People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

UNIVERSITY OF ABDELHAMID MEHRI – CONSTANTINE 2

Faculty of New Technologies of Information and Communication (NTIC)

Department of Fundamental Computing and its Applications (IFA)

MASTER'S THESIS

to obtain the diploma of Master degree in Computer Science

Option: Networks and Distributed Systems

BERT-based Sentiment Analysis for COVID-19 Tweets

Realized by:

BOUCHELAGHEM Ayyoub

ZOUAGHI Anis

Under supervision of:

Dr. ZERABI Soumeya

July 2023

Acknowledgments

"We thank ALLAH for giving us the health and the will to start and finish this master project.

First of all, this work would not be as rich and could not have been born without the help and guidance of Dr. ZERABI Soumeya, we thank her for the quality of her exceptional supervision, for her patience, her rigor and her availability during the preparation of this master project.

Our thanks are also addressed to Pr. GHERZOULI for his practical help and his moral support and encouragement and Dear esteemed members of the jury,we would like to express our heartfelt gratitude.

Finally, we would like to thank all the people who have participated in the preparation of this master project and the teachers who have participated in our training"

Dedication

To my dearest mother

No matter what I do or say, I will not be able to thank you as I should. Your affection covers me, your benevolence guides me and your presence at my side has always been my source of strength to face the different obstacles.

To my dearest father

You have always been at my side to support and encourage me.

May this work express my gratitude and affection.

To you my brothers and sisters Thank you for your love and encouragement.

Abstracts

ملخص

مرض فيروس كورونا (COVID-19) هو مرض معدي تم اكتشافه لأول مرة في نهاية عام 2019 وأعلن عن الجائحة في مارس 2020. أثار الجائحة هذه زيادة في المنشورات والتعليقات من مستخدمي وسائل التواصل الاجتماعي، مما يعكس ثورة من المشاعر تجاه هذا الموضوع. تتناول هذه المذكرة موضوع تحليل المشاعر ويركز على تصنيف مشاعر المستخدم من المنشورات ذات الصلة ب COVID-19الصادرة على Twitter .تمتد فترة الدراسة من مارس إلى منتصف أبريل 2020. تتم معالجة البيانات وتحليلها لغويًا باستخدام تقنيات معالجة لغات طبيعية. يتم تنفيذ تحليل المشاعر باستخدام نموذج BERT القائم على المحولات ومقارنته بمصنفات التعلم العميق والتعلم الآلي المعتادة. تم تدريب النموذج على التمييز بين ثلاث فئات من التغريدات: سلبية ومحايدة وإيجابية.

كلمات مفتاحية : فيروس كورونا, مرض معدي, وسائل التواصل الاجتماعي, تحليل المشاعر, مشاعر المستخدمين, تويتر, تحليل لغوي, تقنيات معالجة اللغة الطبيعية, نموذج BERT, المحولات, التعلم العميق, التعلم الآلي, مصنفات

Abstract

The coronavirus disease (COVID-19) is an infectious disease that was first detected at the end of 2019 and declared a pandemic in March 2020. The pandemic has sparked a surge in posts and comments from social media users, reflecting a wealth of emotions. This master project deals with the topic of sentiment analysis and focuses on classifying user sentiments from COVID-19 related posts originating on Twitter. The study period is from March to mid-April 2020. The data is processed and linguistically analyzed using multiple natural language processing techniques. Sentiment analysis is implemented by using BERT model based on transformers and comparing it with the usual deep learning and machine learning classifiers. The model is trained to distinguish between three classes of tweets: negative, neutral, and positive.

key words: Coronavirus, Infectious disease, social media, sentiment analysis, user sentiments, twitter, linguistic analysis, natural language processing, BERT model, transformers, deep learning, machine learning, classifiers.

Résumé

La maladie coronavirus (COVID-19) est une maladie infectieuse qui a été détectée pour la première fois à la fin de 2019 et a été declare pandémie en mars 2020. Cette pandémie a déclenché une vague de messages et de commentaires des utilisateurs des réseaux sociaux, reflétant une foule d'émotions. Ce projet porte sur l'analyse des sentiments et se concentre sur la classification des sentiments des utilisateurs à partir de messages liés à la COVID-19 publiés sur Twitter. La période d'étude va de mars à la mi-avril 2020. Les données sont traitées et analysées linguistiquement en utilisant plusieurs techniques de traitement du langage naturel. L'analyse des sentiments est mise en œuvre en utilisant le modèle BERT basé sur des transformateurs et en le comparant avec les classificateurs habituels du Deep Learning et du Machine Learning. Le modèle est formé pour distinguer trois classes de tweets : négatif, neutre et positif.

mot-clés: Coronavirus, maladie infectieuse, réseaux sociaux, analyse des sentiments, classification des sentiments, messages, Émotions, twitter, traitement du langage naturel, modèle BERT, transformateurs, classificateurs, deep Learning, machine Learning.

Table of Contents

List of Figures

List of Tables

List of Pseudo-codes

General Introduction

The outbreak of COVID-19 has caused a significant impact on people's lives worldwide, leading to various challenges for governments and healthcare systems. In such a scenario, social media has played a crucial role in spreading information and awareness about the pandemic. Twitter is one of the most widely used social media platforms, where people post about COVID-19 and share their experiences and opinions.

Natural Language Processing (NLP) techniques can be used to classify tweets related to COVID-19 into different categories based on their content. This classification can help in identifying trending topics, detecting misinformation, and monitoring public sentiments towards the pandemic.

In this document, we will discuss the classification of COVID-19 tweets using NLP and its potential applications in the current scenario.

1. General Context

Natural Language Processing (NLP) is a field of study that combines artificial intelligence, linguistics, and computer science to enable computers to understand, interpret, and generate human language. NLP algorithms and models are designed to process and analyze vast amounts of textual data, extracting meaningful information and patterns. It’s techniques include tasks such as text classification, sentiment analysis, named entity recognition, and machine translation

Sentiment analysis is a technique that utilizes natural language processing, machine learning, and computational linguistics to detect and extract subjective opinions and emotions from textual content. Its purpose is to decipher the emotional context within a text and classify it as positive, negative, or neutral. This approach is widely applied in evaluating customer feedback, survey responses, and product reviews. Additionally, sentiment analysis finds utility in Voice of Customer and Voice of Employee analysis, product management, customer support, and various other domains.

Deep learning, a subset of machine learning, focuses on training artificial neural networks with multiple layers to extract high-level representations and insights from complex data. Its impact has been revolutionary across various domains, such as computer vision, natural language processing, and speech recognition. One significant breakthrough in deep learning is the advent of transformer models, which have garnered considerable attention for their efficient processing of sequential data.

Unlike traditional recurrent neural networks (RNNs) that process input sequentially, transformers utilize self-attention mechanisms to simultaneously capture dependencies between different positions in a sequence. This parallel processing capability empowers transformers to effectively model long-range dependencies, resulting in exceptional performance in tasks like machine translation, sentiment analysis, and text generation. Transformers, including OpenAI's GPT, now serve as the fundamental architecture for cutting-edge language models, providing unparalleled language understanding and generation abilities.

2. Overview of the research problem and objectives

2.1 Research Problem

The research problem is to analyze the vast amount of social media data related to COVID-19 and extract useful information that can help in the fight against the pandemic. Social media platforms such as Twitter provide a wealth of information about people's opinions, behaviors, and emotions related to COVID-19. However, analyzing this data manually is time-consuming and challenging. NLP techniques can be used to automate the analysis of social media data related to COVID-19 and classify tweets using deep learning techniques specially transformers.

2.2 Objectives

The objectives of our project are :

- Understand people's opinions and emotions related to COVID-19: NLP can be used to analyze tweets related to COVID-19 and understand people's opinions and emotions related to the pandemic. For example, NLP techniques can be used to identify tweets that express fear or anxiety about the pandemic.
- Classifying what people post and tweets on social media into: Positive, Negative and Neutral tweets.

3. Document Plan

Our master project is composed of 4 chapters and organized as following:

Chapter (1): in this chapter we are going to have the literature review of our project including the background the NLP and its development through these years and its main concepts and finally the most important related works to our subject.

Chapter (2): our second chapter is about the deep learning domain and also about the transformers and their role and architecture.

Chapter (3): in this chapter, we will discuss the process of developing our tweet categorization model, which entails determining the sentiment communicated in a tweet. We will go over the various phases of creating a deep learning model for tweet classification to achieve this.

Chapter (4): This chapter provides a thorough examination of the experimental design, hyperparameters, model training procedure, outcomes, and comparison with prior research in addition to a presentation of a mobile application for the model.

CHAPTER 1

Literature Review

Introduction

In this chapter, we will explore the general idea behind the artificial intelligence and sentiment analysis in Natural Language Processing (NLP). We will cover key concepts, theories, and applications of sentiment analysis, providing a comprehensive understanding of how it works. Additionally, we will discuss the challenges and gaps in existing literature, offering insights into the complexities of analyzing sentiments in real-time. By the end of this chapter, you will gain a solid foundation in sentiment analysis and its significance in extracting valuable insights from textual data.

1.1 Artificial Intelligence

The solution in the data field is all about the artificial intelligence, in our objective we have to deal with tweets text data which has become a proper field called Natural Language Processing that we will discover next in this paper.

AI, which stands for Artificial Intelligence, encompasses a broad field within computer science that focuses on developing intelligent machines capable of executing tasks that traditionally necessitate human intelligence. It is an interdisciplinary domain that encompasses various approaches and techniques. However, the recent progress in machine learning and deep learning, specifically, is driving a transformative shift across numerous sectors of the technology industry. These advancements are enabling machines to learn from data, recognize patterns, and make informed decisions, thereby revolutionizing the potential applications of AI in diverse fields. [1]

1.2 Natural Language Processing (NLP)

As we mentioned in the previous title the Natural Language Processing became a

subfield of artificial intelligence regarding to its importance during the revolution of the world of web and social media. The text data has become large magnifically and in order to process this data a new technology is needed.

Natural language processing (NLP) is a field within computer science, and more specifically, artificial intelligence, focused on enabling computers to comprehend text and spoken language with a level of understanding close to that of human beings. [2]

1.2.1 History and Development of NLP

The realm of Natural Language Processing (NLP) and Artificial Intelligence (AI) research required almost fourteen years (until 1980) to recover from the dashed hopes instigated by overly enthusiastic proponents. This period of stagnation, in some aspects, sparked a fresh wave of innovative ideas as previous notions such as machine translation were discarded, giving way to novel concepts that propelled further exploration, notably in expert systems. The blending of linguistics and statistics, prevalent in early NLP studies, gave way to a focus on pure statistical approaches. The 1980s marked a significant shift, where simplistic approximations superseded in-depth analysis, and the evaluation process underwent a more rigorous transformation. [3]

Prior to the 1980s, a majority of Natural Language Processing (NLP) systems relied on intricate and manually crafted rules. However, a transformative era for NLP emerged in the late 1980s, driven by two factors: the steady advancement of computational power and the adoption of Machine Learning algorithms. While some early Machine Learning algorithms, like decision trees, produced systems resembling the traditional rule-based approaches, research gradually shifted towards statistical models. These statistical models enable the generation of soft, probabilistic decisions. In the 1980s, IBM played a significant role in developing numerous intricate and successful statistical models. [3]

During the 1990s, there was a significant surge in the adoption of statistical models for Natural Language Processing (NLP) analyses. These pure statistical methods proved to be highly beneficial in effectively handling the vast amount of textual data available online. N-Grams, which involve identifying and tracking clusters of linguistic data numerically, emerged as valuable tools in this era. In 1997, the introduction of LSTM recurrent neural network (RNN) models marked a significant development, particularly in voice and text processing. Subsequently, these models found their niche in 2007. At present, neural network models are considered the forefront of research and

development in NLP, driving advancements in understanding and generating text and speech. [3]

In 2011, Apple introduced Siri, which quickly gained recognition as one of the pioneering NLP/AI assistants accessible to the general public. Siri operates through a two-step process. First, the Automated Speech Recognition module translates the user's spoken words into interpreted digital concepts. Next, the Voice-Command system matches these concepts with predefined commands, thereby triggering specific actions. As an illustration, when Siri asks a question like, "Would you like to hear your balance?" it comprehends responses such as "Yes" or "No" and takes appropriate actions based on the user's choice. [3]

Machine Learning techniques enable NLP systems to handle variations in the owner's speaking pattern, eliminating the need for an exact match with predefined expressions. Instead, the system focuses on capturing the meaning accurately, allowing for reasonable similarity in the sounds. Through a feedback loop, NLP engines can enhance the precision of their translations and expand their vocabulary. A proficiently trained system would comprehend queries like "Where can I get help with Big Data?", "Where can I find an expert in Big Data?", or "I need help with Big Data," and provide the relevant response, demonstrating its ability to understand the intent behind different phrasings. [3]

1.2.2 Review of key Concepts in NLP

Natural Language Processing (NLP) is a field that deals with the interaction between humans and computers using natural language. To make sense of language and provide useful information, NLP is based on several key concepts and theories:

- **Syntax**: This is the study of the rules governing the structure of sentences and phrases. It focuses on the arrangement of words, phrases, and clauses to form a grammatically correct sentence. Syntax helps computers understand the structure of language and how different parts of a sentence relate to each other.
- **Semantics**: Semantics is the study of the meaning of words and sentences. It helps computers understand the meaning of language by analyzing the relationships between words, their context, and their intended meanings. This includes understanding word sense disambiguation, synonymy, polysemy, and other related concepts.

