INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DOTS - Detection of Covid-19 Contagion Symptoms and Self-Diagnosis in Social Networks

Clístenes Fernandes da Silva

Work supervised by:

Prof. Dr. Rui Pedro Lopes

Prof. Dr. Arnaldo Candido Junior

Master in Informatics

2020-2021

DOTS - Detection of Covid-19 Contagion Symptoms and Self-Diagnosis in Social Networks

Dissertation presented to the School of Technology and Management to obtain the Master's Degree in Informatics

Clístenes Fernandes da Silva

2020-2021

The School of Technology and Management is not responsible for the opinions expressed in this document

Acknowledgments

First of all, I thank God for everything. Also both Polithecnic Institute of Bragança - IPB and Federal University of Technology Parana - UTFPR for giving me the opportunity to participate in this amazing program.

I thank all the help, patience and availability that my mentors Professor Dr. Rui Pedro Lopes and Professor Dr. Arnaldo Candido Junior had with me before and during this project.

Special thanks to my family, specially my wife Anne K. Fernandes and my parents Siegfried and Araly Riegler that were always there for me even though the distance and gave all the necessary support to conclude my projects.

Abstract

Social media present ways for people to share emotions, feelings, ideas, and even symptoms of disease, and is a great source of data for a variety of analyses. At the end of 2019, an alert was raised for a global pandemic of a virus that has a very high contamination rate and can cause respiratory complications in the contaminated people. To help identify those who may have the symptoms of this disease or to control who are already infected, this paper analyzed the performance of KNN, Naive Bayes, Decision Tree, Random Forest, SVM, simple Multilayer Perceptron, Convolutional Neural Networks and BERT algorithms to classify tweets that contained reports of Covid-19 symptoms or self-reports of infection. The dataset was labeled using a set of disease symptom keywords taken from a list provided by the World Health Organization. The tests on these models showed that the Random Forest algorithm performed best when classifying the tweets in a small dataset. This work demonstrated a superior performance of the Random Forest algorithm over other more robust algorithms for this type of classification and dataset.

Keywords: Machine Learning, Deep Learning, Covid-19, Algorithm Comparison.

Resumo

As redes sociais apresentam meios para as pessoas compartilharem emoções, sentimentos, ideias e até sintomas de doenças, e são uma ótima fonte de dados para as mais diversas análises. No final do ano de 2019, um alerta foi levantado para uma pandemia global de um vírus que tem uma taxa de contaminação muito elevada e que pode causar complicações respiratórias nas pessoas contaminadas. Para o auxilio na identificação de pessoas que possam ter os sintomas desssa doença ou o controle das que já estão infectadas, neste trabalho foram analisados os desempenhos dos algoritmos KNN, Naive Bayes, Decision Tree, Random Forest, SVM, Multilayer Perceptron simples, Redes neurais Convolucionais e BERT para classificação de tweets que continham relatos de sintomas do Covid-19 ou auto-declaração de contaminação. O conjunto de dados foi rotulado utilizando um conjunto de palavras chaves dos sintomas da doença retirada de uma lista disponibilizada pela Organização Mundial da Saúde. Os testes nesses modelos mostraram que o algoritmo Random Forest foi o que obteve melhor resultado ao classificar os tweets em uma base de dados pequena. Este trabalho demonstrou o desempenho superior do algoritmo RandomForest sobre outros mais robustos para este tipo de classificação e conjunto de dados.

Palavras-chave: Machine Learning, Deep Learning, Covid-19, Comparação de Algoritmos.

Contents

List of Tables

List of Figures

Siglas

AI Artificial Intelligence. 3, 5, 10–12

ANN Artificial Neural Networks. 5, 19, 20, 26

BERT Bidirectional Encoder Representations from Transformers. 3, 5, 31, 32, 42, 44, 49–52

CNN Convolutional Neural Networks. xiii, 3, 23, 26, 42, 44, 49–53

DNA Deoxyribonucleic Acid. 6, 7, 10

GPU Graphics Processing Unit. 11

KNN K-Nearest Neighbors. 3, 42, 48–51

ML Machine Learning. 3, 5, 12, 13, 36

MLP Multilayer Perceptron. 21, 22, 42, 49, 50

NLP Natural Language Processing. 2, 3, 11, 12

OSH Optimal Separating Hyperplan. 18

ReLU Rectified Linear Unit. xiii, 23, 24, 42

RNA Ribonucleic Acid. 6, 7, 10

SVM Support Vector Machine. 3, 17, 18, 42, 49–52

SVR Support Vector Regression. 18

WHO World Health Organization. 7, 10, 38

Chapter 1

Introduction

In this chapter we will present an introduction to the work theme. First, we will approach the background in which this work is inserted. Next, we will show the objectives that we expect to achieve and then, the structure that this document follows.

1.1 Background

The year 2020 posed organizations, governments, and even the population a great challenge in the face of a virus that was spreading at a very high speed and that, in some cases, required more accurate information. The delay in obtaining concrete information and mass testing were some of the challenges faced in the effort to control the spread of the virus in the population [1].

Over time, new information about the behavior of the virus and the mapping of its transmissibility has been obtained from more traditional sources of data collection, such as population health surveys or group studies, where investigations of causes, and relationships between risk factors and health consequences are made. However, these approaches can take time to generate relevant reports, and this can be very damaging when taking into account the characteristics of a pandemic [2].

Since the advent of social networks, they have been widely used as a way for people to express emotions, feelings, opinions, information, as well as health concerns and symptoms, making these communication media potential sources for collecting and building a dataset of self-reported symptoms [3].

A possible source where data can be collected for analysis and possible detection of disease symptoms is Twitter, a virtual platform for social interaction and microblog, where users can send and receive updates from other contacts, limiting themselves to a maximum, at the date of this work, of 280 characters per message. To analyze the texts posted on twitter and classify those with self-declared symptom content, Natural Language Processing (NLP) techniques can be applied. NLP is a research and application area that explores how computers can be used to understand and manipulate natural language from text or speech and, although it is not a recent area of study, it has been gaining more and more space in several fields [4]. It is possible to see the growth of NLP in everyday life, being used by a lot of common people, for instance, when using a cell phone or computer to interact with virtual assistants such as Alexa[1], Cortana[2], Siri[3] or Google Assistant[4].

