word embedding table. This step is essential to represent the linguistic elements in a continuous vector space. In each layer, each token is contextualized in the range of the context window with other tokens by means of a parallel multihead attention mechanism that allows amplifying the signal of key tokens and decreasing that of less relevant tokens.

There exists two key ideas that have contributed towards the development of conventional Transformer models [21].

- Self-Attention: It enables the capture of "long-term" dependencies among sequence elements, a task that traditional recurrent models struggle with in terms of encoding such connections. When presented with a sequence of items, self-attention assesses the significance of each item in relation to others. This approach explicitly represents interactions between all entities in a sequence, particularly beneficial for structured prediction tasks. To summarize, a self-attention layer enhances each element of a sequence by incorporating comprehensive global information from the entire input sequence.

- (Self-)Supervised Pre-training: Initially, pre-training is performed on a large data set using supervised or self-supervised methods. Subsequently, the weights acquired during pre-training are adjusted for subsequent tasks using smaller or medium-sized data sets. The pre-training phase, especially when based on self-supervision, has proven to be key to improving the scalability and generalization of Transformer models, facilitating the training of models with parameters exceeding one trillion.

Regarding its architecture [22], it follows an encoder-decoder structure but does not rely on recurrence and convolutions for generating an output. Summarizing it very briefly, we could say in a general way that the encoder maps an input sequence to a sequence of continuous representations, which is then fed into a decoder. The decoder, receives the output of the encoder together with the decoder output at the previous time step generating an output sequence.

The encoder consists of a stack of six identical layers, where each of them is composed

of two sublayers:

- The first sublayer implements a multi-head self-attention mechanism. The multi-head mechanism implements heads that receive a linearly projected version of the queries, keys, and values, each to produce outputs in parallel that are then used to generate a final result.

- The second sublayer is a fully connected feed-forward network consisting of two linear transformations with Rectified Linear Unit (ReLU) activation in between: $FFN(x) = ReLU(W_1x+b_1)W_2+b_2$ The same linear transformations to all the words in the input sequence is applied on each of the layers of the encoder, even though each layer employs different weight (W_1, W_2) and bias (b_1, b_2) parameters to do so. On top of that each sublayer is also followed by a normalization layer.

It is crucial to note that the Transformer architecture lacks an inherent ability to capture information about the relative positions of words in a sequence due to its non-recurring nature. To address this limitation, it is necessary to introduce positional encodings in the input embeddings.

These positional encoding vectors share the same dimensionality as the input embeddings and are elaborated using sine and cosine functions with varying frequencies. These vectors are then added directly to the input embeddings, thus incorporating the essential positional information into the model.

The decoder also consists on a stack of six identical layers that are each composed of three sublayers:

- The initial sublayer of the decoder stack takes the previous output of the decoder, enhances it with positional information, and employs multihead self-attenuation. While the encoder is structured to pay attention to all words in the input stream, regardless of their position in the stream, the decoder is specifically designed to focus only on the preceding words. Consequently, the anticipation of a word at a given position depends exclusively on the established outputs of the words preceding it in the sequence. This selective attention mechanism in the decoder is facilitated by the introduction of a mask applied to the values resulting from the scaled multiplication of two matrices in the multihead attention process.

- The second layer implements a multi-head self-attention mechanism similar to the one implemented in the first sublayer of the encoder. Now, this multi-head mechanism

receives the queries from the previous decoder sublayer and the keys and values from the output of the encoder, allowing the decoder to attend to all the words in the input sequence.

- The last layer implements a fully connected feed-forward network, similar to the one implemented in the second sublayer of the encoder.

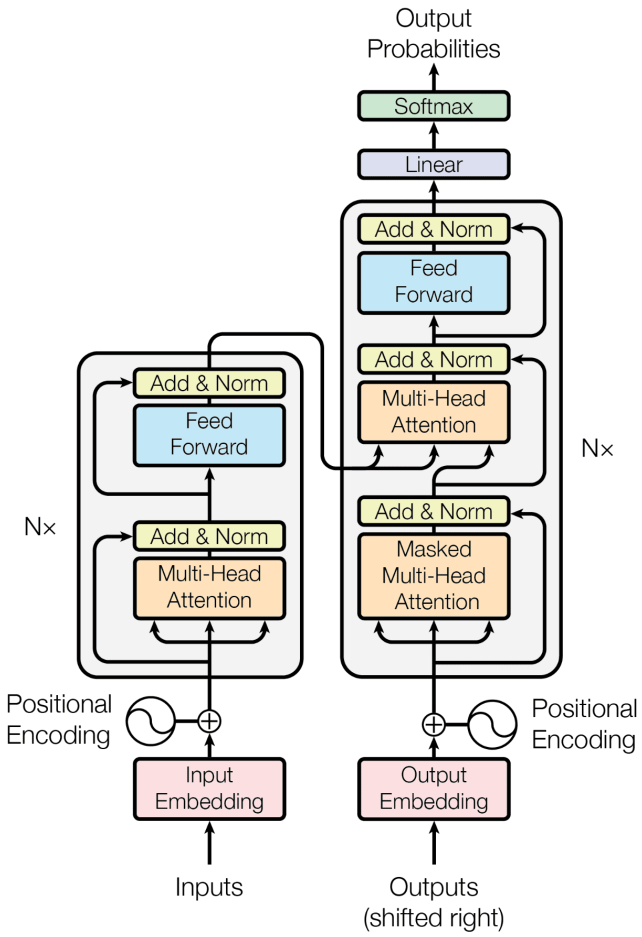

Figure 2.3: Transformers Architecture

As Transformers models have a high complexity, multiple libraries have been created in order to facilitated its implementation. In this subsection we will specially describe the Happy Transformers which is the library that we have attempted to for this project. Happy Transformer is built on top of Hugging Face's transformers library and allows programmers to implement and train Transformer models with just a few lines of code. There are various models we can use such as BERT, DistilBERT, ALBERT and RoBERTa.

Happy Transformer [23] is an open source project with this public repository on GitHub,

it also uses a number of open source projects many of wich have been described before: Transformers, Pytorch, Scikit-learn, Numpy, Pandas, tqdm

The library features seven public methods [24] :

1. Text Generation: Generates a coherent and relevant text from an input or context provided by automated means.
2. Text Classification: Performs text classification by assigning a label to a given text string.
3. Question Answering: The model answers a question given a body, being the outputted answer always a ext-span with the provided information.
4. Word Prediction: Predicting the next word or words of a given sentence.
5. Token Classification: Returns a list with the classified word, the probability of the entity, the predicted entity, the index of the token within the tokenized text, the index of the string where the first letter of the predicted word occurs and the index of the string where the last letter of the predicted word occurs.
6. Next Sentence Prediction: Given two sentences it return the probability of one of the sentences fallowing the other.
7. Text-to-Text: Transforms one type of text into another.