- **Pragmatics**: Pragmatics is the study of the use of language in context. It is concerned with how speakers use language to communicate meaning beyond the literal meaning of words. Pragmatics includes understanding conversational implicature, presuppositions, speech acts, and other related concepts.
- **Discourse analysis**: Discourse analysis is the study of the structure and function of larger units of language such as conversations and texts. It helps computers understand how meaning is constructed across multiple sentences and how discourse is structured.
- **Corpus linguistics**: Corpus linguistics is the study of language as it is used in large collections of texts. It involves using statistical methods to analyze language patterns and frequencies, including collocations, lexical associations, and other related concepts.

By understanding these key concepts and theories, NLP algorithms can be designed to process, analyze, and generate natural language in a way that is useful for various applications such as language translation, sentiment analysis, extremism detection, and more

1.2.3 Application domains of NLP

NLP encompasses various domains, including machine translation, information extraction, question answering, text summarization, named entity recognition (NER), and sentiment analysis. Sentiment analysis specifically focuses on analyzing and classifying the sentiment expressed in text, helping businesses understand customer satisfaction, market sentiment, and public opinion.

1.3 Sentiment analysis

The sentiments analysis appeared in the early 2000s and it is a data analysis technique linked to the NLP, which consists in analyzing body texts in order to extract opinions or feelings and a confidence rate related to them.

The sentiments analysis is able to determine the polarity of a sequence of words, i.e., identify a positive, neutral or negative trend, as well as the sarcasm or irony of that sequence.

This technique is mainly used to analyze online content such as customer reviews on commercial sites, discussions in forums, blog texts, and especially exchanges on social

networks. Their analysis makes it possible to transform unstructured information into concrete and interpretable data on the opinions of Internet users in relation to products, brands, various facts, experiences or any other subject that makes it possible to express an opinion. Analysis of these types of content can help to identify social or business trends, inform companies to facilitate their strategic decisions, optimize their performance and increase their income.

1.3.1 Sentiment & Opinion

In Merriam-Webster's collegial dictionary [4], sentiment is defined as an attitude, thought, or judgment aroused by sentiment, while opinion is defined as a view, judgment, or evaluation formed in the mind on a particular issue. The difference is quite subtle and each of them contains some elements of the other.

The definition of Sentiment Analysis: Sentiment analysis is a procedure to follow the opinions of clients/authors around a specific object or subject. It includes the creation of a framework to gather and review opinions on the item issued in blog entries, remarks, audits or tweets.

1.3.2 Sentiment analysis approaches

There are several approaches to sentiment analysis, which involve analyzing and classifying text to determine the sentiment expressed within it. Here are some commonly used approaches:

- Rule-based Approach: This approach involves defining a set of linguistic rules or patterns to identify sentiment-bearing words or phrases. These rules can be handcrafted or derived from existing resources such as sentiment lexicons. The sentiment of the text is then determined based on the presence and context of these sentiment indicators.
- Machine Learning Approach: In this approach, machine learning algorithms are trained on labeled datasets, where each text is associated with a sentiment label (e.g., positive, negative, neutral). Features are extracted from the text, such as word frequencies, n-grams, or even word embeddings, and used to train a classifier or regression model. Commonly used machine learning algorithms include Naive Bayes, Support Vector Machines (SVM), and Random Forests.
- Deep Learning Approach: Deep learning techniques, particularly neural networks, have gained popularity in sentiment analysis. Models like Recurrent Neural

Networks (RNNs), Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNNs) can effectively capture the sequential and contextual information in text data. These models can be trained end-to-end, taking raw text as input and predicting the sentiment as output.

- Lexicon-based Approach: Lexicon-based methods utilize sentiment lexicons or dictionaries, which contain words or phrases along with their associated sentiment scores. Each word in the text is matched against the lexicon, and sentiment scores are aggregated to determine the overall sentiment. Techniques like VADER (Valence Aware Dictionary and Sentiment Reasoner) are commonly used, which also consider intensity modifiers and punctuation to improve sentiment analysis accuracy.
- Hybrid Approaches: Hybrid approaches combine multiple techniques mentioned above to leverage the strengths of each approach. For example, a hybrid approach might utilize rule-based methods for handling specific linguistic patterns, while also incorporating machine learning or deep learning models for more complex sentiment analysis tasks.

1.4 Related Works

By employing sentiment analysis, businesses can extract valuable insights from extensive quantities of unstructured data derived from various online sources, including social media platforms, emails, chats, blogs, and forums. It is worth noting that unstructured information constitutes a substantial portion, estimated to be around 80-90%, of the entire digital content landscape, as highlighted in a CIO research conducted in 2019. [5]

The current development of the internet has made it easier to access and acquire large volumes of data, which can then be analyzed to generate valuable information [6], [7]. In the domain of sentiment analysis, various studies have utilized traditional machine learning algorithms such as Support Vector Machines (SVM) and Naïve Bayes [8], [9], [10]. These studies primarily focused on sentiment analysis of IMDB movie reviews, with most of them utilizing English datasets and only a few incorporating datasets in Bahasa Indonesian [11], [12], [13], [14].

Additionally, there are studies that have explored sentiment analysis using datasets in English, Chinese, and Indian languages [15], while others have specifically employed the Recurrent Neural Network (RNN) algorithm to analyze sentiment in the Malayalan language, achieving an accuracy of 80% that can be further improved with larger

datasets [16]. Another study assessed the performance of RNN and LSTM models in classifying analytical sentiments in movie reviews, obtaining an accuracy of 86.74% but also identifying susceptibility to overfitting [17]. Furthermore, researchers have investigated sentiment analysis on social media using approaches such as Convolutional Neural Networks (CNN) and word2vec, achieving an accuracy of 45.4% on public film review datasets [18]. Deep learning and word2vec have also been proposed as methods for analyzing sentiment in product reviews [18].

In recent times, Transformers have emerged as efficient models due to their ability to parallelize data, outperforming previous deep learning models like recurrent neural networks [19]. One research paper introduces the application of Transformer methods to address the sentiment analysis sub-task of the REST-MEX@IberLef 2022 shared task [20], which was part of the IberLEF 2022 conference. The task involved developing models to analyze Mexican tourist texts and provide recommendations, classify sentiments, and predict the Covid semaphore. The sentiment analysis sub-task aimed to identify the satisfaction level and type of place described in tourist opinions about Mexican locations. The polarity of opinions was rated on a scale of 1-5 (1 being the worst and 5 being the best), while the type of place could be a "Restaurant," "Hotel," or "Tourist attraction" [21]. The best results were obtained with RoBERTaESP, a pretrained transformer model for Spanish language based on RoBERTa (a variant of the BERT model). This model obtained an accuracy of 76.25% in the Polarity task and 98.84% in the Attraction task in the competition, being the second-best result in the overall task.

Conclusion

In this chapter, we present a comprehensive overview of sentiment analysis and its applications. By engaging with this chapter, a solid understanding of sentiment analysis and its significance will be developed. The knowledge and awareness of the complexities involved in analyzing sentiments in text data are now obtained. This foundation will serve as a stepping stone for further exploration and application of sentiment analysis techniques in the upcoming chapters.

CHAPTER 2

Deep Learning

Introduction

Deep learning is a subset of machine learning that utilizes neural networks, a type of algorithm designed to mimic the functioning of the human brain. Neural networks are structured with multiple layers stacked on top of each other, enabling high-level data processing through a series of linear and nonlinear transformations. This hierarchical architecture allows neural networks to extract increasingly complex features and patterns from the input data. [22]

Deep neural networks, a specific type of neural network, are characterized by having tens or even hundreds of layers stacked on top of each other, resembling a hierarchical structure. This depth enables deep neural networks to excellent in various domains such as speech recognition, image processing, and text analysis. By utilizing the multiple layers, these networks can learn hierarchical representations, capturing intricate relationships and performing sophisticated computations. The depth of deep neural networks has proven to be a significant breakthrough, enhancing their capability to handle complex tasks and achieve state-of-the-art performance in various domains.

2.1 Deep Learning Neuron

Neurons play a crucial role in deep learning models, and a comprehensive understanding of their functioning is essential to grasp the intricacies of deep learning [23].

Neurons in deep learning models are inspired by the biological neurons found in the human brain, and their significance in modern deep learning models cannot be overstated [23].

In deep learning, neurons act as computational units through which data and computations flow [23].

The functioning of neurons can be summarized as follows:

- Input signals: Neurons receive one or more input signals, which can originate from either the raw dataset or neurons positioned in a previous layer of the neural network.
- Computations: Neurons perform calculations on the input signals they receive. These computations typically involve applying weights to the input signals and applying an activation function, which introduces non-linearity to the neuron's output.
- Output signals: Neurons transmit output signals to neurons located in subsequent layers of the neural network through connections called synapses. These output signals carry information that has been processed and transformed by the computations within the neuron.

Here is a diagram of the functionality of a neuron in a deep learning neural net:

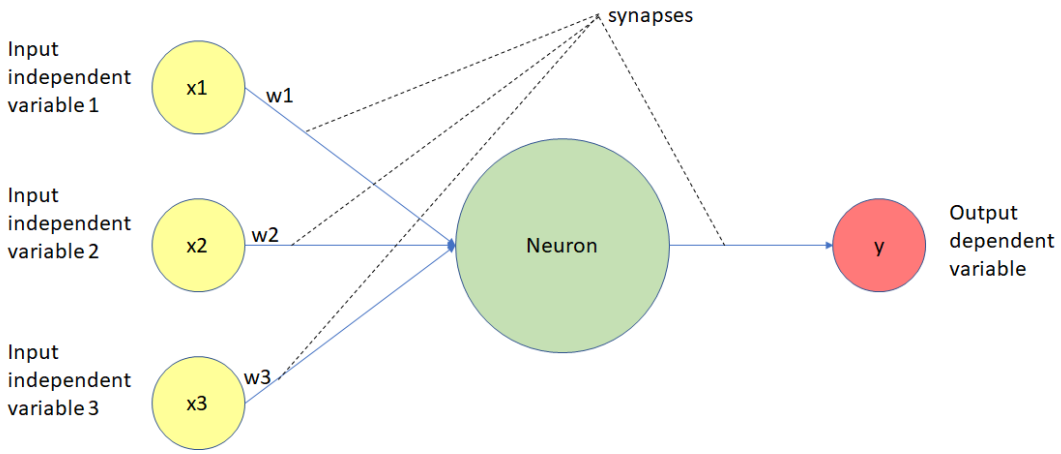

Figure 1.Deep learning neuron [23]

As it seems in the above figure, neurons within a deep learning model possess the capability to establish synapses that connect with multiple neurons in the preceding layer. These synapses are assigned individual weights, which influence the relative significance of the preceding neuron within the broader neural network. [23]

The concept of weights holds significant importance in deep learning, as the adjustment of a model's weights serves as the primary mechanism for training deep learning models.

Following the reception of inputs from neurons in the preceding layer of the model, a neuron proceeds to aggregate these signals by multiplying them with their respective

weights. The resultant values are then forwarded to an activation function for further processing, like this:

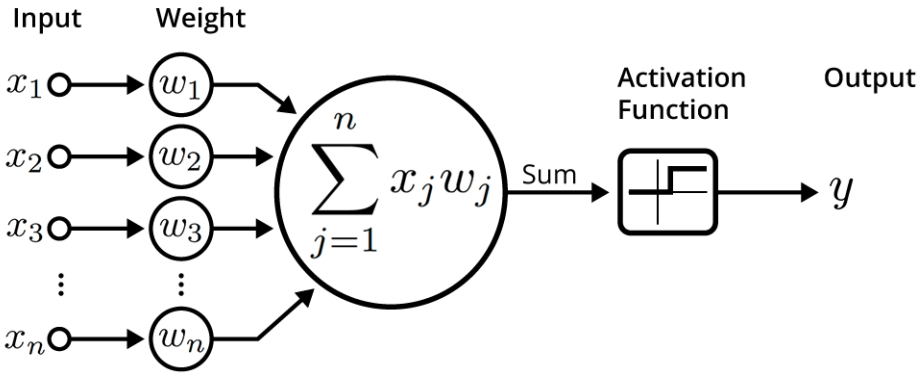

Figure 2.Activation function in a neuron [23]

Activation functions are indeed a fundamental component of deep learning models. They play a crucial role in calculating the output value for a neuron, which is then transmitted to the next layer of the neural network through another synapse. [23]

Activation functions are responsible for introducing non-linearity into the computations performed by neurons. This non-linearity is essential for enabling deep learning models to learn and represent complex patterns and relationships within the data.