In companies, for example, NLP can be used to evaluate consumer acceptance of a certain product, which can make the company more effective in producing the next products or improving the current ones. NLP can also be trained with the goal of extracting relevant and, in some cases, unknown information about characteristics or symptoms of some disease, such as Covid-19.

1.2 Objectives

The objective of this work is to study and apply machine learning techniques and algorithms, based on natural language processing, to identify symptoms and text with self-diagnostic content in social networks. For this purpose, Tweeter was used as a base

[1] https://alexa.amazon.com/

[2] https://support.microsoft.com/en-us/topic/what-is-cortana-953e648d-5668-e017-1341-7f26f7d0f825

[3] https://www.apple.com/br/siri/

[4] https://assistant.google.com/

for collecting and studying texts with Covid-19 related content.

In order to achieve the general objective of this work, the following specifics objectives were made:

- Collect and build a dataset from tweets;
- Label the dataset with positive or negative classes, according to the tweet's characteristic;
- Research and apply Machine Learning algorithms such as Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN);
- Research and apply Artificial Intelligence algorithms such as Multilayer Perceptron, CNN, Bidirectional Encoder Representations from Transformers (BERT);
- Analyze and compare the performance of the algorithms.

1.3 Document Structure

This document is composed of five Chapters. As we are currently in the Introduction, the rest of the document is organized as follows:

- In the Second Chapter, we will talk about the theoretical basis involving the Coronavirus, concepts of Artificial Intelligence (AI), Machine Learning (ML) and NLP.
- In the Third Chapter, we will show the materials and methods used to implement this work.
- In the Fourth Chapter, we will post the results obtained after testing the models and discuss about them.
- In the Fifth Chapter, we will present our conclusions about the work and suggest future directions around the theme of this project.

Chapter 2

Context and Technologies

This chapter will cover the concepts regarding the Coronavirus on Section 2.1, Artificial Intelligence on Section 2.2, Machine Learning on Section 2.4, Artificial Neural Networks on Section 2.5, Transformers on Section ?? and BERT on Section 2.8.

2.1 Coronavirus

Covid-19 (Corona Virus Disease-2019) is a contagious disease recently discovered as result of the infection from a new variation of an existing group of viruses that includes SARS-CoV (Severe Respiratory Acute Syndrome). This disease is caused by the virus SARS-CoV-2.

Two other events related to this virus occurred in the years of 2002 and 2012, with SARS-CoV in Asia and MERS-CoV (Middle East Respiratory Syndrome Coronavirus) in the Middle East, both leading to the death of a large number of people.

2.1.1 Virus

Viruses are very small parasites, usually ranging from 0.02 to 0.3 micrometers in size. Viruses depend on a host to reproduce, such as bacteria, plants, or animals [5]. Viruses usually have a very simple structure: they are covered by proteins, called capsids, and in some cases also by lipids. The nucleus may contain Deoxyribonucleic Acid (DNA), Ribonucleic Acid (RNA), or even both. Each type of virus can contain single or double-stranded genetic material. Single-stranded RNA viruses can have either positive-sense or negative-sense polarity [6]. A positive-sense RNA can have its genome immediately translated by the host cell while the negative-sense needs to be converted to positive-sense before being translated. Figure 2.1 shows a representation of a double-stranded RNA and its translation into viral protein.

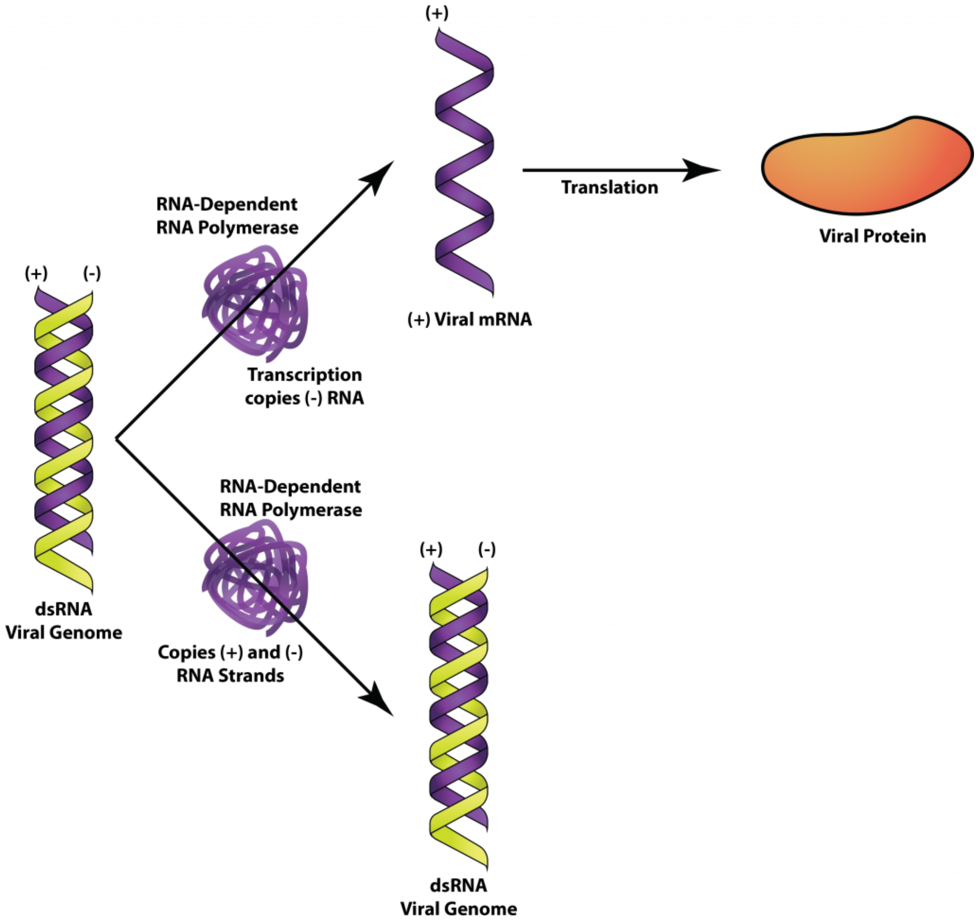


Figure 2.1: Double-stranded (dsRNA) RNA [7]

The proteins present in the capsid are particular to each type of virus. Some viruses may consist of just the nucleocapsid, which is RNA or DNA surrounded by the capsid, while others may have a wrapping or envelope on the surface of the nucleocapsid, called encapsulated or enveloped viruses [5].