To carry out this project we specially focus on the Text Classification library as we aim to classify different text which contain news and assign each of them a label.

2.1.6 Environments

To carry out this project different environments and platforms have been used and in this section we re going to describe all of them.

The first step this project was finding a suitable dataset for what we used Kaggle [25] which is an online platform, subsidiary of Google. It allows users to find or publish datasets, it also enables working collaboratively on different projects. Apart from that, the platform organizes competitions co-hosted by world-class research organizations and companies to solve data science challenges and offers signed certificate courses toe learn new techniques with no cost. Kaggle's community is a diverse group of 15 million data scientists from students to distinguished researchers present in over 190 countries.

The project was developed using an instance of JupyterLab provided by the Intelligent Group System (GSI), which enables the use of notebooks for coding. JupyterLab represents the latest web-based interactive development environment designed for working with notebooks, code, and data. Its versatile interface enables users to configure and organize workflows in fields like data science, scientific computing, computational journalism, and machine learning. With a modular design, JupyterLab encourages the integration of extensions, providing the flexibility to expand and enrich its functionality.

When we started developing the model using the Transformers, working on a CPU as we had been doing up until this point do was not enough as to implement this sort of models a higher computational capacity was required. In order to solve the problem, the Intelligent Systems Group (GSI) provided us with a GPU where we could run our project successfully as it gave us more computational power .

Another platform that has been very helpful during the development of the project is Hugging Face . This is a site focused towards Artificial Intelligence, users collaborate on Machine Learning models, datasets and applications. It also provides the infrastructure to demo, run and deploy Artificial Intelligence (AI) in live applications. This is the site we have used for finding the Happy Transformer model applied in our data. They have a section dedicated to guides on different tasks "Task Guides" this has been a really powerful resource as we ended up using the "Text Classification Guide" to implement the model with our data.

2.2 Related Work

The field of "fake news" detection has grown tremendously recently and it will continue to do so over the next few years. A lot of this research has been done looking to solve the need of being able to detect the veracity of the news that we are constantly receiving from multiple sources of information.

The various studies use different sources of data, models and NLP techniques in order to find the best combination for creating a system that reports the truthfulness of the pieces of news we come across accurately.

In this subsection we aim to find different approaches of the same problematic by comparing other work previously done on this certain topic.

Ensemble methods [26] use multiple learning algorithms to obtain better predictive performance that any of the constituent learning algorithms alone. In this work by Vinnytsia

National Technical University, the database contained a significant number of metaparameters, therefore that data had to be split in three, training, testing and validation. After prepocesing the data, multiple machine learning techniques were implemented for the classification of short statements, the techniques used were some of the following, logistic regression, naive Bayes classifier, Random Forest classifier, Support Vector Machines and Deep Neural Networks. The results of each of this methods were evaluated separately. Once the algorithms had been implemented separately, a stacking ensemble was used for combining the results. Several combining algorithms were used to make the final classification decision:

a) Simple voting. Each model contributes a single vote to the class to which a statement belongs based on its own classification. These individual votes from each model are then aggregated, the class with the highest total votes is declared as the classification result. In the event of a tie, the classification outcome is randomly chosen from the classes with the highest number of votes.

b) Weighted voting. As in Simple Voting, each model assigns a single vote to the class to which a statement belongs based on its own classification. The model vote is weighted according to its classification accuracy on the validation dataset. The aggregated weighted votes from each model are added, and the class with the highest sum of weighted votes is declared as the classification result. If there is a tie, the classification outcome is randomly chosen from the classes with the highest number of votes, just as in Simple Voting.

For all of the ensembles two different metrics were measured:

a) Classification accuracy based on six available categories

b) Binary classification accuracy. This metric counts the accuracy as if there were only two possible categories for the statement, true or false, when in this work there are actually six that go from “Pants on Fire!” (completely false) to “True”

Analyzing the result the conclusion is that the best results were shown by the stacked ensemble of random forest classifier, support vector machines, deep neural network, naive Bayes classifier.

Another study by K.J. Somaiya College of Engineering of Mumbai, India [26] used another dataset with only true or false labels and three different models to find out which one was the best fit for this problematic. The algorithms used were:

- Naïve Bayes: It makes quick predictions for the machine learning models and works best with text classification. It is used for multi-class and binary classifications, the disadvantage of using it is that it fails to learn the relationship between features as it treats all features independent of each other.
- Support Vector Machine (SVM): The model is constructed after it has already been trained. The main motive of SVM is to categorize new data that comes under.
- Passive Aggressive algorithms: Online learning algorithms used for both regression as well as classification. It is easy to use and work fast but is not as accurate.

After the analysis, the conclusion is that SVM gives the highest accuracy. The approach of this work is more similar to the one we will take, using three different models and evaluating the accuracy of each of them for the same dataset.

The approach used in our work for solving the problematic can be similar in some aspects to those used on the work presented above. Logistic Regression is used both in our work and in the first study presented, although we use it as an individual model and the mentioned work uses it as apart of an ensemble method. Support Vector Machine is also used in our work and booth studies presented, in this case both studies use the method in its own. In the conclusions of both studies, Support Vector Machine is the best performing model.

On one hand our work differs from the first one presented on the type of data selected, our news can only be categorized into two categories where as the first work presented has up to six different ones. On the other hand, and as we have mentioned above, the second work presented does use the same data structure as we do.

CHAPTER 3

Models

3.1 Introduction

In this chapter all the models used during development are collected. We will dedicate a subsection to explain in depth each of them, describing both the theoretical basis on which they are based and how they are implemented.

3.1.1 TF-IDF

Term Frequency - Inverse Document Frequency [27], TF-IDF, is mainly used to get a numeric representation of how important a word is across a set of documents, or news in our case. TF-IDF determines the importance of a term by evaluating its relevance within a particular document and then adjusting it according to its relevance across all news.

The TF-IDF stands out as a widely adopted method for term weighting in NLP applications. It assigns a numerical value to a term based on its importance within a document, adjusted proportionally for its relevance in the entire document corpus. This mathematical process effectively filters common English words, highlighting the terms that bring the most meaning to the text. Tasks such as text summarization, information retrieval and sentiment

classification take advantage of TF-IDF thanks to its powerful weighting mechanism.