There are several types of activation functions commonly used in deep learning models. Here are four main types:

- Threshold functions: These activation functions produce a binary output based on whether the input value exceeds a certain threshold or not. They are simple but rarely used in modern deep learning due to their limited expressive power.
- Sigmoid functions: Sigmoid functions, such as the logistic function, map the input value to a continuous range between 0 and 1. They are widely used as activation functions in the past but have been largely replaced by other functions due to their vanishing gradient problem.
- Rectifier functions (ReLUs): Rectified Linear Units (ReLUs) are a popular choice for activation functions in deep learning. They output the input value as it is if it is positive, and zero if it is negative. ReLUs alleviate the vanishing gradient problem and have been shown to work well in many deep learning architectures.
- Hyperbolic Tangent functions: Hyperbolic tangent functions (tanh) are similar to sigmoid functions but map the input value to a range between -1 and 1. They have a steeper gradient compared to sigmoid functions, which can aid learning in certain scenarios.

2.2 Neural Networks

Neural networks were limited by computational power and therefore limited in complexity. However, advances in big data analysis have led to the creation of larger and more sophisticated neural networks, allowing computers to observe, learn and react to complex situations faster than humans.

The neural networks has been in a huge evolution since then created in 1940s, today there is a huge number of algorithms under the title of neural networks algorithms. Firstly we have to mention and discuss the racine neural network algorithm which is feed forward neural networks which are ANN algorithm

2.2.1 Artificial neural networks (ANNs)

Artificial neural networks (ANNs) draw inspiration from the information processing and distributed communication nodes observed in biological systems. However, ANNs possess distinct differences when compared to biological brains. Notably, artificial neural networks typically exhibit a static and symbolic nature, whereas the biological brains of most living organisms exhibit dynamic (plastic) and analog characteristics. [24]

An artificial neural network (ANN) comprises three or more interconnected layers. The input layer consists of neurons that serve as the initial layer for receiving information. These neurons transmit the received information to subsequent layers, which then propagate the transformed information towards the final output layer. The hidden layers, situated between the input and output layers, dynamically adapt and modify the information as it is passed from one layer to another. The remarkable characteristic of ANNs is their capacity for each layer to function both as an input and output layer, thereby enabling the network to comprehend increasingly intricate inputs.

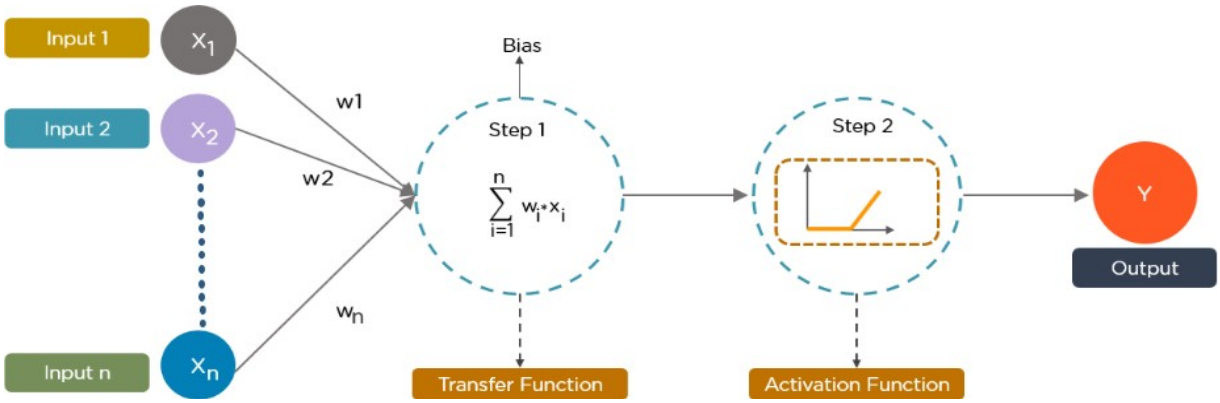

Figure 3.Functioning of neural networks [24]

This type of neural networks has many limits such as:

- Lack of sequential memory: ANNs lack the ability to retain information from previous inputs in a sequence. They treat each input independently, which can be a limitation in tasks that require capturing temporal dependencies or processing sequential data.
- Fixed input size: ANNs typically require a fixed-size input, meaning they struggle with variable-length input sequences. This restricts their effectiveness in tasks where the length of the input varies or when the order of inputs is crucial.
- Computational complexity: Training ANNs can be computationally expensive, especially when dealing with deep networks that have many layers and parameters. The training process may require significant computational resources and time, making it less efficient for certain applications.

It's important to note that while ANNs have these limitations, they still excellent in various domains and have been successful in tasks such as image classification, pattern recognition, and regression problems.

Knowing that in our case study we have to deal with the sequantial data as language to preform tweets sentiment analysis, and because of this problem in the NLP field in general the RNNs was created.

2.2.2 Recurrent Neural Networks (RNNs)

The RNN was created mainly to effectively process and model sequential data. Traditional feedforward neural networks, such as Artificial Neural Networks (ANNs), lack the ability to handle sequential information and capture dependencies over time. RNNs were specifically designed to address these limitations and cater to tasks involving sequential data.

The developers aimed to create a neural network architecture that could retain and utilize information from previous inputs to make contextually informed predictions or decisions. By introducing recurrent connections and internal memory, RNNs became capable of capturing temporal dependencies, handling variable-length input sequences, and modeling long-term relationships within sequential data.

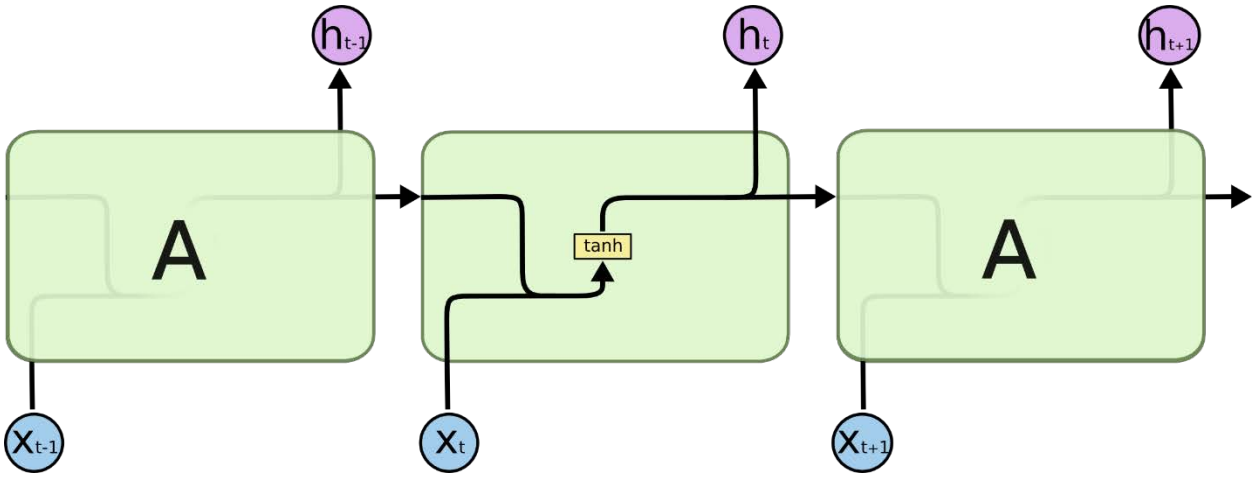

Figure 4.RNN architecture [25]

RNNs have recurrent connections that allow them to retain and utilize information from previous steps in the sequence. This enables them to capture dependencies and context over time. RNNs operate by recursively applying the same set of weights to each step in the sequence, using the current input and the previous hidden state as inputs to produce an output and update the hidden state. This recurrent structure allows RNNs to model sequential patterns, make predictions at each step, and maintain an internal memory that can store information relevant to the sequence. In this way, RNNs can effectively analyze and generate sequences, making them well-suited for tasks like natural language processing, speech recognition, and time-series analysis.

But RNNs still have a limitation in capturing long-term dependencies due to the vanishing gradient problem in long sequences like texts. To overcome this, LSTM which are type of RNNs was invented, introducing memory cells and gating mechanisms that enable better retention of information over extended sequences.

2.2.2.1 Long Short-Term Memory (LSTM)

LSTMs, a specific type of recurrent neural network (RNN), possess the ability to learn long-term dependencies in data. Hochreiter and Schmidhuber first introduced LSTMs in 1997, marking a significant milestone in the field. [25]

LSTM layers consist of four crucial components known as gates: the input gate, forget gate, output gate, and cell state. The input gate regulates the amount of the current input that contributes to the cell state. The forget gate determines how much of the previous cell state is discarded. Lastly, the output gate controls the part of the cell state that is emitted as the output of the layer.

All recurrent neural networks are characterized by their inherent structure, which consists of a chain of repeating modules comprising neural networks. In the case of standard RNNs, these repeating modules exhibit a simple architecture, often represented by a solitary tanh layer. [25]

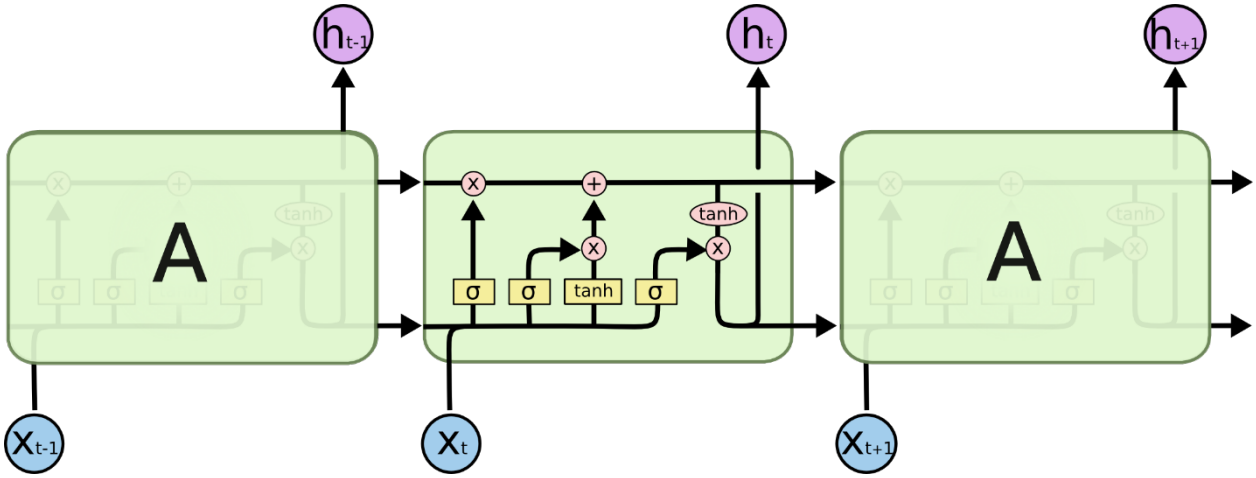

Figure 5.Lstm architecture [25]

But the lstm has the same problem in the grand scale data which is capturing dependencies between distant words or positions in a sequence. While LSTM processes sequential data step by step, it still faces limitations in modeling long-range dependencies effectively.

So the Transformer architecture introduced after years which solves the problem of dependencies in a sequence for ever by the concept of self-attention mechanisms that allow to capture global dependencies in parallel .

2.2.3 Transformers

The Transformer was proposed in the paper 'Attention is All You Need' [26]. as a specific type of deep learning architecture that has revolutionized various natural language processing (NLP) tasks. Transformers are neural network models that excellent in capturing long-range dependencies and understanding contextual relationships in sequential data, such as sentences or paragraphs. This architecture has gained significant attention and popularity due to its ability to process and generate text with remarkable accuracy and fluency.

The key innovation introduced by Transformers is the self-attention mechanism, also known as scaled dot-product attention. This mechanism allows the model to weigh the importance of different words or tokens in a given sequence, enabling it to focus on the most relevant context for each word. By leveraging attention-based computations,

Transformers can capture intricate patterns and semantic relationships across long distances, leading to improved performance in tasks like machine translation, text summarization, sentiment analysis, and language generation.