It is common for DNA viruses to replicate in the nucleus of the host cells, while RNA viruses usually replicate in the cytoplasm, which is the region of the cell where the nucleus and organelles and other structures with specific functions are located [8]. However, retroviruses, which are positive-sense single-stranded RNA viruses, use a distinct method of replication [9]. Retroviruses replicate viral RNA using reverse transcription, where the transcriptase enzyme is carried by the virus inside its wrapper [10].

In the infection process, as mentioned previously, the virus needs a host cell, where it will attack one or several of the receptor molecules. The viral genetic material is then unencapsulated and replicated inside the host cell, requiring some specific enzymes. The host cell usually dies after the viral components are completely formed and releases new viruses that can infect other host cells [11].

2.1.2 Covid-19

Coronaviruses are positive-sense simple-stranded RNA viruses. They are enveloped in a coat of fat and protein, ranging in diameter from 60 to 140 nanometers [12], and the Spike Protein on their surface, which binds to the ACE2 enzyme, making infection easier. The arrangement of this protein resembles a crown (Figure 2.2).

Covid-19 has its known origin from the Chinese province of Wuhan, where a wave of infections began in late 2019. Several tests were done to detect the possible causes of these infections, which until then were being treated as pneumonia. Later, it was discovered that a new coronavirus was in action. Some of the symptoms that these viruses have in common are respiratory complications, which can range from a milder case to one with more severe complications, possibly leading to death.

Some sources claim that the virus began circulating from a wild animal sales market and from then on began to spread across the country and soon took over the whole world, leading to the death of approximately 3.20 million people to the date of this work.

According to the World Health Organization (WHO) [14], on average, it takes approximately one week for the infected person to show the symptoms like:

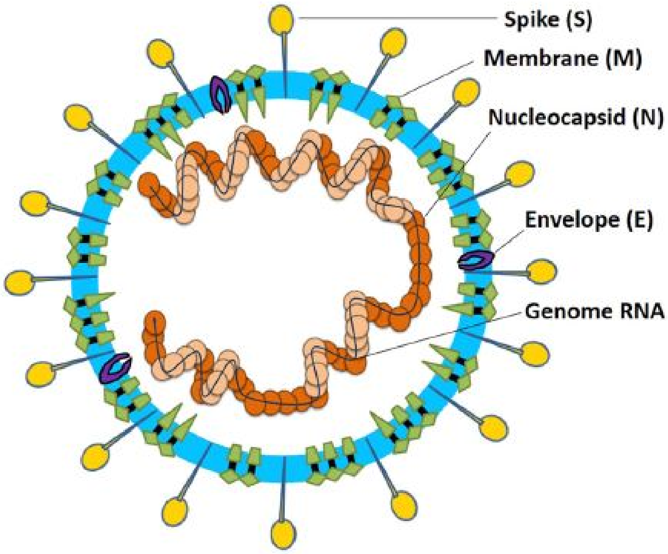


Figure 2.2: Coronavirus Structure [13]

- **Most common symptoms:**
  - Fever;
  - Dry cough;
  - Tiredness.
- **Less common symptoms:**
  - Aches and pains;
  - Sore throat;
  - Diarrhea;
  - Conjunctivitis;
  - Headache;
  - Loss of taste or smell;
  - Skin rashes, or discoloration of fingers or toes.
- **Serious symptoms:**

- Difficulty breathing or shortness of breath;
- Chest pain or pressure;
- Loss of speech or movement.

2.1.3 SARS-CoV-2 Variants

In 2020, the coronavirus caused enormous devastation and, seeking to stop it, there was a great rush to develop a vaccine that would be able to immunize the population against this virus. By the end of 2020, some countries were already starting vaccination campaigns following their respective protocols, which in most cases started with professionals who were on the front lines of fighting the virus and people in an older age group [15]–[17].

After the start of vaccination, it was hoped that the entire pandemic situation would soon be under control worldwide. However, in late 2020 and also in early 2021, new Covid-19 variants were discovered: 501Y.V2 (B.1.351) in South Africa, 501Y.V1 (B.1.1.7) in the United Kingdom, and P.1 (501Y.V3) in Brazil [18].

Variants of viruses usually arise from the various infections that the virus makes and from adaptations to the environment and conditions that surround them. One of the many concerns that now arises in connection with these new variants is whether the vaccines being developed will also be effective against them [19].

Some of the Covid-19 variants are:

- The British Variant: named a Variant of Concern 202012/01 (lineage B.1.1.7, VOC 202012/01), it began to be highlighted in late November 2020. It has a very large number of genetic mutations, in which many of them are in the Spike protein [20]. According to a report by the New and Emerging Respiratory Virus Threats Advisory Group (NERVTAG), this variant appears to have a higher transmissibility than other variants, and some initial studies have indicated a possible association of this new variant with an increased lethality rate [21].

- The South African Variant: also known as 501Y.V2 (B.1.351). Like the British variant has mutations in the receptor binding domain (RBD) of the Spike protein. This variant is a consequence of the first wave of COVID-19 in South Africa and was initially detected in Nelson Mandela Bay [22]. Mutations occurred in both the British and South African variants are reported to contribute to increased transmission and a potential to escape relevant antibodies [23].

- The Brazilian Variant: the P.1 (501Y.V3) lineage, as the Brazilian variant of SARS-CoV-2 is called, was discovered in early 2021 after a group of passengers were routinely tested at an airport in Japan upon their return from Amazonas, Brazil. This variant has mutations in the Spike protein similar to those found in B.1.1.7 and B.1.351, plus 17 different mutations than usual in amino acids [24].

2.1.4 Vaccines

With the urgency that the pandemic demanded for a form of immunization of the population, there was a great effort from scientists and researchers from many areas to develop vaccines against the Coronavirus in a very short period of time, compared to the normal development of a vaccine [25], [26]. According to the WHO [27], at the moment there are several vaccines already developed, such those that use a weakened part of the virus to generate antibodies, the protein-based ones, which are small protein fragments that generate an immune response by “mimicking” the Covid-19 virus, and the RNA and DNA-based ones, which use a genetic modification technique of the RNA or DNA to generate a protein that provokes an immune reaction. Some of the vaccines using these techniques are already in use, while others are awaiting regulation by the health authorities in each country.