In mathematical terms it is described by the following formula: On the equation above

$$TFIDF(T, d, D) = TF(t, d) * IDF(t, D)$$

Figure 3.1: TFIDF function

t=term, d=document, D=set of documents. We can see how the value of TF-IDF is obtained by multiplying the terms TF and IDF calculated separately beforehand. We will now go into a little more detail on each of these two terms.

$TF(t, d)$, measures the term frequency, giving a value describing how many times a certain word appears in a document among the number of times all words appear in that same document. Essentially it quantifies how important is that word to this specific document.

This is shaped by the following equation:

$$TF_ij = n_i, j \sum_k n_i, j$$

Figure 3.2: TF_ij function

$IDF(t, D)$, measures Inverse document frequency, answering the question of, how common is this word among all the documents.

The equation to calculate it is the following:

$$idf(t, D) = \log |D||d \in D : t \in d|$$

Figure 3.3: IDF(t, D) function

The ratio (total documents)/(documents that contain the word) is inverted to give a higher value to words that are less common among all the documents. Otherwise, high IDFs would have an astronomical effect on the TF-IDF value.

3.1.2 N-gram model

Count Vectorizer [28] is a scikit-learn package that uses count vectorization to convert a collection of text documents to a matrix of token counts. Because machines cannot understand either characters or words, when dealing with text data we need to represent it in numbers to be understood. This technique enables using text data easily and directly in Machine Learning and Deep Learning models like text classification. It works tokenizing the text while performing basic preprocessing techniques.

It works by tokenizing the data and dividing it into units known as n-grams, with the option of specifying the length using a tuple in the ngram_range parameter. For example, setting it to 1.1 provides unigrams or 1-grams such as "whey" and "protein", while setting it to 2.2 provides bigrams or 2-grams, such as "whey protein". To implement CountVectorizer, we initialize the class, providing the necessary arguments, and then we call fit_transform(). This process involves executing the fit() function on the data, followed by the transform() function. These steps establish a vocabulary of n-grams derived from the documents and encode them into a vector. Afterwards, we use toarray() to convert the vector into an array and transfer the data to Pandas for further analysis.

Another parameter included is max_features, which enables the regulation of the vocabulary size. This parameter facilitates the inclusion of only the most frequently encountered terms, determined by their term frequency across documents within the corpus.

3.2 Traditional Classifiers

Traditional Classifiers [29] are based on Supervised Learning and the and the connotation of classifier is due to the fact that the attribute that labels it is categorical in type. Classification involves identifying, understanding and grouping concepts and items into categories. Machine learning applications leverage pre-established training datasets and employ multiple algorithms to categorize the next datasets based on learned patterns. Algorithms use input training data to predict the probability that subsequent data will fall into one of the predetermined categories.

The most popular algorithms used for classification are: Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Tree and Support Vector Machines.

Using these models we can help solve problems such as Image classification, Fraud detection, Document classification, Spam filtering, Facial recognition, Voice recognition, Medical diagnostic test, Customer behavior prediction, Product categorization or Malware classification.

Particularly in our case, where we want to a assign a label true/false to each statement with the highest possible accuracy we have selected some of these models among the many that exist because these are the ones that we think will archive higher performance, after applying all of the them will be able to reach a conclusion about which one is more suitable for our data and our purpose.

3.2.1 Logistic Regression

Logistic Regression [30] is a classification algorithm in Machine Learning that predicts the probability of specific classes by analyzing dependent variables. In simpler terms, the algorithm calculates a sum of the input features and evaluates the logistic function of the outcome. In contrast to linear regression it accepts both continuous and discrete variables as input and its output is qualitative, additionally it predicts a discrete class. The algorithm analyses relationships between variables helping predict the probability of an event happening or not being either 0 or 1.

Using the Sigmoid function the algorithm assigns probabilities between 0 and 1 to a discrete outcome which results on 0 or 1. When working with binary predictions you can section the population in two by a cut-off on 0.5, whatever is above it belongs to group A and the remaining part is group B. A hyperplane is used as a decision line to separate both groups once data points have been assigned to a class using the function.

The Sigmoid function is the one shown underneath:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Figure 3.4: Sigmoid Function

As we can appreciate in the formula X is the parameter we want to learn, train or optimize. The Sigmoid function forms an S shaped graph, which means as x approaches infinity, the probability becomes 1, and as x approaches negative infinity, the probability becomes 0. The model sets a threshold that decides what range of probability is mapped to which binary variable.

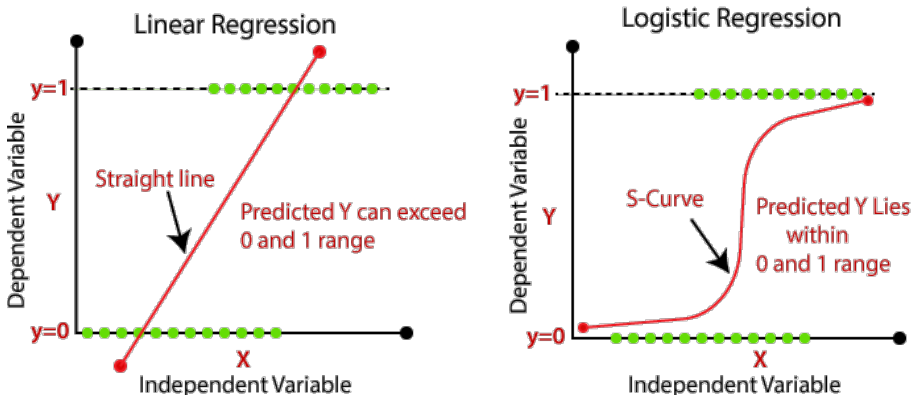

Figure 3.5: Linear Regression vs Logistic Regression

In the image above we can visually note the difference between both liner and logistic regression [31]. The equation by which the variables are related to each other used for linear regression would be $y = a0 + a1x2 + a2x2 + ... + aixi$ whereas for logical regression is $y(x) = \frac{e(a0+a1x1+a2x2+\ldots+aixi)}{(1+e(a0+a1x1+a2x2+\ldots+aixi)}$. The probability equation also differs from one type of regression to the other being the one associated to the linear type $p = a0 + a1x1 + a2x2 + \ldots + aixi$ and the one associated to regression $\ln(\frac{p}{1-p}) = b0 + b1x1 + b2x2 + \ldots + bkxk$, where p refers to the probability.