2.2.3.1 The transformer architecture

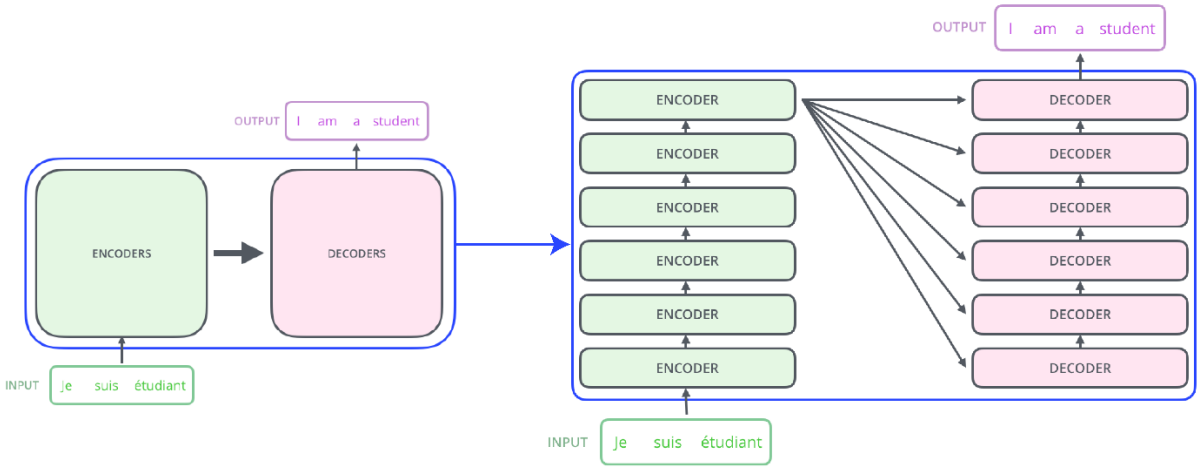

Figure 6.A Transformer-encoder-decoder architecture [27]

The Transformer architecture consists of an encoder-decoder framework that is based on self-attention mechanisms. Unlike traditional recurrent neural networks (RNNs) that process sequential data sequentially, Transformers can parallelize the computation, making them highly efficient for training and inference.

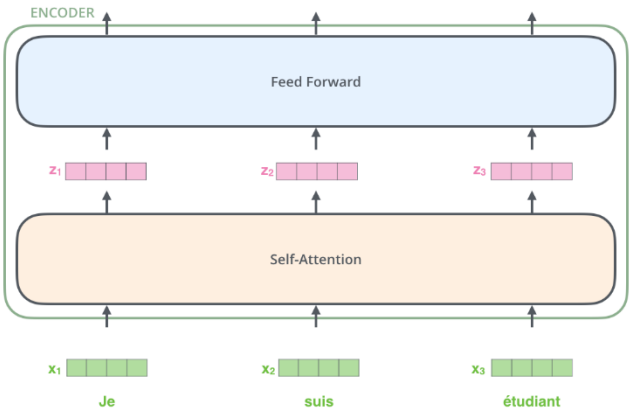

Figure 7.Encoder architecture [27]

The encoder in the Transformer architecture takes an input sequence and generates a sequence of contextualized representations for each word or token. Through self-attention, the encoder assigns weights to the words in the input sequence, allowing the

model to attend to the most relevant information at each position. This enables the Transformer to capture dependencies between words that are far apart, effectively overcoming the limitations of sequential processing.

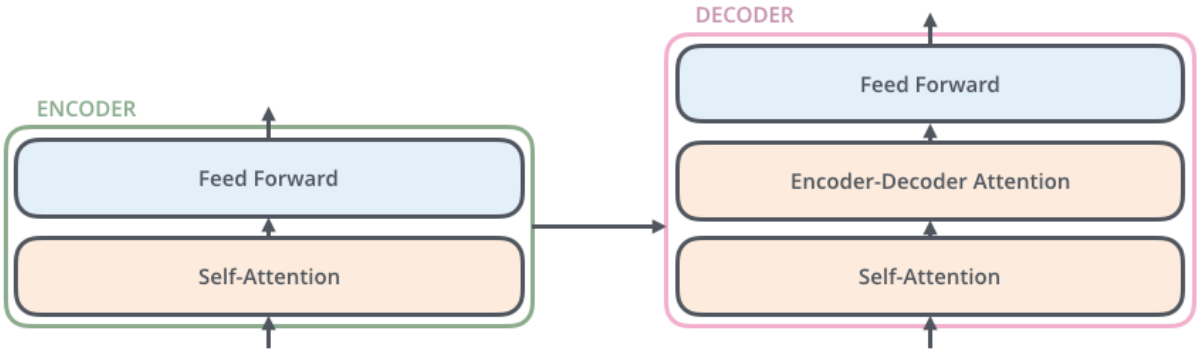

Figure 8.Decoder pass-through [27]

The decoder, on the other hand, utilizes the encoder's output and employs an additional attention mechanism to generate the output sequence. During the decoding process, the model attends to the encoded representations of the input sequence and learns to predict the next word or token based on the context. By using self-attention both within the encoder and between the encoder and decoder, Transformers achieve a comprehensive understanding of the input sequence and produce coherent and contextually appropriate outputs.

2.2.3.2 BERT Model

In 2018, a revolutionary advancement called BERT (Bidirectional Encoder Representations from Transformers) emerged in the feild of natural language processing. Developed by researchers Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova at Google AI Language [28], BERT leverages the capabilities of transformer-based architectures and bidirectional training. This pioneering approach enabled BERT to attain remarkable performance on a wide range of NLP tasks, setting new benchmarks in the field.

BERT created by leveraging the Transformer architecture. It involves pretraining the model on a large corpus of unlabeled text data, where BERT learns to predict masked words and sentence relationships. Key modifications include tokenization, special tokens, and input representations. BERT utilizes token embeddings, segment embeddings, and position embeddings to encode meaning, sentence differentiation,

and positional information. After pretraining, BERT is fine-tuned for specific tasks by adding task-specific layers on top of the pretrained model.

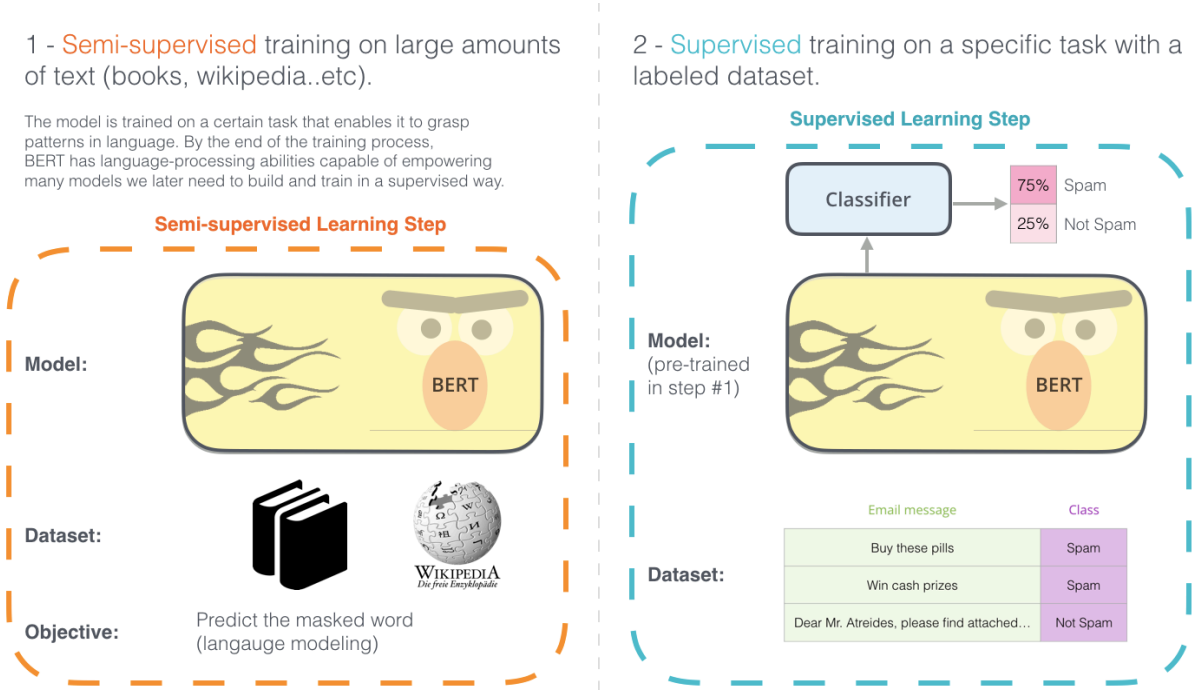

Figure 9.BERT training [29]

The BERT model comes in two sizes, each one of them has a number of encoder layers referred to as Transformer Blocks. The Base version consists of 12 encoder layers, while the large version has 24. Additionally, these versions feature larger feedforward networks with 768 and 1024 hidden units, respectively. Moreover, they incorporate more attention heads, with 12 and 16 attention heads, respectively. In comparison, the default configuration of the Transformer in the initial paper 'attention is all you need' [26] comprises 6 encoder layers, 512 hidden units, and 8 attention heads. [29]

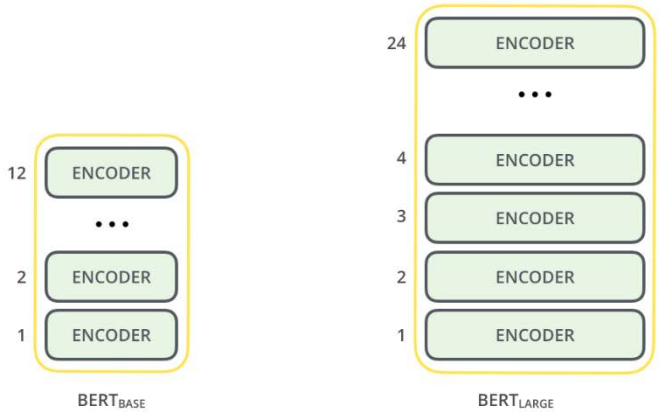

Figure 10.Representation of Bert model sizes [29]

In this work, we employ BERT as a powerful tool for language understanding and analysis. We leverage its ability to capture contextual information and semantic relationships within text data. By fine-tuning BERT on our specific task, we aim to achieve enhanced performance .The utilization of BERT provides us with a robust foundation to explore and advance the field of NLP in our research.

Conclusion

The architecture of the Transformer, with its attention mechanisms and parallel processing capabilities, has played a crucial role in the success of various NLP tasks. It has not only outperformed traditional models but also paved the way for advancements in tasks like machine translation, natural language understanding, and text generation. The Transformer's ability to capture long-range dependencies and contextual relationships has made it a cornerstone in the field of deep learning and NLP, and because of that we have addopted this architecture in our model building in the next chapter.

CHAPTER

3

Methodology

Introduction

In this chapter, we will explore the methods and the process of building our tweet classification model which involves identifying the sentiment expressed in a tweet. To accomplish this, we will discuss the various stages in developing our deep learning model for tweet classification.

3.1 Proposed Model

In order to develop our classifier model, we propose a methodology that consist of 4 phases

These phases include data observation, preprocessing, model building and evaluation of the model.

In the first phase we will discuss the data source and its form. Secondly, we will explore text the cleanup techniques as preprocessing step which is essential for preparing tweet data for analysis. Next, we will discuss the next step in preprocessing phase which is tokenization. These techniques are necessary for transforming the raw text data into a format that can be used as input to a deep learning model.

After preprocessing the data, we will discuss the architecture of the model that we

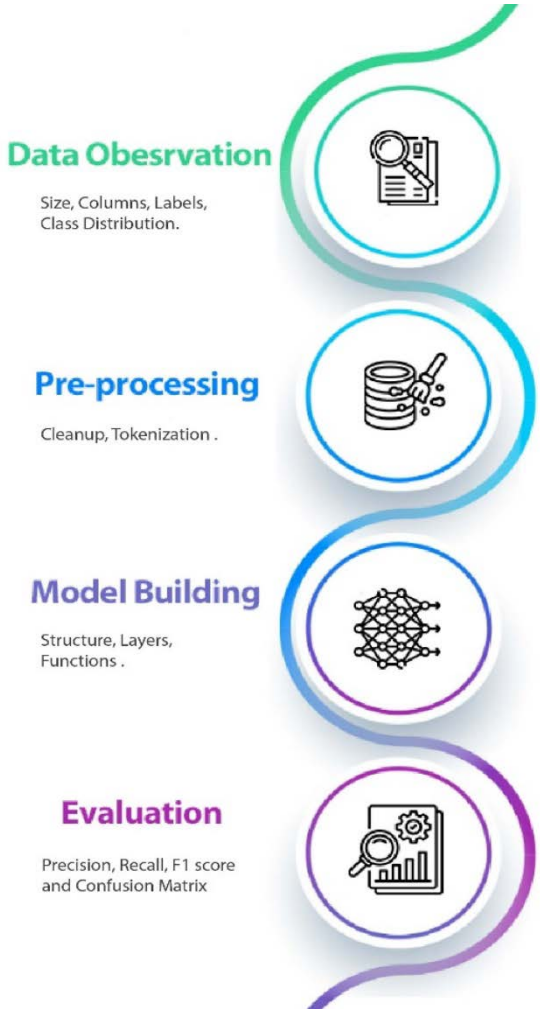

Figure 11.Graphical representation of The proposed model

used for tweet classification. We will explain the role and functionality behind each layer in the neural network, beside the optimization technique that we adopted in our model to perform better.

Finally, we will evaluate the model performance and represent the results with different manner.

3.2 Data Observation

The field of sentiment analysis lack high-quality datasets, as they play a vital role in developing and evaluating effective models. These datasets serve as the fundamental building blocks for training and testing deep learning models in this domain.