2.2 Artificial Intelligence

The term AI, first introduced by John McCarthy, known as the father of the AI [28], refers to the ability of a machine to perform tasks commonly done by intelligent beings. These are systems programmed with the goal of emulating intelligent behavior through computational processes [29] and, from that, performing a certain action or decision similar to what a human would perform.

Technological advances allowed major advances in the area [30]. This includes the increase in computational capacity over the years built on the development of more powerful processors, memories with greater capacity and emergence of Graphics Processing Units (GPUs). As a result, machines, in some tasks, can already surpass humans. Some of the best known examples are DeepBlue, a supercomputer developed by IBM specifically to play chess, with the ability to analyze approximately 200 million positions per second, which beat the then world chess champion Garry Kasparov [31], and AlphaGo, a program developed by Google specifically to play the Chinese game Go, which beat the also world champion at the time, Lee Sidol, by 4 to 1 [32].

The area of AI has been overcoming many challenges and being a great asset not only in various socioeconomic activities, but also in medicine by helping doctors and researchers to detect and prevent various diseases [33]. Despite the great advances, there are still many perspectives and challenges in the field of artificial intelligence [34].

2.3 Natural Language Processing

NLP is a subfield of Artificial Intelligence that studies the understanding of natural language by machines [35]. One of the goals of NLP is to make the machine analyze and produce written or spoken texts, recognizing their context and being able to perform processes such as:

- Phonetic analysis: it studies the acoustic processes of speech, that is how a word actually sounds when spoken by someone.

- Phonology analysis: studies phonemes, which are acoustic units of a language, that may or may not have a concrete meaning, and the way they are organized.

- Morphology analysis: is the study of the structure, formation, and classification of words. It studies how words are formed from morphemes, which are minimal units capable of expressing meaning.
- Lexicon analysis: is the set of words that belongs to a language.
- Syntax analysis: studies the words within sentences or clauses and the relationship they create with each other to make up meaning.
- Semantic analysis: studies the meaning and interpretation of the meaning of a word, sentence, phrase, or expression in a given context.
- Pragmatic analysis: studies the influence and use of context in the interactions between speaker and listener.

NLP can also be used for text classification [36] or audio classification [37], where the algorithm is fed with a document or just a sentence and each word in that sentence can be assigned to a value. This value can be assigned in different manners, including one-hot Encoding, a 1xN matrix used to differentiate each word of the vocabulary, consisting of zeros all over the vector with the exception of a single 1 that identifies a particular word [38]. There is also the approach of Word Embeddings [39], i.e., dense vector representations of words. In this approach, each word is also assigned a value and the more semantically close one word is to another, the more similar the value will also be.

2.4 Machine Learning

ML can be seen as a subarea of AI that provides systems with the ability to learn from a dataset of past examples or experiences, and then make some inference on future similar tasks. Mitchell [40] defined ML as:

> A computer program is said to learn from experience E referring to a class of tasks T and performance measure P, if its performance on task T, measured by P, improves with experience E.

To illustrate the quote, one can use the task of correctly sorting apples and oranges into a basket from a previously provided data set, where:

- Task T: correctly classify what is apple and what is orange;
- Experience E: set of data previously provided for the machine to get a sense of what apples and oranges are;
- Performance P: number of correct matches obtained by the machine during the classification.

Depending on the type of task, ML algorithms can be trained from techniques such as Supervised Training, Unsupervised Training, or Reinforcement Learning.

Supervised Training is a training technique for ML algorithms in which the system is provided with labeled data with correct task responses. From this, the parameters for the algorithm can be adjusted during training so that the machine can learn to correctly, or with the lowest possible error rate, classify the unlabeled data [41]. K-Nearest Neighbors, Decision Trees, Random Forest and Support Vector Machine are some of the machine learning algorithms that apply the supervised approach.

The Unsupervised Training, unlike the previous one, does not provide much or any information about the data. So the algorithm must observe some patterns among the data and group them according to the similarities found. Some data that are very different from the others can be seen as anomalies [42].

Reinforcement Learning is a technique in which no training data is given to the agent [42]. In this case, the agent should perform an action in an environment where it should be rewarded or punished depending on how its behavior was during the task. With each action the behavior should become more complex and thus should behave more effectively during iterations [43].

2.4.1 K-Nearest Neighbors

K-Nearest Neighbors is a supervised algorithm used in machine learning and verifies how similar the data is to each other. The training data consists of vectors of n dimensions. The algorithm calculates the similarity of a new data with other data that have already been classified, and tries to find the K nearest neighbors of that new data [44]. The similarity between the data can be calculated using the Euclidean distance, Manhattan, Minkowski, Weighted or other metrics.

The classification process of the K-Nearest Neighbors algorithm receives an unclassified data, that will have its distance calculated with the other data that have already been classified. After the calculation of the distances, the K smallest distances will be obtained. From the quantity measured by the algorithm, the class that has the most is the one that should be assigned to the new data [45].

Figure 2.3 demonstrates an example of how K-Nearest Neighbors works during the classification process, where the red dot in the middle of the circle has not been classified yet, so it will calculate the distance to the other data already classified (yellow and purple dots). After calculating, for $K = 3$, the red dot should be assigned to the purple class, because it has the greatest number of neighbors in the surrounding. In case that $k = 6$, it should be assigned to the yellow class because from the 6 neighbors the yellow appears the most.

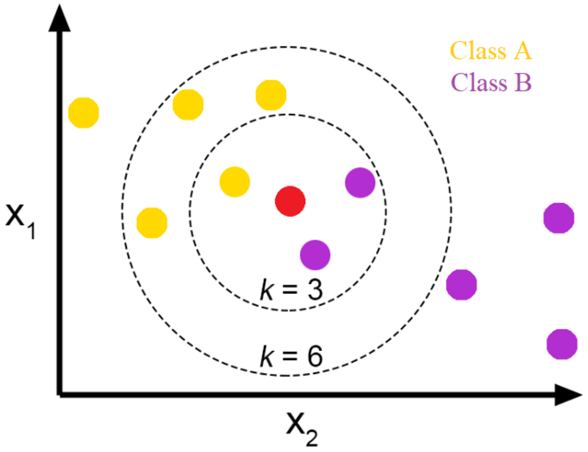


Figure 2.3: KNN classification example [45]

2.4.2 Naive Bayes

Naive Bayes, in machine learning, is a supervised algorithm used as a classifier that is based on the probability of an event occurring, without the correlation between the features being taken into account. It is widely used in text classification, especially for classifying spam in an e-mail box, document separation, and other tasks.