For training and testing our data using this model, the first step is to import the model, Logistic Regression, from the Sklearn, Linear Model library. The model has multiple parameters we can adjust in order to find the best fit for our data, the only parameter that has been adjusted is the Maximum number of iterations taken for the solvers to converge which by default is 100 and we have set a value of 5000.

3.2.2 Linear Support Vector Machine

Support Vector Machine (SVM) [32] is a supervised learning algorithm in Machine Learning designed to address challenging classification, regression and outlier detection challenges. To achieve this goal, SVM executes optimal data transformations that establish boundaries between data points according to predefined classes, labels, or outputs. These algorithms are highly beneficial in a variety of fields, such as healthcare, natural language processing, signal processing applications, and speech and image recognition.

Now that we have introduced the model and have a better overview of what it is and how it works, we are at a good point to start digging into the technical details.

In technical terms, SVMs are mainly intended for binary classification problems, with the objective of identifying a hyperplane that effectively separates data points of different classes. The location of the hyperplane is optimized for maximum margin, ensuring a clear boundary between the classes.

On the image shown above we can see the hyperplane previously described. The margin denotes the largest width of the range parallel to the hyperplane without holding any internal support vectors. Although it is simpler to define these hyperplanes for linearly separable problems, in real life, the SVM algorithm attempts to maximize the margin between support vectors. This search can lead to misclassifications on smaller segments of data points.

In the particular case of our dataset there are two labels, true/false, that categorise the news. We aim to have a classifier for these tags that classifies data into either one or the

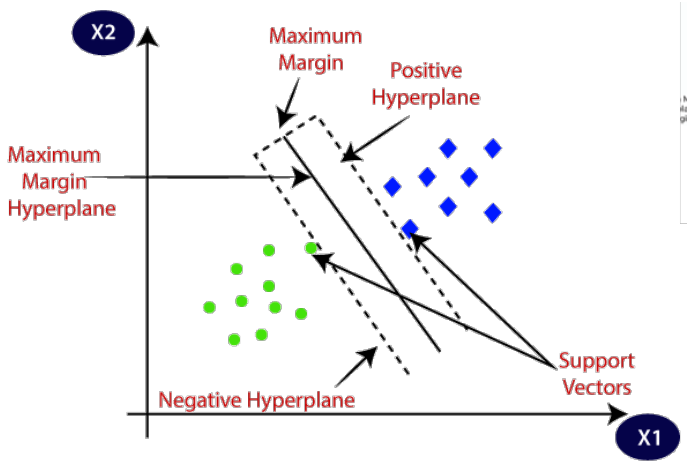

Figure 3.6: SVM Hyperplane

other. Normally, SVM splits the data points according to the true or false labels using, in this case, a two -dimensional line hyperplane. The decision boundary line is denoted by the hyperplane, in which the data points belong to the true or false category. The hyperplane, represented as a line, tends to widen the margins between the closest labels. The distance between the most inmediate label is the largest, making it easier to classify the data.

In the case of non-linear data, a simple straight line is not suitable for separating the distinct data points. In order to classify the data on this scenario another dimension is needed to feature space, unlike in linear data were two dimensions are enough.

When applying the model to our data we used the SVM model imported from the Sklearn.svm library. The model allows us to modify multiple parameters and attributes adjusting them to our needs according to the data used. In our case we used the parameter 'C' which corresponds to regularization, influencing the balance between having a soft decision boundary and correctly classifying the training points. The simpler the value of the parameters, the simpler the decision surface, with higher values the decision boundary is more complex. As well as that, we also modified the parameter 'kernel' assigning it the value 'linear', specifying the hyperplane used for classification. If the parameter 'kernel' is not specified by default, a non-linear type will be used.

3.3 Transformers

After a deeper dive into Machine Learning models, we will now focus on transformers which rely on deep learning, a subset of machine learning.

In the previous chapter we explained Transformers in depth, in this section we will focus in particular on the model we have used, Bert and specifically on the one used from the Hugging's Face library, bert-base-uncased.

3.3.1 BERT

BERT [33]is a recent Transformers model of language representation that has obtained surprisingly good results in several comparative tests of language comprehension. This result indicates the possibility that BERT networks capture structural information about language. This model BERT (Bidirectional Encoder Representations from Transformers) is a bidirectional adaptation of Transformer networks. It is trained to simultaneously predict a masked word within its context and classify whether two sentences are consecutive. The resulting model can be fine-tuned for various downstream Natural Language Processing (NLP) tasks, such as question answering and linguistic inference, with minimal adjustments. It has demonstrated superior performance to state-of-the-art models on all eleven NLP tasks evaluated in the GLUE benchmark, outperforming them by a notable margin.

The Transformer architecture has two distinct mechanisms [34], an encoder to process text input and a decoder to generate task predictions. Since the goal of BERT is to create a linguistic model, only the encoder mechanism is needed. Contrary to sequential models that process text input in a linear manner (left-to-right or right-to-left), the Transformer encoder understands the entire sequence of words simultaneously. It is therefore called bidirectional, although a more accurate term would be non-directional. This feature allows the model to capture the context of a word by taking into account its entire context, both to the left and to the right of the word.

When training language models, there is a challenge of defining a prediction goal. To overcome this challenge, BERT uses two training strategies:

- Masked LM (MLM): Before introducing word sequences into the BERT, 15% of the words in each sequence are replaced by a [MASKED] token. The model then attempts to predict the original values of the masked words, taking advantage of contextual clues provided by the other unmasked words in the sequence. In technical terms, the prediction process involves:
  - Feeding a classification layer on top of the encoder output.
  - Transforming the output vectors into the vocabulary dimension by multiplying them by the embedding matrix.

- Employing softmax to compute the probability of each vocabulary word.

The BERT loss function focuses specifically on the prediction of the masked values, disregarding the predictions of the unmasked words. Consequently, the model exhibits a slower rate of approach compared to directional models. However, this feature is outweighed by its greater knowledge of the context.

- Next Sentence Prediction (NSP): During the BERT training process, the model receives pairs of sentences and learns to predict whether the second sentence in the pair follows the first sentence in the original document. Throughout the training, 50% of the inputs consist of pairs in which the second sentence is the next sentence in the original document. The other 50% include a randomly selected sentence from the corpus, which is assumed to have no direct relationship to the first sentence, as the second sentence.