The dataset used in this project was sourced from Kaggle thanks to 'AMAN MIGLANI'. The dataset titled " Coronavirus tweets NLP - Text Classification" [30]. It was specifically selected for its relevance to the study's objective of sentiment analysis in the context of the COVID-19 pandemic. It contains a collection of tweets data related to COVID-19.

```
# Observing the dataset
print(f'Columns names {df.columns.values}')
print(f'Dataframe shape {df.shape}')
```

```
Columns names ['UserName' 'ScreenName' 'Location' 'TweetAt' 'OriginalTweet' 'Sentiment']
Dataframe shape (41157, 6)
```

Pseudo-code 1.Observation of Dataset shape

The dataset comprises of 41157 number of records and each record has 6 columns of data user name, screen name, location, date of tweet, the tweet itself and the sentiment.

UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
4934	49886	Kampala, Uganda	17-03-2020	I think there is something that the Gov't is n...	Negative
14829	59781	Devon	20-03-2020	I donÂ□t understand I DO NOT UNDERSTAND panic ...	Extremely Negative
17536	62488	Darwin, NT - Australia.	21-03-2020	So @Coles and @Woolworths are on this high hor...	Positive
7867	52819	Gloucester, New South Wales	18-03-2020	Leadership during the COVID-19 Crisis: Be a mo...	Positive
41942	86894	NaN	11-04-2020	@MilenaRodban Been exchanging lusty looks with...	Extremely Negative

Figure 12.Dataset example simples

The sentiment labels are categorized as positive, negative, neutral, extremely positive and extremely negative indicating the sentiment expressed within the corresponding tweet. refined to include only positive, negative and neutral categories to align the other researches that have been conducted on this dataset on different studies.

```
# Refine the classes
df['Sentiment'] = df['Sentiment'].replace({'Extremely Negative': 'negative',
                                           'Negative': 'negative',
                                           'Neutral': 'neutral',
                                           'Positive': 'positive',
                                           'Extremely Positive': 'positive'})
```

Pseudo-code 2.Refine classes

In order to work with this dataset, we have to make sure that no relevant data are missing so we did missing values check and we found that there are no missing values, except for the attribute 'location' which has 8590 missing values. This column is considered irrelevant.

```
# Check null values in each column
df.isnull().sum()
```

```
UserName            0
ScreenName          0
Location         8590
TweetAt             0
OriginalTweet       0
Sentiment           0
dtype: int64
```

Pseudo-code 3.Check missing values

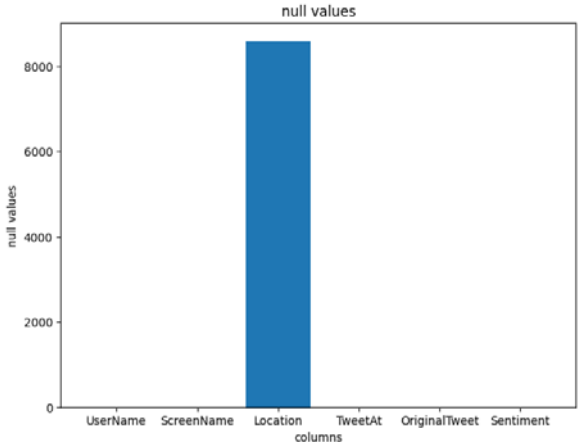

Figure 13.Graphical representation of null values

Finally, we did a check on the labels distribution to avoid the imbalance problem which could pose challenges in the training because the model may become biased towards the majority class.

```
# Observe classes
df['Sentiment'].value_counts()
```

```
positive    18046
negative    15398
neutral      7713
```

Pseudo-code 4.Observation of the distribution of dataset classes

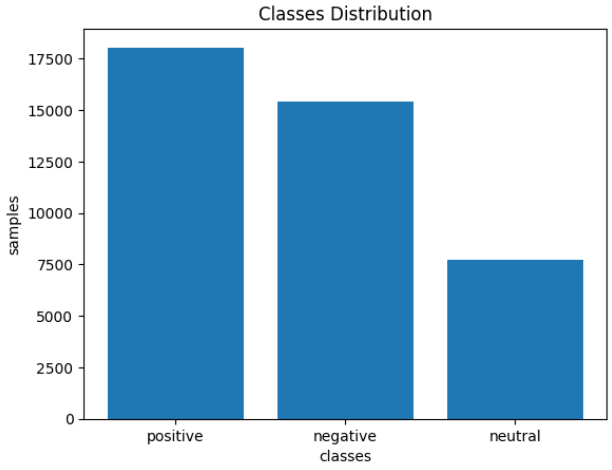

Figure 14.Graphical representation of classes distribution

3.3 Data cleanup

The text extraction and cleanup is an important step in preparing data for deep learning model training. It involves filtering raw data into a text format in way that all the text is meaningful and make it clear to the model to define his patterns in classification. cleanup methods are essential in practically all NLP tasks because it considerably improves the performance of the classification algorithms. This step is significantly important in our research as Twitter data is extremely noisy, which contain a large amount of irrelevant information and making it difficult to extract useful insights in the presence of emojis, URLs, retweets, tags, hashtags and the internet slang that uses special words that are commonly used on social media.

In our cleaning pipeline we have came across serval steps to completely clean all the data before the training phase. the following cleaning steps are adopted:

- Deleting irrelevant: Deleting irrelevant columns helps streamline the data and remove unnecessary information that may not contribute to the task at hand, such as user name, screen name, tweet date, location in this case.

```
#drop the unnecessary columns
df=df[['tweet','label']]
```

Pseudo-code 5.Dropping unnecessary columns

- Deleting retweets: Retweets are essentially duplicate tweets, and removing them will help reduce noise in the data and prevent duplicate tweets from influencing the model.

```
# Deleting retweets
df = df.loc[~df['OriginalTweet'].str.startswith("RT @")]
```

Pseudo-code 6.Deleting retweets

- Converting to lowercase: Converting all the text to lowercase helps standardize the text and ensure that the model treats words with the same spelling but different capitalization as the same word.

```
# Convert to lowercase
text = text.lower()
```

Pseudo-code 7.Converting to lowercase

- Removing URLs: URLs can be considered noise in the data and removing them can help improve the model's accuracy.

```
# Remove URLs
text = ' '.join(word for word in text.split() if not word.startswith('http'))
```

Pseudo-code 8.Removing URLs

- Remove mentions (@) and hashtags (#): Mentions and hashtags can be important for social media analysis, but for our specific task of sentiment analysis, they are not relevant. Removing them can help simplify the text and reduce noise.

```
# Remove mentions (@) and hashtags (#)
text = ' '.join(word for word in text.split() if not word.startswith('@') and not word.startswith('#'))
```

Pseudo-code 9.Removing hashtags and mentions

- Remove emojis: Emojis can add context and meaning to text, but they can also be considered noise in some cases. For our specific task, removing them may help simplify the text and make it easier for the model to classify since there is a huge number of emojis and a lot of them have an ambiguous meaning.

```
# Remove emojis
text = emoji.replace_emoji(text, '')
```

Pseudo-code 10.Removing emojis

- Handle internet slang using a dictionary: internet slang can be difficult for models to understand, so it's important to handle slang words before training the model. Using a dictionary from analyticsvidhya website [31] to map the words to their full form.

```
# Handle abbreviations
# Dictionary of English Contractions   from   https://www.analyticsvidhya.com
abbreviations = {"bro": "brother", "fyi":"for your information", "u": "you"}
```

Pseudo-code 11.Handling internet slang

- Remove HTML tags: HTML tags are not relevant to the text and removing them can help simplify the text and improve the model's accuracy.

```
# Remove HTML tags
text = re.sub('<[^<]+?>', '', text)
```

Pseudo-code 12.Removing HTML tags

- Remove special characters: Special characters will also be considered noise in NLP tasks, and removing them can help simplify the text and improve the model's accuracy.

```
# Remove special characters and digits
text = ''.join(char for char in text if char.isalnum() or char.isspace())
```

Pseudo-code 13.Removing special characters and digit

- Remove numbers: Similar to special characters and digits, numbers are not relevant to our specific task and removing them will help simplify the text and reduce noise.

```
# Remove numbers
text = ' '.join(word for word in text.split() if not word.isdigit())
```

Pseudo-code 14.Removing numbers

- Remove extra whitespaces: Extra whitespaces can be considered noise in the text, and removing them will help simplify the text and make it easier for the model to classify.

```
# Remove extra whitespaces
text = ' '.join(text.split())
```

Pseudo-code 15.Removing extra whitespaces

3.4 Tokenization

In this section, we will discuss the preprocessing techniques used to prepare the tweet data for deep learning model training. We employed the BERT preprocessor tokenizer to tokenize the raw text data into subword units (tokens), which is an effective technique for dealing with the challenges of out-of-vocabulary words and rare words that often occur in social media text data.

By utilizing the BERT preprocessor, we can efficiently process and transform the tweet data into a format that can be used as input to the BERT base model.

To achieve the transformation of raw data to format that can be used as input to the BERT model, the BERT preprocessor utilizes a tokenization process.

Tokenization is the process of splitting a text into individual words or subwords, which are then assigned unique integer values called tokens. In BERT, the tokenization process involves breaking down the input text into a sequence of word pieces or subwords, rather than individual words.

BERT uses WordPiece tokenization, which is a type of subword tokenization. This means that the text is first split into words, and then each word is further divided into smaller units called subwords. For example, the word "university" might be broken down into the subwords "uni" and "versity".

At the end we get from the BERT tokenizer a list of tokens indices from the vocab file of the BERT model and a clear readable token of the input_ids and special tokens to mark the beginning of the sentence [CLS] and end of a sentence [SEP].

```
BERT_preprocessor = hub.load("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
text_inputs = ["let's remember that together, we can overcome any challenge."]
tokenized_inputs = BERT_preprocessor.tokenize(text_inputs)
print(tokenized_inputs.flat_values)
```

```
tf.Tensor([2292 1005 1055 3342 2008 2362 1010 2057 2064 9462 2151 4119 1012], shape=(13,), dtype=int32)
```

Pseudo-code 16.BERT tokenizer

3.5 Architectural Considerations

The success of deep learning models in natural language processing tasks has led to a surge of interest in using these models for social media analysis. In this section, we describe the architecture of the deep learning model used for tweet classification. The model consists of several layers that extract relevant features from the input text, and a final classification layer that maps the extracted features to the target classes. We begin by providing an overview of the model architecture and then delve into the details of each layer.

3.5.1 Overview

The model architecture starts with an input layer that takes in text data as input. The input shape is a scalar with a string data type.

After the input layer we have the BERT preprocessor tokenizer layer.

Next, a Keras layer from TensorFlow Hub is used for preprocessing the input text data. The preprocessed text data is then passed to the BERT encoder layer. This layer is also a Keras layer from TensorFlow Hub that uses the BERT algorithm for encoding the input text data.

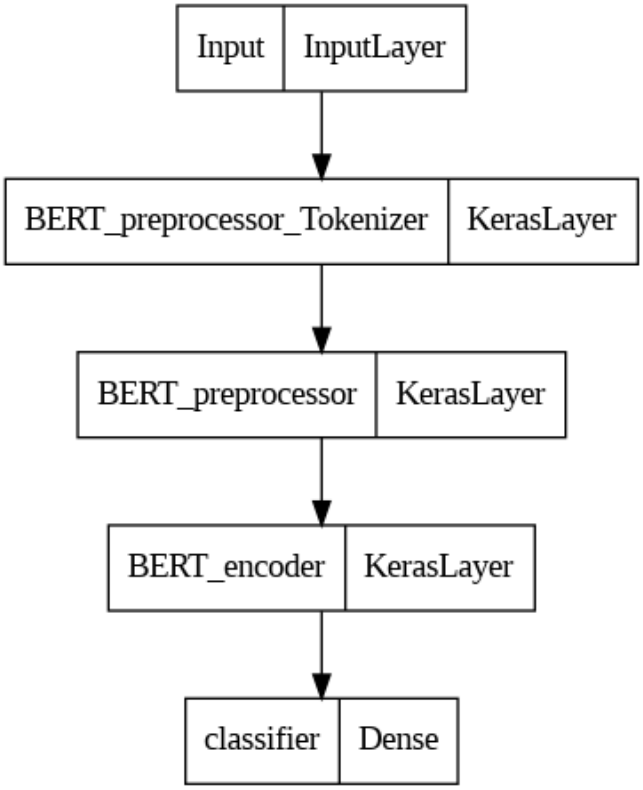

Figure 15.Model architecture

The encoder layer is trainable, which means that the weights of this layer can be updated during the training process (transfer learning).

The output of the BERT encoder layer is a dictionary containing two keys: "sequence_output" and "pooled_output". The "sequence_output" key contains the hidden state outputs for each token in the input sequence, while the "pooled_output" key contains a fixed-size representation of the entire input sequence.

Then the "pooled_output" representation is used as the input to the classifier layer. Which is a dense layer with a softmax activation function and it represent the output layer for our model. The output of this layer is a probability values between 0 and 1, indicating the likelihood of the input text belonging to a specific classes.