The Naive Bayes algorithm is inspired by Bayes' theorem [46], that describes the probability of an event, based on *a priori* knowledge and it is represented in Equation 2.2.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

In Equation 2.2 $P(A|B)$ is called *posterior* probability, and it represents the degree of confidence of A after observations of the feature vector B. $P(A)$ is the initial degree of confidence in A and it is independent of B.

$$P(y|x_1, x_2, x_3, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)P(x_3|y)\ldots P(x_n|y) * P(y)}{P(x_1)P(x_2)P(x_3)\ldots P(x_n)} \tag{2.2}$$

Given that B is a feature vector, it can be assumed as $B = x_1, x_2, x_3, \ldots, x_n$ and $A = y$, as it is the objective output. So if substitute the values of B and A, the Equation will be like in Equation 2.3. This equation can be simplified by using the product operator that, in this case, indicates that each feature x from $i = 1$ til $i = n$ is multiplied by y, as represented in Equation **??**.

$$= \frac{P(y)\prod_{i=1}^{n} P(x_i|y))}{P(x_1)P(x_2)P(x_3)\ldots P(x_n)} \tag{2.3}$$

As $P(B) = P(x_1), P(x_2), P(x_3), \ldots, P(x_n)$ will remain as a constant, and will be the same for every feature, it may be ignored and removed from the equation.

The Naive Bayes algorithm tries to return the most probable class, so it makes use of the *argmax* function to return class with maximum probabilities from the obtained. For instance, if the model returns a probability of 0.7 for the Positive class over 0.3 for the Negative, in this case the model should return return a probability the greater. The Naive Bayes formula is represented in Equation 2.4.

$$y = argmax_y(P(y)\prod_{i=1}^{n} P(x_i|y)) \tag{2.4}$$

2.4.3 Decision Tree and Random Forest Classifier

Decision Tree is a type of classifier that sorts the attributes of an instance from the root of the tree. Attributes will compose internal nodes of the tree while classes will compose the leaves [40]. Figure 2.4 demonstrates a Decision Tree for predicting whether a person will play tennis or not. The attributes (Outlook, Humidity, Wind) are on the nodes and each of them will be split with the values of the respective attribute in which the edges are going to be attribute values.

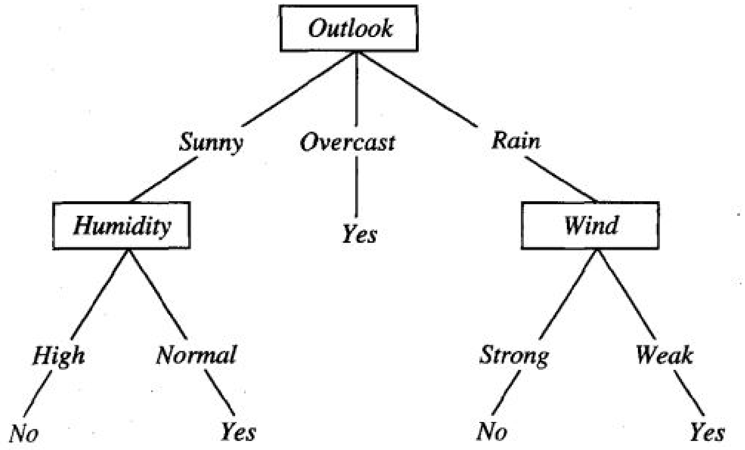


Figure 2.4: Decision Tree for playing tennis prediction [40]

Random Forest Classifier is a machine learning algorithm used in various classification and regression tasks. It is built by creating a huge number of decision trees where each tree relies on the values of a random vector with similar distribution and independent samples [47]. Each of the trees returns a classification that will be "voted" by the algorithm. The class with the most votes will be the chosen one [48].

In order to decrease the correlation of the trees, which could negatively affect the generalization of the model, approaches like Bootstrap Aggregation or Feature Randomness can be used. In Bootstrap Aggregation, a random sample is detached from the tree keeping the original size of the data. In Feature Randomness, each tree has a random set of features that can be used.

2.4.4 Support Vector Machine

Support Vector Machine-SVM is a machine learning algorithm that plays a good role for classification or regression tasks. SVM aims to create the best hyperplane in an N-dimensional space, being N the number of features, to classify discretely the data [49].

SVM has shown to be more accurate in some tasks than decision trees or neural network based approaches [50]. Given a set of training examples, the objective of the hyperplane is to separate the set in a way that every instance with the same labels stays on the same side [51]. Figure 2.5 shows two representations for separable and non-separable data.

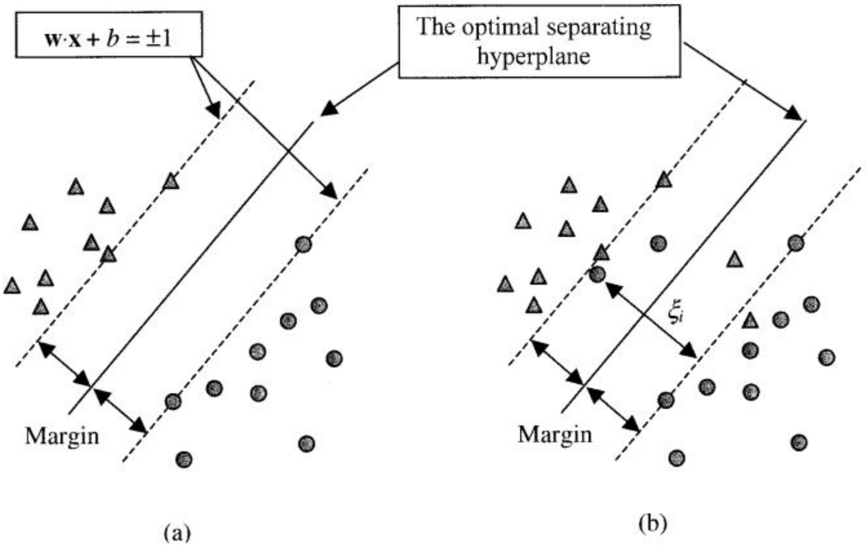

$\mathbf{w}\cdot\mathbf{x} + b = \pm 1$
The optimal separating hyperplane
ξ_i
Margin
Margin
(a)
(b)

Figure 2.5: SVM hyperplanes for separable and non-separable data [50]

On Figure 2.5 two classes are presented, $+1$ and -1, represented as circles and triangles, with k pairs $(x_1, y_1), ..., (x_n, y_n)$ where $x_i \in \mathbb{R}^n$ are the inputs of an n-dimensional space.