  To help the model distinguish between the two sentences during training, the input is undergone the following preprocessing steps before being introduced into the model:

  - A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
  - Each token is assigned a phrase embedding that indicates whether it belongs to phrase A or to phrase B. Phrase embeddings work similarly to token embeddings, but have a vocabulary of 2. A positional embedding is added to phrase B.
  - A positional embedding is added to each token to indicate its position in the sequence, a concept introduced and detailed in the Transformer article. To predict the connection between the first and second sentences, the following steps are executed:
    * The entire input sequence is processed through the Transformer model.
    * The output of the [CLS] token undergoes transformation into a 2×1 shaped vector using a straightforward classification layer, involving learned matrices of weights and biases. probability of IsNextSequence is calculated using softmax.

  During BERT model training, Masked LM and Next Sentence Prediction are concurrently trained with the objective of minimizing the combined loss function derived from both strategies.

There are two main approaches [35] for using BERT: feature extraction and finetuning.

- In feature extraction, the architecture of the BERT model is preserved, as if the model's parameters were 'frozen'. Features are extracted from the pretrained BERT model and then fed into a classifier model to solve a given task.
- In finetuning, the model's parameters are finetuned by adding extra layers to the original BERT architecture. The new layers are used to train the model on the downstream tasks.

3.3.2 bert-base-uncased

The Hugging Face Hub offers many models for a variety of specific Machine Learning tasks, those models, which have been trained and adapted for an specific matter are stored in repositories. They benefit from all the features of every repository on the platform, in addition model repos have attributes that make exploring and using models as easy as possible.

After searching among the different models that perform the task which we aim to archive in this project we came a cross bert-base-uncased, this model has been our model of choice to train and test the data with a Transformers model as our last approach of this development.

The model [36] has been pretrained on English language using a masked language modeling (MLM) objective, as it is uncased it does not make a difference between english and English. The model is primarily aimed at being fine-tuned on tasks that use the whole sentence to make decisions, such as sequence classification, token classification or question answering.

The texts are lowercased and tokenized using WordPiece and a vocabulary size of 30,000. The details of the masking procedure for each sentence are the following:

- 15% of the tokens are masked.
- In 80% of the cases, the masked tokens are replaced by [MASK].
- In 10% of the cases, the masked tokens are replaced by a random token (different) from the one they replace.
- In the 10% remaining cases, the masked tokens are left as is.

The model was trained on 4 cloud TPUs in Pod configuration (16 TPU chips total) for one million steps with a batch size of 256. The sequence length was limited to 128 tokens

for 90% of the steps and 512 for the remaining 10%. The optimizer used is Adam with a learning rate of 1e-4, $_1 = 0.9$ and $_2 = 0.999$, a weight decay of 0.01, learning rate warmup for 10,000 steps and linear decay of the learning rate after.

When fine-tuned on downstream tasks, this model achieves the following results:

MNLI-(m/mm)	84.6/83.4
QQP	71.2
QNLI	90.5
SST-2	93.5
CoLA	52.1
STS-B	85.8
MRPC	88.9
RTE	66.4
Average	79.6

Table 3.1: bert-base-uncased results

CHAPTER 4

Evaluation

4.1 Introduction

In this chapter we are going to evaluate the results obtained when applying the different models, which were already explained on the previous chapter. These models were evaluated on the two different datasets gathered for this project. On the sections below the two mentioned datasets will be explained and analysed in depth, then the metrics applied in the models will be collected and after the design decisions will be detailed. To close the chapter, the results will be presented.

4.2 Materials

To carry out this project two different datasets have been used to evaluate the various models.

The first dataset used, called "Fake News Detection Datasets", was obtained from the platform Kaggle [37]. The folder contained two files "False.csv" and "True.csv" which we merged into just one dataframe called "news", it has six columns: index, title, text, subject,

date, label. The dataframe contains over 40,000 news which are labeled either true (1) or false (0), this column was added when both files were merged. The column "text" contains the news statement that have been previously labeled, in this case the text is not pre-processed so in order to work with this data the first step will be pre-processing to remove all the none essential information in the text.

In this dataframe, the total for the news labeled as true is 21417 where as for the ones named as false is 23481. Therefore 47.60 % would be true and the remaining 52.40 % false, which shows the amount of each labels is highly balanced.

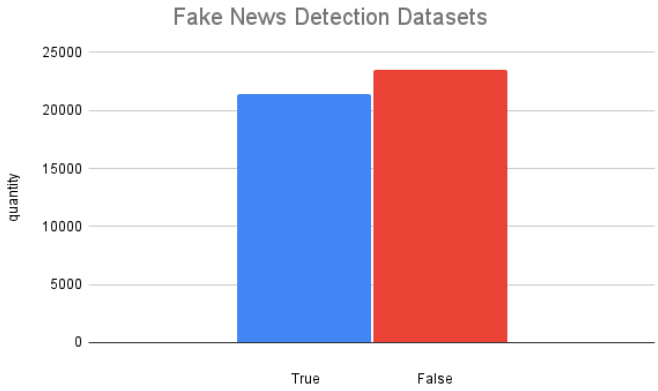

Figure 4.1: "Fake News Detection" dataset

The second dataset [38], called "Fake News", was obtained from the site Kaggle as well. In this case the folder contained also two files, but instead of being separated by the label they were already separated into train and test, not being necessary to merge both files.

This dataframe just contained one column at the beginning with the format (text;label), to process the data that unique column was separated into two different ones, text and label with the particularity that in this case the text had already been pre-processed. As in the previous dataset "label" had two possible values true (1) or false (0) and "text" contained news statements.

The file "test.cvs" contains a total of 4035 news statements, out of all of those 2056 are true (1) and 1979 false (0), therefore 50.94 % are true, and 49.06 % are false, once again the amount of each is highly balanced. Analyzing the "train.csv" file, it has a total of 16646 news statements although in this case the amount of true/false statements is exactly the same 8323 each.

Taking both files and analysing them as a whole, there are a total of 20681 statements, being 80.41 % destined for training and 19.59 % for testing.