3.5.2 Model layers

To fully understand the model architecture, it is necessary to traverse all layers and understand the significance of each layer. In our architecture we have the following layers

3.5.2.1 Input layer

The first layer in this model is the Input layer. This layer defines the shape and data type of the input to the model. In this case, the input to the model is text data, represented as a string data type.

The 'shape' parameter in the Input layer specifies the shape of the input data. Since the text input can vary in length, we set the shape to scalar.

In neural networks, each layer usually has a fixed number of neurons, but in the case of the input layer, it can handle variable-length inputs.

The 'dtype' parameter specifies the data type of the input data. In this case, we set it to tf.string to indicate that the input is a string type.

```
text_inputs = [tf.keras.layers.Input(shape=(), dtype=tf.string)]
```

Pseudo-code 17.Implementation of the input layer

3.5.2.2 BERT preprocessor tokenizer

In this layer we implemented the BERT preprocessor tokenizer which will handle the tokenization phase in our model as explained in the tokenization phase above.

```
preprocessor = hub.load("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
Tokenization_layer=hub.KerasLayer(preprocessor.tokenize, name="BERT_preprocessor_Tokenizer")
```

Pseudo-code 18.Implementation of the BERT preprocessor tokenizer

3.5.2.3 BERT Preprocessor layer

The BERT preprocessing layer is an essential component of the BERT model. It is responsible for input tokens into the format and shape expected by the BERT encoder. This transformation is necessary because the BERT encoder has a predefined input shape.

The BERT preprocessor layer is created using the bert_pack_inputs function from the preprocessor model. This layer takes the tokenized inputs and packs them into the required format for input to the BERT model.

The packed inputs include the following components:

1. Input word IDs: Each token in the tokenized input is assigned a unique integer ID that represents its index in the BERT model's vocabulary. The BERT preprocessor layer ensures that the tokenized inputs are converted into the corresponding input IDs.
2. Attention Mask: The attention mask is a binary mask that indicates which tokens should be attended to (1) and which tokens should be ignored (0) during the self-

attention mechanism of the BERT model. It helps the model differentiate between actual tokens and any padding tokens added to achieve the fixed sequence length. The layer generates the attention mask for the packed inputs.

3. Token Type IDs: In certain tasks, such as question-answering or sentence pair classification, BERT models require additional information about the segments of the input. Token type IDs are used to indicate different segments within the input sequence. For example, in a sentence pair classification task, the first segment might represent the premise, while the second segment represents the hypothesis. This layer handles the creation of token type IDs based on the tokenized inputs.

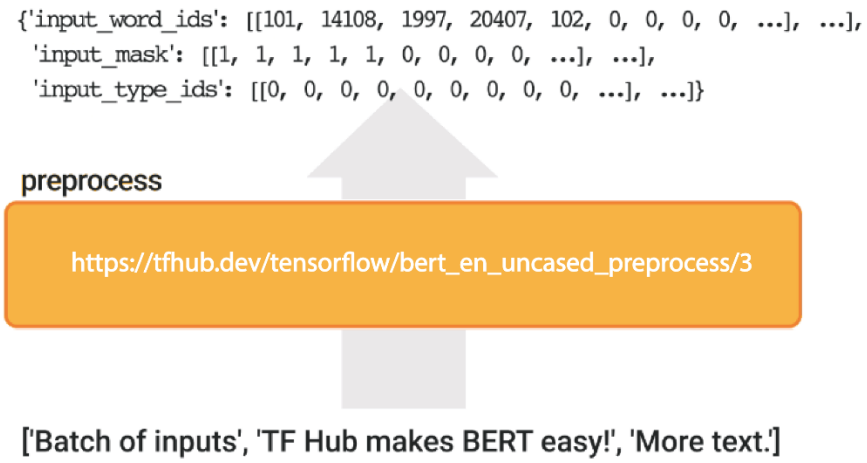

Figure 16.BERT preprocessing model functionality [32]

To achieve high accuracy results we had to test and observe many techniques and concepts and we eventually achieved significantly improved results by using the specified sequence length in the BERT preprocessing model, which allows us to reduce the number of supplemented words that the preprocessing model add as a padding to the original input in purpose of making the outputs in the same format to the BERT encoder layer.

With this technique we specified the sequence length to the maximum number of words per tweet, in our dataset the maximum words are 60 word per tweet

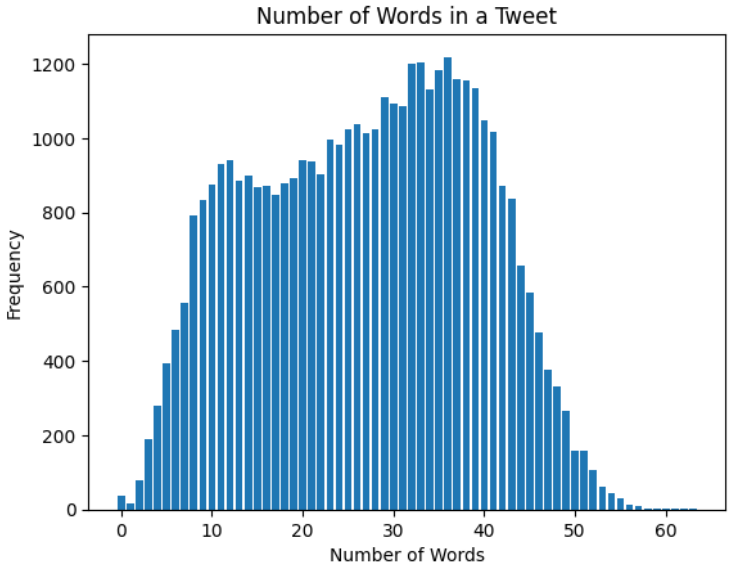

Figure 17.Representation of the numbers of words per tweet

in this way we have reduced the padding words to the minimum as possible for this dataset, which will give the original words (the tweet words) almost the full dependance in the calculations that make the final classification result without any paddings that will affect the meaning or the value of the pooled output of the BERT encoder model for the tweet.

```
seq_length = 60
bert_pack_inputs = hub.KerasLayer(
  preprocessor.bert_pack_inputs,
  arguments=dict(seq_length=seq_length), name="BERT_preprocessor",)
bert_encoder_inputs = bert_pack_inputs(tokenized_inputs)
```

Pseudo-code 19.Implementation of the BERT preprocessor layer

3.5.2.4 BERT encoder layer

In this layer we imported the BERT base model, a deep pre-trained neural network employing the Transformer architecture which offers dense vector representations.

In our case we imported The BERT base model that consists of 12 transformer layers, 12 attention heads, a hidden size of 768, and a total of approximately 110 million parameters.

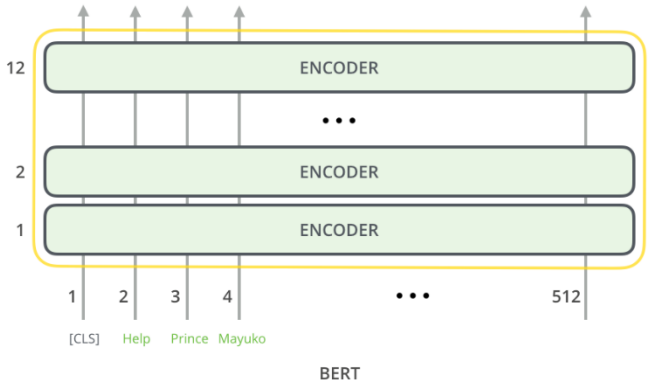

Figure 18.Representation of the BERT base model [29]

And here is the implementation of the Bert Model as trainable to fine-tune the model to our purpose

```
bert_encoder_layer = hub.KerasLayer(BERT_Model_url,   trainable=True,   name="BERT_encoder")
```

Pseudo-code 20.Implementation of the BERT Base layer

And the output of this layer will be a python dictionary of keys :

- Pooled output: refers to a fixed-length representation of the entire input sequence. It is obtained by applying a pooling operation over the sequence-level representations produced by the final layer of BERT.
- Encoder outputs: refers to the token-level representations generated by each layer of the BERT model. These representations capture contextual information for each token in the input sequence.
- sequence output: refers to the token-level representations or contextual embeddings of each token in the input sequence. It represents the output of the BERT model at the token level after processing the input sequence

3.5.2.5 Output layer

This layer is a dense layer named classifier with 3 output neuron that uses SoftMax function as activation function.

The SoftMax activation function is a non-linear function that outputs a value between 0 and 1. This is useful for multi-class classification tasks, as it allows the model to output a probability for each class.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ = softmax

$\vec{z}$ = input vector

e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

e^{z_j} = standard exponential function for output vector

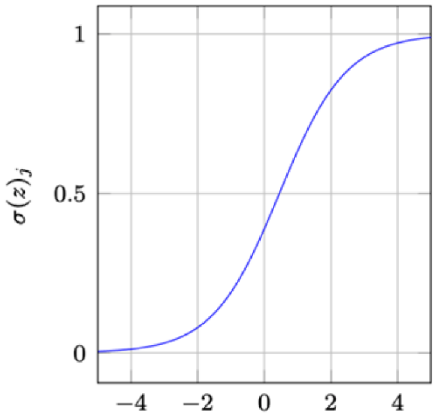

Figure 19.Representation of the SoftMax activation function

The dense layer receives the pooled output of the BERT encoder. This allows the model to learn the relationship between the long-term dependencies in the input text and the output class.

The dense layer is a key part of the model architecture. It is responsible for making the final prediction about the class of the input text.

```
output_layer=keras.layers.Dense(3,   activation='softmax',   name="classifier")
```

Pseudo-code 21.Implementation of the output layer

Conclusion

By the end of this chapter the reader should have the full understanding of the model structure and the context that leads us to adopt this structure to our task of COVID-19 tweet analysis. In the next chapter we will get more to how this model preform and we will discover how our decisions to adopt some techniques to improve our model has affected the model performance to achieve high results.

CHAPTER 4

Experiments and Results

Introduction

This chapter delves into the practical aspects of our deep learning model for sentiment analysis. This chapter presents a comprehensive analysis of the experimental setup, hyperparameters, model training process, obtained results, and a comparative study with the existing literature. By conducting these experiments, we aim to evaluate the effectiveness and performance of our model in accurately classifying covid tweets according to their sentiment.

4.1 Experiment Environment

We used various software and hardware environments to develop and evaluate our deep learning model for sentiment analysis of related tweets. Key platforms and libraries include Google Colab, TensorFlow Hub, TensorFlow, Keras, and Kaggle.

Figure 20.Environments

Google Colab, a cloud-based Jupyter notebook environment, provided us with the flexibility to run experiments seamlessly on powerful hardware, such as GPUs and TPUs. It eliminated the need for extensive local computing resources and integrated well with Google Drive for data storage and collaboration.

TensorFlow Hub, a repository of pre-trained models and embeddings, played a vital role in our experiment environment. We leveraged the vast collection of pre-trained models to speed up model development using transfer learning, improving tweet classification performance.

TensorFlow, an open-source deep learning framework, and Keras, a high-level neural networks API, formed the foundation of our model development. TensorFlow provided

essential tools for building and training deep learning models, while Keras offered an intuitive interface for constructing and configuring neural networks.

We also utilized Kaggle, a popular data science platform, to access relevant datasets and collaborate with the data science community. Kaggle provided diverse datasets, including our COVID-labeled tweet dataset, enabling comprehensive experiments and evaluating the model's effectiveness on real-world data.

By leveraging these cutting-edge tools and platforms, our goal was to create a productive and reliable experiment environment that ensures consistent and reproducible results.

4.2 Hyperparameters

This section delves into the critical choices and configurations that define the behavior and performance of our deep learning model for classifying sentiments in COVID-19 related tweets. Hyperparameters play a crucial role in shaping the learning process, influencing the model's ability to generalize and make accurate predictions. In this section, we discuss the key hyperparameters we considered and their impact on the model's performance.

Selecting appropriate hyperparameters involves a delicate balance between bias and variance, regularization, optimization algorithms, and the loss function. These choices heavily influence the model's ability to learn meaningful patterns from the data and avoid overfitting or underfitting.

One of the fundamental hyperparameters we focused on is the optimizer. We employed the widely-used Adam optimizer with a learning rate of 2e-5. The Adam optimizer adapts the learning rate dynamically, allowing the model to converge efficiently and find an optimal solution. By carefully setting the learning rate, we aimed to strike a balance between convergence speed and model accuracy.

Furthermore, the choice of loss function is crucial in defining the objective of the model during training. We utilized the CategoricalCrossentropy loss function, which is suitable for multi-class classification tasks. This loss function measures the discrepancy between the predicted class probabilities and the true labels, enabling the model to learn from the data and improve its predictions iteratively.

```
# Compiling the model with the hyperparameters
classifier_model.compile(optimizer=keras.optimizers.Adam(2e-5),
              loss='CategoricalCrossentropy',
              metrics=['accuracy','Recall','Precision'])
```

Pseudo-code 22.The compiling method and hyperparameters

4.3 Model Training

In this section, we discuss the key aspects of the training process, including the dataset partitioning, batch size, and the number of epochs. This phase plays a pivotal role in enabling the model to learn from the data and optimize its performance.