Once the algorithm has the data, it gets the training samples, support vectors, that stays on boundaries of the class and creates the Optimal Separating Hyperplan (OSH). The OSH separates the data with maximum margin and can be achieved by minimizing the norm of w [51]. In Figure 2.5, the hyperplane is created in a way that best separate both classes, also is created two additional planes that have maximum margin between the central hyperplane and the nearest support vector, in this case, the support vector will the triangle and circle points.

The SVM kernel presents a way to solve problems not linearly separable. SVM Kernel tries to convert a problem from a low dimensional space into a higher dimensional. In Figure 2.6, for instance, are presented two classes that are disposed in a way that is not possible to separate them linearly in a 2-dimension space. By setting a Kernel of 3, in this case, would convert the space 2-D space into a 3-D space, and make the creation of a hyperplane possible.

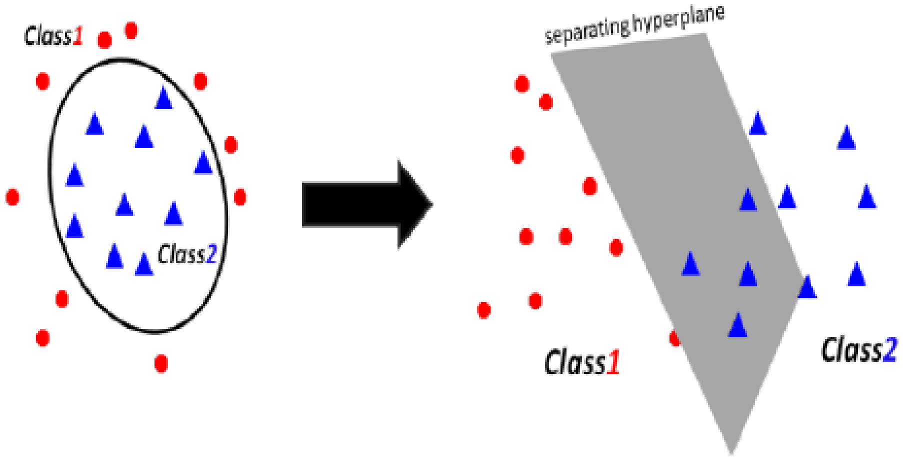


Figure 2.6: Separating classes in a higher dimensional space [52]

SVM can be applied to regression tasks and, in this case is called Support Vector Regression (SVR). SVR attempts to find a function that presents the greatest deviation among all examples [53]. In the regression approach margins are also used, but no costs are incurred for the instances that are within this margin.

2.5 Artificial Neural Network

Artificial Neural Networks (ANN) are computational techniques inspired by the human brain that use mathematical models to try to simulate the operation of biological neural networks.

Neurophysiologist Warren McCulloch and mathematician Walter Pitts were the first to introduce a mathematical model to simulate a biological neural network [54]. Their work is considered a landmark in the history of ANNs [55].

In 1949, Hebb proposed a theory that learning in complex nervous systems could be reduced to a local process, and the intensity of synaptic connections is changed only as a function of locally detected errors [56]. Later, Frank Rosenblat created the Perceptron network, a network with multiple neurons of linear discriminator type [57].

In the 1970s, with the publication of Marvin Minsky's book *Perceptron* [58], research in the area practically came to a stop, as the book emphatically pointed out the limitations of the Perceptron network, such as not being able to solve the or-exclusive problem.

The field resumed its development in the 1980s with models such as the *Adaline* [59] and *Madaline* [60], and later with the *Backpropagation* algorithm. Since then ANN had successes and failures in their models. However their use has been growing rapidly in many applications [61] and has being widely used in several areas such as Computer Vision, Natural Language Processing, among others.

2.5.1 Neuron

Neurons are cells capable of receiving, processing, and transmitting information by electrochemical signals called action potentials. Neurons are composed of body (soma) with nucleus, axons, and dendrites, as shown in Figure 2.7.

The dendrites are thin, branched extensions that carry nerve impulses picked up from the environment or from other cells toward the cell body. The impulse is processed and new ones are generated. These impulses are conducted by the axon, which is responsible for taking the impulses coming from the cell body to the synapse, and passed on to another

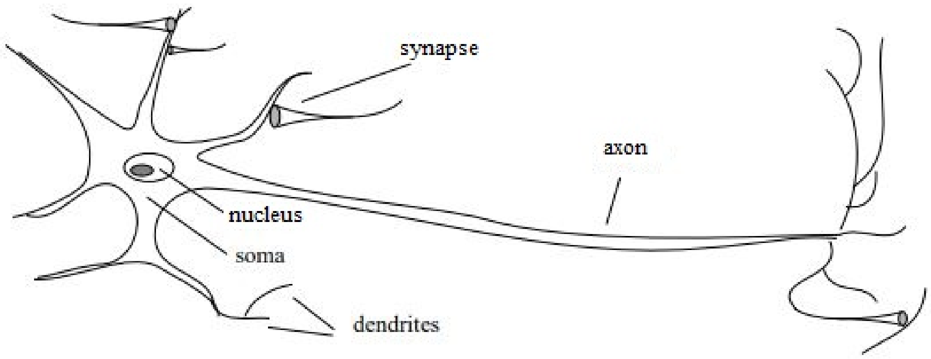


Figure 2.7: Biological Neuron [62]

neuron. The synapse is responsible for making contact between two neurons, consisting of the presynaptic side, from where the impulse is sent, and the postsynaptic side, where the impulse is received. When the neuron is sufficiently excited it fires, and then performs the whole process of transmission between neurons [62].

ANNs are based on the functioning of the biological neuron. They are composed of nodes or units connected by directed links. They represent a simple mathematical model that presents a linear combination when exceeds some threshold triggers the neuron. Its connections have associated numerical weights. As in linear regression models, each unit has an input x_i with an associated weight w_i, where a weighted sum of all inputs and a bias is computed.

2.5.2 Perceptron

The Perceptron network introduced by Frank Rosenblat is a representation of the artificial neuron model [30], and although it does not have great potential today, its study is justified by its simplicity and its historical role.