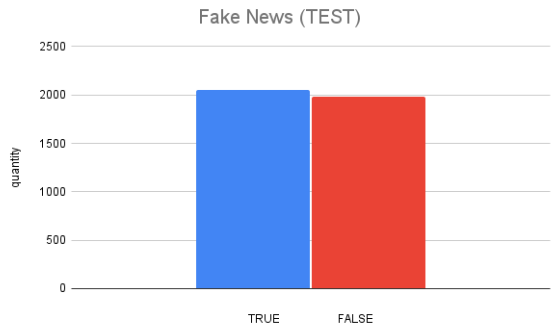

Figure 4.2: True and False statements in Test

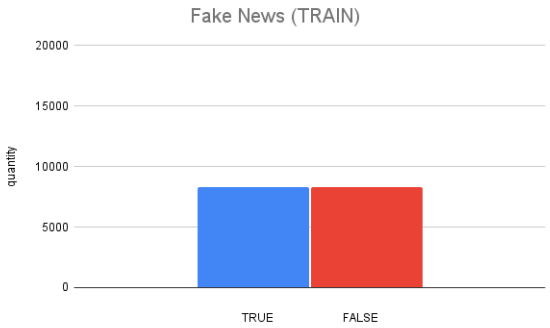

Figure 4.3: True and False statements in Train

4.3 Design

Once the datasets with which we will be working have been explained, we now have enough context to deepen into the details of the design.

When the data is loaded and organized, meaning, all the transformations needed like creating columns or joining different files and any others already described on the section above have been done, we are ready to pre-process our data. This is fundamental to maximize performance of the models, the process was only done on the first dataset used, "Fake News Detection Datasets" because the other dataset had already pre-processed data.

The first step when pre-processing is to split the text into sentences, after this our text has been broken down in multiple units, facilitating the analysis. Following that, we then break those sentences into words making it even simpler for machines to understand, this whole process of breaking down the text is called tokenization which aims to transform text into tokens. With the text divided into words the next step is to lemmatize those words, reducing them to their root or base, called the lemma, in order to extract that part of the word which contains the meaning. As this process is about taking steps to obtain the meaningful information on the text, the next phase is removing "stopwords", those words are typically used in language but they do not provide any meaning. In relation with removing with those "stopwords" our data also goes through a process of removing punctuation which is also frequent in text but do not provide any meaning. After this process our initial text has been reduced to the key information.

In order to make the data suitable for Machine Learning algorithms it is crucial that we vectorize it, this involves converting textual information into numerical vectors. Once it is vectorized, in the case of the first dataset we need to split the dataframe into train and

test, this can be done by using the "train_test_split"[1] function from the scikit-learn library, the default values are 25 % of the data is used for testing, and 75 % is used for training.

Afterwards, with the training and testing sets and the vectorized data we first train the model and then evaluate it, this is done individually for each of the models previously explained.

4.4 Metrics

Prior to discussing the results obtained from the model it is necessary to detail the metrics used to evaluate the models. Defining these metrics will help us to compare the results obtained with each model and to see which one best suits our needs.

To evaluate our models we have used the ´´classification report´´from the skelearn library. This reports includes various metrics, below we present the definition to each of them as well as their equations .

On the presented equations [39] the acronyms have the following meaning: TP = True Positive, PP = Predicted Positive, FDR = False discovery rate, P = Positive, FNR = False negative rate, PPV = Positive predictive value, TPR = True Positive Rate, FP = False Positive, FN = False Negative.

- Precision: It is calculated as the ratio of true positives to the sum of true positives and false positives. It measures the accuracy of the positive predictions made by the model.
  $= \frac{TP}{PP} = 1 - FDR$
- Recall: Measures the ability of the model to capture the key information of the positive class, it is figured as the ratio of true positives to the sum of true positives and false negatives.
  $= \frac{TP}{P} = 1 - FNR$
- F1-Score: Provides a balance between precision and recall, expressed the harmonic mean of precision and recall, being especially useful in the presence of an uneven class distribution.

[1] Scikit-Learn Documentation, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split

$$= \frac{2PPVxTPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

- Support: Helps to understand the distribution classes in the dataset by counting the occurrences of the class in the dataset.

- Accuracy: The ratio of correctly predicted instances to the total instances. While accuracy is a common metric, it may not be suitable for imbalanced datasets.

  $$= \frac{TP + TN}{P + N} = 1 - FNR$$

- Macro Avg and Weighted Avg: Macro average calculates the average of the unweighted per-class metrics, while weighted average calculates the average of the metrics with weights proportional to the support of each class.

4.5 Results

In this section we will present the obtained results with the metrics explained above.

First of all, for the "Fake News Detection Datasets", we started by evaluating the data with the Logistic Regression model, obtaining the results presented on the chart below.

	Precision	Recall	F1-Score	Support
fake	0.99	1.00	1.00	5873
not fake	1.00	0.99	0.99	5352
accuracy			1.00	11225
macro avg	1.00	0.99	0.99	11225
weighted avg	1.00	1.00	1.00	11225

Figure 4.4: Results when applying Logistic Regression to "Fake News Detection Datasets"

After obtaining such good results we decided to also evaluate Logistic Regression with the "Fake News" dataset and as we can contrast with the results presented above, the results were more realistic due to using a more complex dataframe.

	Precision	Recall	F1-Score	Support
fake	0.95	0.94	0.95	1995
not fake	0.94	0.95	0.95	2040
accuracy			0.95	4035
macro avg	0.94	0.95	0.95	4035
weighted avg	0.94	0.95	0.95	4035

Figure 4.5: Results when applying Logistic Regression to "Fake News"

The next model implemented on our "Fake News" dataset was Support Vector Machine, it was evaluated with two different NLP techniques, first Count Vectorizer was applied and then TF-IDF obtaining different results as we can see in the charts below.

The results obtained when using both Count Vectorizer and SVM are the ones presented on the following table:

	Precision	Recall	F1-Score	Support
fake	0.94	0.93	0.93	2007
not fake	0.93	0.94	0.93	2028
accuracy			0.93	4035
macro avg	0.93	0.93	0.93	4035
weighted avg	0.93	0.93	0.93	4035

Figure 4.6: Results when applying SVM with Count Vectorizer

After combining SVM with Count Vectorizer, we then switched to the other mentioned technique, TF IDF, and the results obtained were the following:

	Precision	Recall	F1-Score	Support
fake	0.97	0.97	0.97	1984
not fake	0.97	0.97	0.97	2051
accuracy			0.97	4035
macro avg	0.97	0.97	0.97	4035
weighted avg	0.97	0.97	0.97	4035

Figure 4.7: Results when applying SVM with TF-IDF

The last model evaluated on the "Fake News" dataset was the Transformers model BERT.