To ensure an effective evaluation of our model's performance, we partitioned our dataset using a split of 0.2 for test and 0.8 for training. This approach allowed us to create a separate subset of data that was not involved in the training process but was used to assess the model's generalization ability.

Additionally, we carefully considered the batch size, which determines the number of samples processed in each training iteration. We opted for a batch size of 50, which strikes a balance between computational efficiency and the model's ability to learn meaningful patterns from the data. A larger batch size can provide better gradient estimates but might require more memory, while a smaller batch size can lead to noisy updates but may converge faster. By selecting an appropriate batch size, we aimed to optimize the training process and enhance the model's learning dynamics.

Furthermore, we conducted our model training for a total of 3 epochs. An epoch represents a complete pass through the entire training dataset. By training for multiple epochs, the model can progressively refine its internal representations and improve its predictive capabilities. The choice of the number of epochs involves a trade-off between computational resources and convergence to an optimal solution. With 3 epochs, we aimed to strike a balance between training the model sufficiently and avoiding overfitting to the training data.

The figure below describe the model performance evolution during the training time

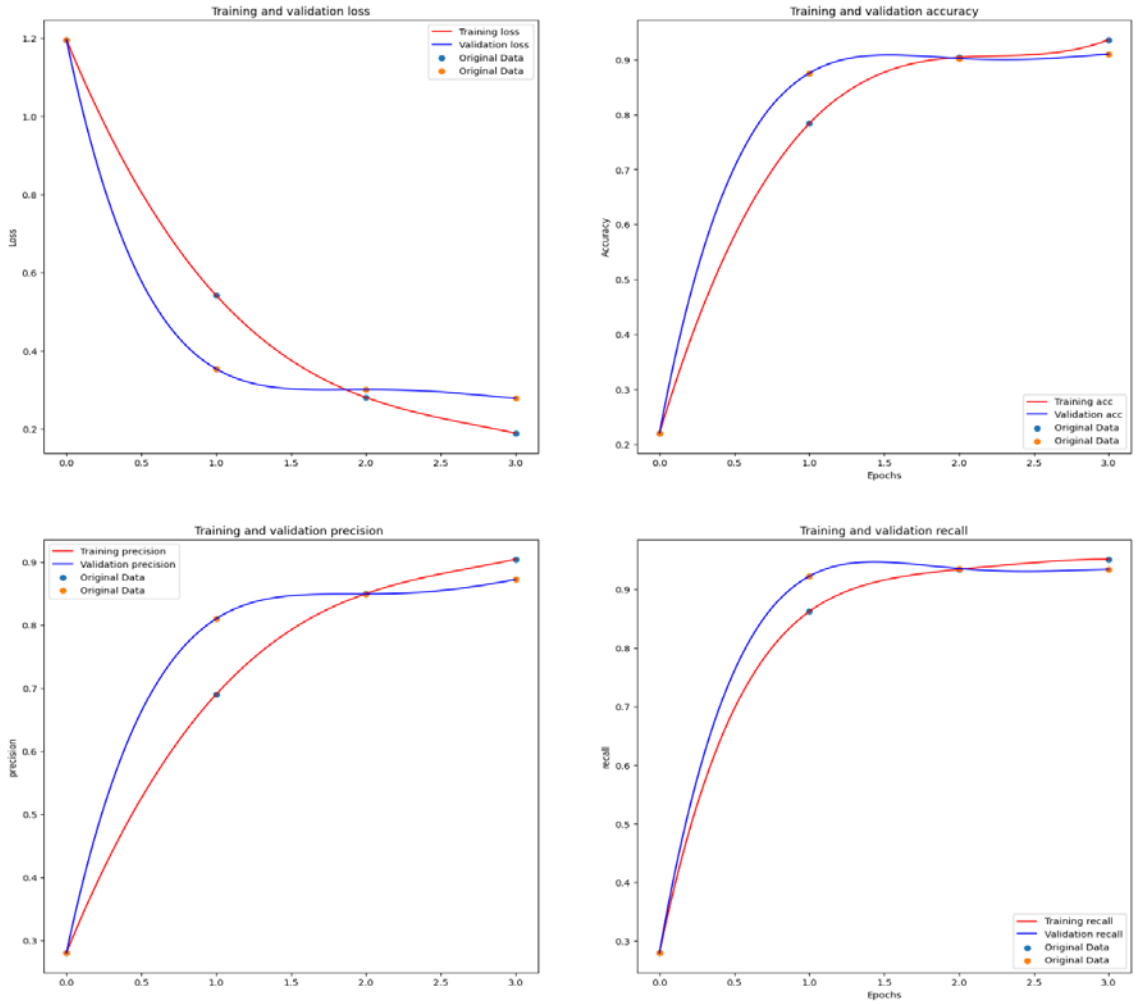

Figure 21.Plotted measures during the training

Finally, under these specifications the model requires a time of approximate 30 minutes to train completely.

4.4 Results

In this section we present a comprehensive analysis of the performance and effectiveness of our deep learning model for tweet classification. This section serves as a critical evaluation of the model's ability to accurately classify covid tweet sentiments into relevant categories, considering the three classes: positive, negative and neutral.

We begin by providing an overview of the evaluation metrics used to assess the model's performance. These metrics include commonly employed measures such as accuracy, precision, recall, and F1 score.

Accuracy is a metric commonly used to assess the overall performance of a model across all classes. It is particularly valuable when all classes have equal importance. The accuracy score is calculated by dividing the number of correct predictions by the total

number of predictions made by the model. [33]

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

Figure 22.Accuracy metric equation [34]

Precision evaluates the correctness of positive predictions compared to the total number of positive predictions made.

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

Figure 23.Precision metric equation [34]

Recall, also known as sensitivity or true positive rate, assesses the ratio of accurately predicted positive instances to the total number of actual positive instances.

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

Figure 24.Recall metric equation [34]

The F1 score is a balanced measure of a model's performance, calculated as the harmonic mean of precision and recall. It takes into account both precision and recall to provide an overall assessment of the model's effectiveness.

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}}$$

TP = number of true positives
FP = number of false positives
FN = number of false negatives

Figure 25.F1-score metric equation [35]

Metric	Accuracy	Precision	Recall	Avg F1-Score
Score	91.4 %	92.02 %	90.39 %	89 %

Table 1.Model scores

Label	Precision	Recall	F1- score
Positive	89 %	91 %	90 %
Neutral	92 %	80 %	86 %
Negative	88 %	91 %	90 %

Table 2.Model based labels scores

Additionally, we present a detailed analysis of the confusion matrix, which provides

valuable insights into the model's classification accuracy across different classes. The confusion matrix is a tabular representation that showcases the number of instances classified correctly and incorrectly for each class. By examining the confusion matrix, we can identify any patterns or tendencies in the model's classification errors and gain a deeper understanding of its strengths and limitations.

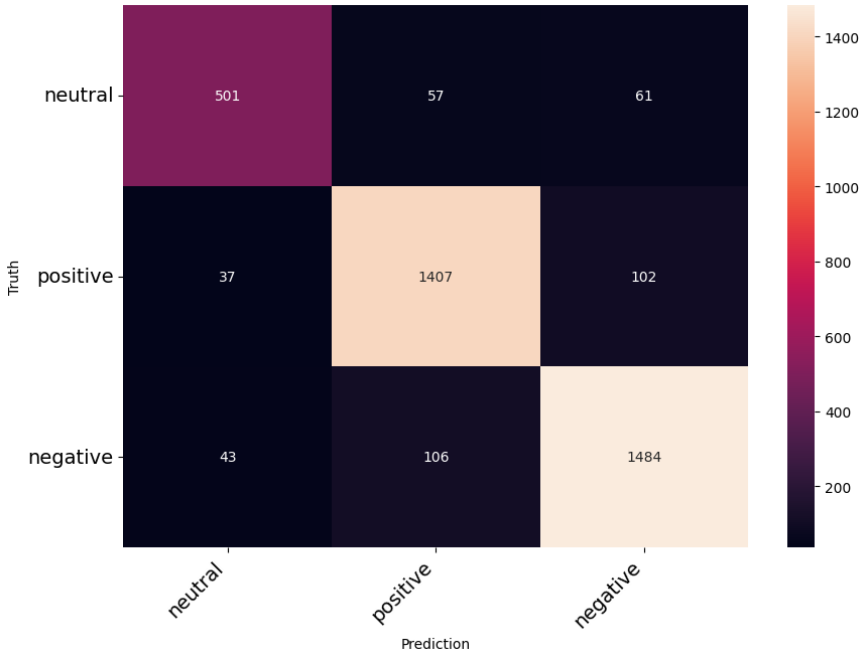

Figure 26.Confusion matrix

Based on the confusion matrix we can recognize that there is no classification bias problem that could happen because of the imbalance labels on the dataset so we conclude that this model needs no further operation needed to rebalance the labels such as data augmentation.

4.5 Comparison

Here we conduct a thorough comparison between our deep learning model's results and those achieved by other researchers in the field of tweet classification. This analysis allows us to gain a broader perspective on the effectiveness and performance of our model in relation to existing approaches.

We compared the results obtained from Our proposed model based on BERT using the following metrics (Accuracy, Precision, Recall, f1-score) with some deep learning and machine learning models as follows :

- LSTM-model: which is based on LSTM algoriothms in deep learning. [36]
- MKH-SVM model: MKH-SVM is an acronym for Khaldoun M. K. H. Halawani Support Vector Machine, a variant of the popular support vector machine (SVM)

algorithm created by Khaldoun M. K. H. Halawani. SVM is a widely used machine learning algorithm employed for classification and regression tasks. [37]

- Neural Net model: model based RNN algorithm. [37]
- Linear-SVM: machine learning model variant from SVM algorithm that uses a lineaer Kernal function to separate classes. [37]

Each of these models was applied to the test dataset. All the results obtained are summarized in the followiong charts:

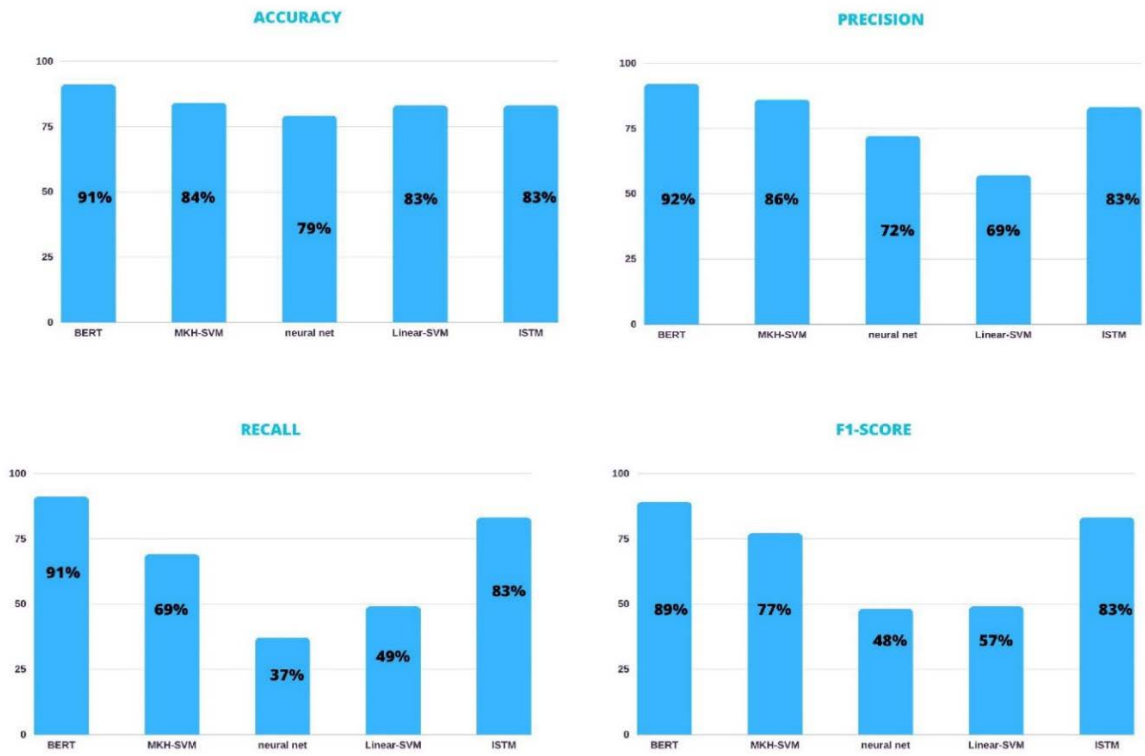

Figure 27.Bar chart comparison between different models

Model	Accuracy	Precision	recall	F1-score
BERT	91.4%	92.02%	90.39%	89%
LSTM [36]	83%	83%	83%	83%
MKH-SVM [37]	84%	86%	69%	77%
Neural Net [37]	79%	72%	37%	48%
Linear- SVM [37]	83%	69%	49%	57%

Table 3.Comparison between different models metrics

4.5.1 Discussion

As can be seen in table 3, our model gives the best results in all metrics, surpassing both deep learning model (LSTM) and machine learning algorithms. These results prove that BERT outperforms other techniques.