The Perceptron is built around McCulloch-Pitts' model of a neuron [63]. It can be viewed as the simplest type of *Feedforward* neural network, in which information flows only from the input layer to the output layer. It can be seen as a classifier for linearly separable problems.

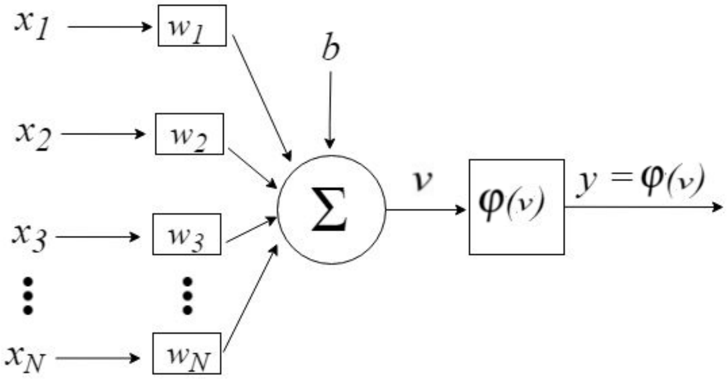


Figure 2.8: Perceptron [64]

As represented in Figure 2.8, the Perceptron is fed with a vector of inputs $\hat{X} = [x_1, x_2, x_3, \cdots, x_N]$ which is multiplied by a vector of weights $\hat{W} = [w_1, w_2, w_3, \cdots, w_N]$ corresponding to each input and then combined with the *bias* b, as represented in Equation 2.5 [64].

$$v = \sum_{i=1}^{N} x_i w_i + b \tag{2.5}$$

The output v of Equation 2.5 will produce a value that is passed to a given activation function $\varphi(v)$. In the case of Perceptron, the Step function is used, which evaluates whether the activation threshold is greater than zero, returning 1 in this case and representing that the neuron is active. Otherwise, there will be no activation, as seen in Equation 2.6.

$$\varphi(v) = \begin{cases} 1, if \geq 0 \\ 0, if < 0 \end{cases} \tag{2.6}$$

2.5.3 MultiLayer Perceptron

One of the problems pointed out by Minsky [58] about the simple Perceptron network is the inability to solve non-linearly separable problems, such as the or-exclusive problem. The Multilayer Perceptron (MLP) network is similar to the Perceptron network but with more neurons and layers of neurons.

The MLPs are composed of an input layer to receive the signal, an output layer where the decision or prediction of the input layer will be returned, and between them, an arbitrary number of hidden layers [63].

MLPs are usually trained using the algorithms Backpropagation and Gradient Descent (or variations) [65]. In Figure 2.9, a fully connected MLP of type *Feedforward* is illustrated [66]. In fully connected networks, a neuron in any layer of the network is connected to all nodes/neurons in neighboring layers. In the Feedforward networks, the signal flows from the input layer to the output layer without loops. The *Backpropagation* algorithm is based on the error correction learning rule. It is a supervised training algorithm, where a training set with a correct labeling of the outputs is used.

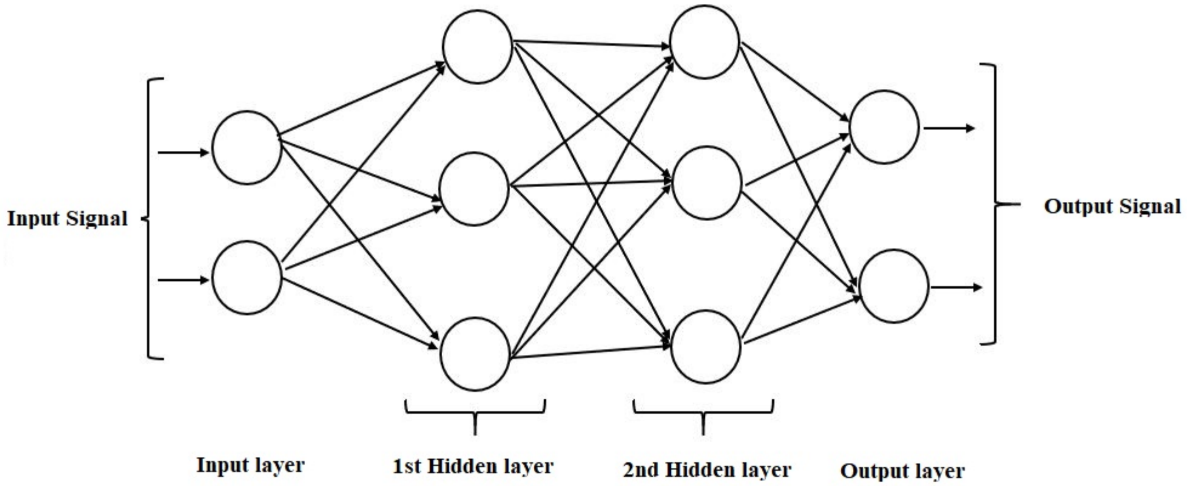


Figure 2.9: Fully connected feedforward MLP [63]

2.5.4 Activation Functions

Activation functions determine how the neuron will activate. There are several types of activation functions, including the Logistic Sigmoid, Hyperbolic Tangent, ReLu, among others. The choice of activation function depends on the type of network or problem being studied, since it can influence the performance and complexity of the network [67].

The Logistic Sigmoid Function generally works better when it have to deal with the output prediction as probability. It uses a real input value, x, and returns a value in the range between 0 and 1 [68]. It is continuous, and differentiable at all its domain [63]. Its formula is represented by Equation 2.7.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

The Hyperbolic Tangent function, *tanh*, is very similar to the Logistic Sigmoid. It is mainly used in tasks where the desired classification is between two classes and it is defined by the division of hyperbolic sine by hyperbolic cosine, which can also be represented by the division between the difference of two exponential functions at points x and $-x$, and the sum of these same functions, according to Equation 2.8. The function *tanh* is similar to the logistic sigmoid, but its range is from -1 to 1 [69]. Figure 2.10 shows the graph of the two functions.