	Precision	Recall	F1-Score	Support
fake	0.99	0.99	0.99	1995
not fake	0.99	0.99	0.99	2040
accuracy			0.99	4035
macro avg	0.99	0.99	0.99	4035
weighted avg	0.99	0.99	0.99	4035

Figure 4.8: Results when applying BERT

Once we have analyzed the results of each model separately is useful to get an overview of all the models comparing the accuracy of their F1-Score as it is the most useful metric to compare the models and evaluate their performance.

The "Fake News Detection Datasets" data set was only used to train one model. Meanwhile "Fake News " data set was evaluated with three different models one of each was assesed with two NLP techniques.

“Fake News Detection Datasets”	
Logistic Regression	1.oo
“Fake News ”	
Logistic Regression	0.95
SVM + TF-IDF	0.97
SVM + Count Vectorizer	0.93
Transformers (BERT)	0.99

Figure 4.9: Global F1 score accuracy comparison

CHAPTER 5

Conclusions

5.1 Introduction

Once we have gather all the results and know how to interpret the metrics given, we can discuss those results in order to come to a conclusion on which one is most appropriate. In this chapter we will describe the final conclusion as well as reviewing the objectives set at the beginning and discussing future work.

5.2 Conclusions

In this project we aim to develop a system to detect "fake news" using Machine Learning algorithms and Natural Language Processing helping us predict if a given news statement is true or false.

"Fake news" have a great impact in our society in all aspects, including crucial areas such as politics or health, and both their popularity and frequency have increased over the last few years as result of an increased use of social media.

The system developed in this project is a progress towards achieving the detection of

"fake news" on the different sources of information with a higher degree of accuracy. By using two datasets, Natural Language Processing Techniques and three Machine Learning models we have developed a system which has the previously presented results.

When we first started developing we used the "Fake News Detection Datasets" but because of its simplicity and after using it to apply the Logistic Regression model, we soon realized we had to switch to a more complex dataset in order to continue developing our system with a more realistic scenario.

Once we had switch to the "Fake News" dataset we started by applying the Logistic Regression model which is the least time-consuming out of all of the models applied. If we compare the results obtained with this same model applied on the first dataset we can see how this results are more realistic, due to a more complex data.

The next model used was Support Vector Machine (SVM), it was implemented with two different Natural Language Processing techniques, Count Vectorizer and TF-IDF. Contrasting the results obtained for each technique, it states that the model performs better when using TF-IDF. This model is however more computationally demanding when comparing it to the first one applied.

The last model used was the Transformers model BERT, this is by far the most demanding in terms of both time and computational resources. Because of it computational complexity a GPU was required in order to process the data given to the model. If we analyze the results obtained we can appreciate that they are almost perfect. Although, as this is the most powerful model we have used during this development, it should perform best compared to the rest, such outstanding results mean that the second dataset is also not complex enough. This dataset has been good to compare the different models but for future research it would be convenient to test with even more complex data.

In conclusion, the Transformers model BERT does in deed perform best but with really high costs of time and computational resources while in this case other models such as Support Vector Machine combined with the appropriate NLP techniques can perform almost as well with much lower costs of both time and computational resources.

Evaluating the objectives set at the beginning, we can check how we have been meeting each one of them:

1. Multiple datasets were gathered, as it had been previously stated on the evaluation chapter, we managed to find two different dataset with varying complexities and each of them was used to train different models looking for the most appropriate one in order to accomplish the desired objective.

2. By training and testing different classification models we have been able to compare how well they perform. A part from using various models with different complexity, we have also combined the models with some of the multiple NLP techniques aiming to find the most accurate combination.

3. After gathering the results obtained when evaluating the models we have been able to come to a conclusion on which is the most suitable one for our needs. Because of always using the same metrics to evaluate each of the models this has allowed us to compare how well each model performed in comparison to the rest.

5.3 Future Work

In the future, this work could be continued in different aspects in order to obtain better and more accurate results. Different tasks could be done to archive this goal.

A good starting point to continue developing this systems would be to use other Machine Learning models and see if there are any of them which will provide better results. During the development of this system we have implemented three different algorithms but there are many more that could be suitable for this same purpose.

One of those mentioned task that could be carried out would be using ensemble learning, which is based on training different models separately to then combine them in one, creating a more optimal model. This was previously seen on the chapter were related work was discussed and can be a powerful resource to use once we have evaluated various models separately.

All of the data used had already been pre-labeled, in the future it would be interesting trying to train models with unlabeled data. This could be a great improvement towards helping users detect the veracity of the news they come across. Besides a new data that it not pre-label it would be important to use more complex data, this will help obtain more accurate and realistic results, training the models with high complexity data gives a better understanding of how well each model performs for the desired purpose.

In addition to all of the possible changes and improvements mentioned above, it would be really interesting to create a platform to facilitate users detect if a piece of information is true or not.

APPENDIX A

Impact of this project

This appendix reflects, quantitatively or qualitatively, on the possible impact.

A.1 Social impact

The development of "Fake News" detection system has a tremendous impact on our actual society. "Fake News" influence the public opinion when it comes to decision making, this can have severe consequences when it regards topics such as health or politics.

By analysing the language and patterns on the pieces of news this project aims to offer the public truthful information avoiding both disinformation and misinformation, helping society to be well informed and to be able to make decisions based on reliable information.

On the other hand, if the models do not accurately predict the veracity of the news, the results can be fatal. As we have exposed above, the news we consume daily have a great impact on the decisions we make, these affect all areas of our lives, from political, medical or education issues to which is the safest place to spend our vacation. An erroneous prediction of the news would cause what is known as misinformation, incorrect or misleading information. Although it is unintentional since the misinformation is due to the malfunctioning of

the model, we are still giving incorrect information to the public.

A.2 Environmental impact

Training such large datasets with these complex algorithms require high computational power provided by a Graphic Processing Unit (GPU). Unfortunately GPUs lead to energy consumption causing a significant environmental impact. A part from the energy they consume, the carbon emissions associated with GPU operations contribute to climate change and if the data centers rely on coal or other non-renewable energy sources it may have an even higher environmental impact.

On the bright side by providing an environment with verified information there is no need for user to contrast information on multiple sites, reducing the demand of energy use in data transmission, processing and storage resulting in the reduce of energy consumption.

A.3 Economic impact

Spreading "fake news" about business or economic indicators can have real a financial impacting economies by influencing stock markets, consumer behavior, and investor confidence.