4.6 Mobile Application

In purpose of presenting our model we have found tha it will be effective to build a mobile application to enhance the accessibility of the model and provide a user friendly interface that aligns with the latest trends in mobile applications .

This Flutter application provides a simple user interface with 3 fields to discover beside the home page.

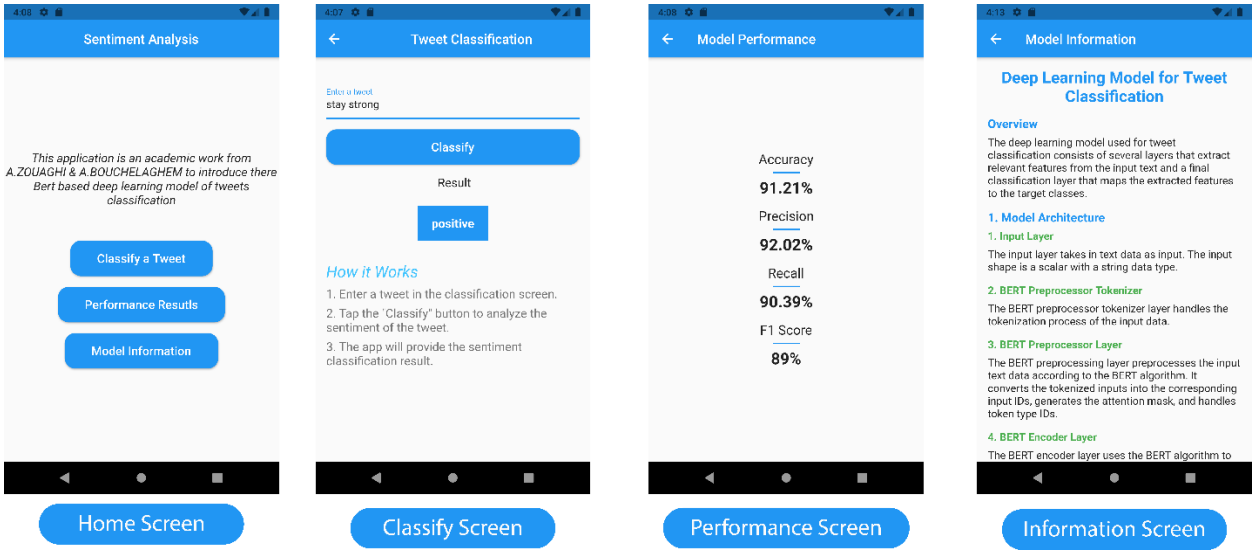

Figure 28.Mobile application interface

- Classify screen: contain text input field for entering a tweet. When the "Classify" button is pressed, the '_classify Sentiment' method is called, which makes an HTTP POST request to the flask server endpoint. The response is displayed as it received under result on the screen.
- Performance screen: Shows the performance and the model results as described above
- Information screen: In this screen the user can go more in depth and enjoy the simple explanation of this model and how it was built without going into the details of this master project which make the user have a strong idea and advanced understanding of the overall work that has been done on this project

In the backend side the flask server will handle the classification requests. The server

accepts the input text, preprocess it, and pass it through our model for sentiment analysis. then return the predicted sentiment (positive, negative, or neutral) as the response to the Flutter application.

The Flask server set with a '/classify' route that accepts POST requests containing the tweet text. It retrieves the text from the request payload, preprocesses it if necessary, passes it through our trained model and returns the predicted sentiment as a JSON response.

Conclusion

In this chapter, we conducted experiments and presented the results of our deep learning model for tweet classification. We discussed the experiment environment, hyperparameters and model training, beside representing the evaluation results using various metrics. We compared our model with existing approaches, finally we introduced the mobile application that built to enhance the accessibility for the model. These findings demonstrate the effectiveness of our model and provide valuable insights for the field of sentiment analysis and tweet classification.

General Conclusion

The objective of the study was to evaluate the effectiveness of the BERT model for conducting sentiment analysis on Twitter COVID-19 related data.

The study used a BERT-Base architecture followed by a final classification step and refined the model for the task of sentiment analysis. The BERT model is a pre-trained model for Natural Language Processing that can perform multiple tasks such as question answering systems, text classification, and sentiment analysis. BERT has successfully achieved state-of-the-art accuracy on 11 common NLP tasks, outperforming previous top NLP models, and is the first to outperform humans.

The BERT model has been used in various studies for sentiment analysis on Twitter posts that address the topic of COVID-19.

The performance of the classifiers was measured by considering two sets of tweets data, and the results showed remarkable accuracy of over 93.92% in the train phase and 91.4% in the test phase.

The study concluded that the power of the BERT model contributes significantly to a good classification of tweets. It's a breakthrough in the use of Machine Learning for Natural Language Processing, and it is approachable and allows fast fine-tuning, which will likely allow a wide range of practical applications in the future.

As futur work, we will :

- Focus on improving our model's efficiency, reducing training time, and addressing issues such as overfitting or the sensitivity to hyperparameters.
- Focus on model compression techniques to reduce the size and memory requirements of BERT while maintaining its performance.
- Testing our model on other datasets and increasing the number of data.
- Testing other transformers like:GPT3,T5,BART……

Bibliography

[1] Schroer, Alyssa. “What Is Artificial Intelligence? How Does AI Work? | Built In.” *Builtin*, 3 Mar. 2023, builtin.com/artificial-intelligence.

[2] IBM. “What Is Natural Language Processing? | IBM.” *Www.ibm.com*, 2023, www.ibm.com/topics/natural-language-processing.

[3] Foote, Keith D. “A Brief History of Natural Language Processing (NLP).” *DATAVERSITY*, 22 May 2019, www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/#.

[4] Merriam-Webster, Collegiate Dictionary, 2019.

[5] “What Is Sentiment Analysis? - Sentiment Analysis Guide | Simplilearn.” *Simplilearn.com*, 7 Mar. 2023, www.simplilearn.com/what-is-sentiment-analysis-article?tag=sentiment%20analysis. Accessed 19 June 2023.

[6] B. a. D. Y. a. L. X. a. L. W. S. a. Y. P. S. Liu, "Building text classifiers using positive and unlabeled examples," in *Third IEEE international conference on data mining*, 2003, pp. 179--186.

[7] G. a. H. S. C. a. C. K. a. J. R. Li, "Micro-blogging sentiment detection by collaborative online learning," in *2010 IEEE International Conference on Data Mining*, 2010, pp. 893--898.

[8] B. a. L. L. a. V. S. Pang, "Thumbs up? Sentiment classification using machine learning techniques," *arXiv preprint cs/0205070,* 2002.

[9] L.-C. a. L. C.-M. a. C. M.-Y. Chen, Exploration of social media for sentiment analysis using deep learning, Springer, 2020, pp. 8187--8197.

[10] W. a. S. J. a. Z. L. a. C. C. a. Y. Q. Feng, "A support vector machine based naive Bayes algorithm for spam filtering," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, 2016, pp. 1--8.

[11] H. Shirani-Mehr, "Applications of deep learning to sentiment analysis of movie reviews," in *Technical report*, Stanford University, 2014.

[12] "Sentiment analysis on movie reviews using recursive and recurrent neural network architectures}," *Semantic Scholar,* pp. 1--5, 2015.

[13] L. a. C. C. Zhang, "Sentiment classification with convolutional neural networks: An experimental study on a large-scale chinese conversation corpus," in *2016 12th International Conference on Computational Intelligence and Security (CIS)*, 2016, pp. 165--169.

[14] P. a. R. R. a. K. P. Mishra, "Sentiment analysis of Twitter data: Case study on digital India," in *2016 International Conference on Information Technology (InCITe)-The Next Generation IT Summit on the Theme-Internet of Things: Connect your Worlds*, 2016, pp. 148--153.

[15] A. L. a. C. D. E. a. C. A. L. a. C. D. E. Caterini, Recurrent neural networks, Springer, 2018.

[16] F. a. B. S. Miedema, "Sentiment analysis with long short-term memory networks," *Vrije Universiteit Amsterdam,* vol. 1, pp. 1--17, 2018.

[17] X. a. Z. P. a. L. C. H. a. L. L. Ouyang, "Sentiment analysis using convolutional neural network," in *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*, 2015.

[18] J. a. B. J. a. R. T. Panthati, "Sentiment analysis on customer reviews using deep learning," *Int. J. Comput. Sci. Eng,* 2018.

[19] A. a. S. N. a. P. N. a. U. J. a. J. L. a. G. A. N. a. K. {. a. P. I. Vaswani, "Attention is all you need," *Advances in neural information processing systems,* 2017.

[20] J. a. O. Y. D.-L. a. P.-C. I. a. R.-M. K. a. S.-T. J. P. M. a. G. O. J. Alcibar-Zubillaga, "Participation of escom's data science group at rest-mex 2022: Sentiment analysis task," in *Proceedings of the Third Workshop for Iberian Languages Evaluation Forum (IberLEF 2022), CEUR WS Proceedings*, 2022.

[21] M. {. a. D.-P. {. a. A. R. a. R.-G. A. Y. a. F.-D. D. a. G.-R. R. a. B.-M. L. \'A}lvarez-Carmona, "Overview of rest-mex at iberlef 2022: Recommendation system, sentiment analysis and covid semaphore prediction for mexican tourist texts," *Procesamiento del Lenguaje Natural,* pp. 289--299, 2022.

[22] S. U. A. A. V. C. Harleen Kaur1, "A Proposed Sentiment Analysis Deep Learning Algorithm," *Springer Science+Business Media,* 2021.

[23] McCullum, Nick. "Deep Learning Neural Networks Explained in Plain English." *FreeCodeCamp.org*, 28 June 2020, www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/.

[24] Wikipedia Contributors. "Deep Learning." *Wikipedia*, Wikimedia Foundation, 10 May 2019, en.wikipedia.org/wiki/Deep_learning.

[25] Olah, Christopher. "Understanding LSTM Networks -- Colah's Blog." *Github.io*, 2015, colah.github.io/posts/2015-08-Understanding-LSTMs/.

[26] A. S. N. P. N. U. J. J. L. G. A. K. L. &. P. I. Vaswani, "Attention Is All You Need," in *NIPS*, 2017.

[27] Alammar, Jay. "Jekyll Now." *GitHub*, 17 June 2023, github.com/jalammar/jalammar.github.io/blob/master/_posts/2018-06-27-illustrated-transformer.md. Accessed 19 June 2023.

[28] Devlin, Jacob, et al. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." *Google Research*, 2018, research.google/pubs/pub47751/.

[29] Alammar, Jay. "The Illustrated BERT, ELMo, and Co. (How NLP Cracked Transfer Learning)." *Github.io*, 2018, jalammar.github.io/illustrated-bert/.

[30] A. MIGLANI, "Coronavirus tweets NLP - Text Classification," [Online]. Available: https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification.

[31] Sharma, Abhishek. "A Beginner's Guide to Exploratory Data Analysis (EDA) on Text Data (Amazon Case Study)." *Analytics Vidhya*, 26 Apr. 2020, www.analyticsvidhya.com/blog/2020/04/beginners-guide-exploratory-data-analysis-text-data/. Accessed 19 June 2023.

[32] *Making BERT Easier with Preprocessing Models from TensorFlow Hub*. blog.tensorflow.org/2020/12/making-bert-easier-with-preprocessing-models-from-tensorflow-hub.html?_gl=1. Accessed 19 June 2023.

[33] Gad, Ahmed Fawzy. "Accuracy, Precision, and Recall in Deep Learning." *Paperspace Blog*, 12 Oct. 2020, blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/.

[34] "Accuracy, Precision, and Recall in Deep Learning." *Paperspace Blog*, 12 Oct. 2020, blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy. Accessed 19 June 2023, https://www.facebook.com/ahmed.f.gadd.

[35] "F-Score." *Wikipedia*, 12 Mar. 2021, en.wikipedia.org/wiki/F-score.

[36] Vernikou, Sotiria, et al. "Multiclass Sentiment Analysis on COVID-19-Related Tweets Using Deep Learning Models." *Neural Computing and Applications*, 6 Aug. 2022, https://doi.org/10.1007/s00521-022-07650-2. Accessed 7 Aug. 2022.

[37] H. K. &. S. U. A. &. B. A. &. V. Chang2, "A Proposed Sentiment Analysis Deep Learning Algorithm," 2021.

Acronyms

COVID-19	Coronavirus Disease 2019
NLP	Natural language processing
RNN	Recurrent Neural Network
GPT	Generative Pre-trained Transformer
AI	Artificial Intelligence
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
CNN	Convolutional Neural Network
VADER	Valence Aware Dictionary and sEntiment Reasoner
IMDB	Internet Movie Database
BERT	Bidirectional Encoder Representations from Transformers
ANN	Artificial Neural Network
T5	Text-to-Text Transfer Transformer
BART	Bidirectional and Auto-Regressive Transformer