The development of models to combat "fake news" may stimulate the growth of a market for anti-disinformation technologies. With the increasing demand of this services, the number of professionals and the jobs positions offered grow as well. In order to provide this service companies need different engineers to perform the involved tasks such as developing and maintenance. A growth on the demand is associated to higher profits for companies providing this services.

APPENDIX B

Economic budget

This appendix details an adequate budget to bring about the project.

B.1 Physical resources

This project was develop on a 2017 MacBook Air with a 1,8 GHz Intel Core i5 dual core processor, 8 GB x 1600MHz DDR3 RAM, 512GB of local storage and Intel HD Graphics 6000 1536 MB. The mentioned computer has a value of 711 euros.

As we have previously mentioned a GPU was needed, the one provided by the Group of Intelligent System (GSI) was an NVIDIA GeForce RTX 2080 Ti which costs around 600 euros. The use of a GPU is associated with a high consumption of energy of approximately 250W, taking into consideration we have spend about 250 hours using this GPU and being 0.17 euros the average price for the kW/h the total cost of using the GPU is 611 euros

There were no software license needed to buy, making the cost 0 euros for software.

B.2 Human resources

As this was an individual project, only one salary was needed.

The total of hours dedicated to the development and implementation of the project adds up to a total of 400 hours of work. An average salary for a trainee is around 600 euros when working part-time, as a consequence for each hour the trainee charges 6 euros meaning this project had a total cost of 2400 euros regarding human resources.

B.3 Conclusions

If we add up all the cost previously mentioned, we obtain as a result that the total cost for this project has been 3722 euros.

Bibliography

[1] A brief history of fake news. https://www.bbc.co.uk/bitesize/articles/zwcgn9q.

[2] Carlow Today Tomorrow. How digital media has changed communications — Carlow Today Tomorrow. https://blog.carlow.edu/2021/12/16/how-digital-media-has-changed-communication/, December 2021.

[3] Northeastern University. Research Subject Guides: Fake News/Misinformation/Disinformation: What is fake news? https://subjectguides.lib.neu.edu/fakenews, December 2023.

[4] Python. www.python.org/doc/essays/blurb/.

[5] Pandas - Python Data Analysis Library. https://pandas.pydata.org/about/.

[6] Alexandre. Pandas : the Python library dedicated to data science. https://datascientest.com/en/pandas-the-python-library, October 2023.

[7] Moez Ali. NLTK Sentiment Analysis Tutorial for Beginners. https://www.datacamp.com/tutorial/text-analytics-beginners-nltk, March 2023.

[8] Gsi-Upm. Intro ScikitLearn. https://github.com/gsi-upm/sitc/, 2022.

[9] Scikit-Learn Cheatsheet: Methods for Classification and regression. https://www.turing.com/kb/scikit-learn-cheatsheet-methods-for-classification-and-regression, May 2022.

[10] Nate Rosidi. Clustering with SciKit-Learn: A Tutorial on Unsupervised Learning - KDNuggets. https://www.kdnuggets.com/2023/05/clustering-scikitlearn-tutorial-unsupervised-learning.html: :text=According

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[12] What is PyTorch? https://www.nvidia.com/en-us/glossary/data-science/pytorch/: :text=PyTorch

[13] Abhimanyu Chopra, Abhinav Prashar, and Chandresh Sain. Natural language processing. *International journal of technology enhancements and emerging engineering research*, 1(4):131–134, 2013.

[14] Sohom Ghosh. Natural Language Processing Fundamentals. page 374, March 2019.

[15] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *Machine Learning Definition and Basics*, pages 1–17. Springer International Publishing, Cham, 2019.

[16] Jason Brownlee. A tour of machine learning algorithms. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/, October 2023.

[17] GeeksforGeeks. Regression and classification supervised machine learning. https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/, January 2023.

[18] Laurent Gatto. Chapter 4 Unsupervised Learning — An Introduction to Machine Learning with R. https://lgatto.github.io/IntroMachineLearningWithR/unsupervised-learning.htmlintroduction, February 2020.

[19] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022.

[20] Wikipedia contributors. Transformer (machine learning model). https://en.wikipedia.org/wiki/Transformer$_(machine_learning_model), January$2024.

[21] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Comput. Surv.*, 54(10s), September 2022.

[22] Stefania Cristina. The transformer model. https://machinelearningmastery.com/the-transformer-model/, January 2023.

[23] HappyTransformer. https://pypi.org/project/happytransformer/1.0.2/: :text=Happy

[24] Happy Transformer. https://happytransformer.com/.

[25] Kaggle: your machine learning and data science community. https://www.kaggle.com/.

[26] Mikhail Granik and Vladimir Mesyura. Fake statements detection with ensemble of machine learning algorithms. *Problems of Information Technology*, 09:48–52, July 2018.

[27] Darla M. How TF-IDF works. https://towardsdatascience.com/how-tf-idf-works-3dbf35e568f0, March 2022.

[28] Matt Clarke. How to use CountVectorizer for N-Gram Analysis. https://practicaldatascience.co.uk/machine-learning/how-to-use-count-vectorization-for-n-gram-analysis, December 2021.

[29] Rachel Wolff. 5 Types of classification algorithms in Machine learning. https://monkeylearn.com/blog/classification-algorithms/, August 2020.

[30] Sonia Jessica. How does logistic regression work? https://www.kdnuggets.com/2022/07/logistic-regression-work.html, July 2022.

[31] Vijay Kanade. Linear vs. logistic regression. https://www.spiceworks.com/tech/artificial-intelligence/articles/linear-regression-vs-logistic-regression/, June 2022.

[32] Vijay Kanade. All you need to know about support vector Machines. https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/: :text=A

[33] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language?, 2019.

[34] Rani Horev. BERT explained: State of the art language model for NLP. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270, November 2018.

[35] Ali Saleh Alammary. Bert models for arabic text classification: a systematic review. *Applied Sciences*, 12(11):5720, 2022.

[36] Bert-base-uncased · Hugging face. https://huggingface.co/bert-base-uncased, June 2023.

[37] EMINE BOZKUŞ. Fake news Detection datasets. https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets, December 2022.

[38] William Lifferth. Fake news. https://kaggle.com/competitions/fake-news, 2018.

[39] Wikipedia contributors. F-Score. https://en.wikipedia.org/wiki/F-scoreDiagnostic$_t esting, December$2023.

[40] Jasmine Shaikh and Rupali Patil. Fake news detection using machine learning. In *2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)*, pages 1–5, 2020.