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

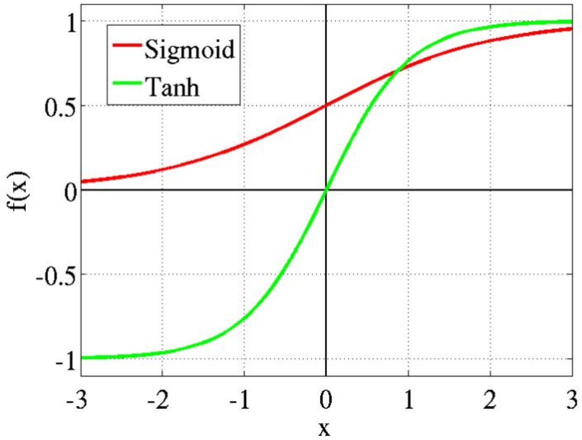


Figure 2.10: Logistic Sigmoid and Tanh activation functions [70]

ReLU is a piecewise activation function very used in models like CNN, for instance [71]. It suits better in complex models. Some of the advantages is that it avoids and rectifies the vanishing gradient problem. It will also only deactivate the neurons if their result is less than 0. In this case, the negative values will be equal to 0 as disposed on Equation2.9 in which the maximum value between 0 and the input value is returned, and when the value is smaller than 0 it should return 0. The plot of the ReLU function is represented in Figure 2.11.

$$\sigma(x) = max(0, x) \tag{2.9}$$

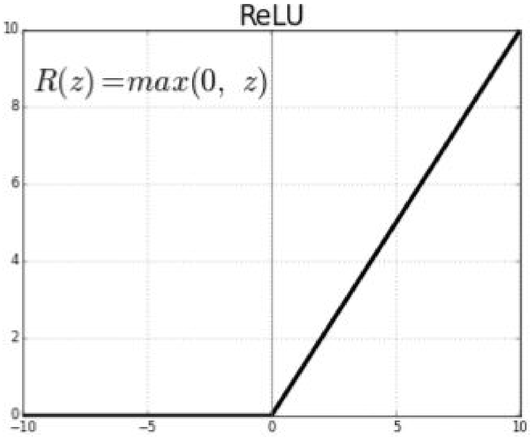


Figure 2.11: ReLU [72]

2.5.5 MultiLayer Perceptron Training

The Backpropagation is divided into two parts: forward pass and backward pass. During the forward pass phase, input signals flow through the network and output predictions obtained. In the backward pass phase, the gradient of a given loss function in the final layer of the network is calculated and using the chain rule will update the weights and biases of the network from the output to the input [73]. This process should repeat to the next set of inputs until an acceptable error is reached.

The Gradient Descent algorithm determines a vector of weights that minimizes the prediction error. The vector of weights of the network are initialized arbitrarily and, at each step, the it is changed in the direction of the gradient that produces the largest drop along the error surface [74].

The Backpropagation uses, for adjustment of the weights and biases, the derivative of a cost function, such as Mean Square Error (MSE) cost, and the derivatives of the activation functions involved, such as the logistic Sigmoid function. The calculation for adjusting the weights in layers is represented in Equation 2.10 adapted from [40], where the weight w_{ij} connects the input neuron i to the output neuron j. So, the weight w_{ij} will be adjusted by summing its value with its derivative.

$$w_{ij} = w_{ij} + \Delta w_{ji} \tag{2.10}$$

The calculation of the Δw_{ij}, in the output layers is shown in Equation 2.11, where is applied a given learning rate η to the calculation of the obtained output o and the expected output t

$$\Delta w_{ji} = \eta(t_j - o_j)o_j(1 - o_j)x_{ji} \tag{2.11}$$

In hidden layers the term δ is used to the adjustments of the weight for unit j. The calculation is represented in Equation 2.12 where is made a summation of the input weights for the unit and multiplied by the output obtained o and the difference between 1 and the output obtained o.

$$\delta_j = o_j(1 - o_j) \sum w_{jk}\delta_k \tag{2.12}$$

The adjustment of bias is represented in Equation 2.13, in which the current value of bias subtracted by the product between the learning rate and the corresponding δ term is calculated.

$$b_j = b_j - \eta\delta_j \tag{2.13}$$

2.5.6 Convolutional Neural Network

Convolutional Neural Network-CNN is a type of ANN oriented towards processing grid type data. Despite great developments in other types of neural networks, CNNs have provided an even greater leap forward in several areas involving pattern recognition, with tasks ranging from image processing to speech recognition [66]. Eventually, CNNs also came to be used for tasks involving text, such as text classification or sentence modeling [75] where the process throughout the classification task, for instance, is similar to the image classification. The only difference is that the input of the text will be a matrix of word vectors [76].

CNN is typically composed of three layers in its architecture: the convolution layer, the pooling layer and the fully-connected layer, as show in Figure 2.12:

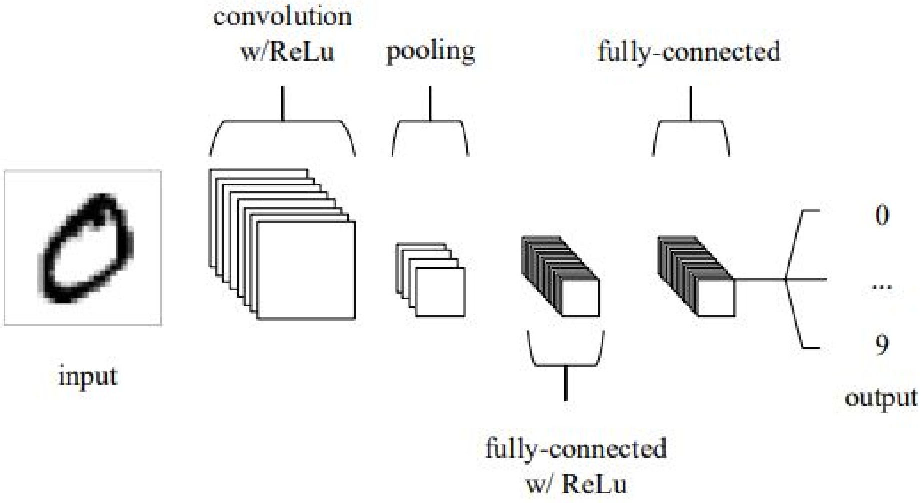


Figure 2.12: CNN Architecture on an image classification task [77]

The input to a convolutional network will be of matrix type data, that may be composed of width, height and depth. For images with more than 2 dimensions, or in RGB format, the images are divided into channels, as represented in Figure 2.13.

In the convolution layer these inputs should pass through a filter or kernel where the characteristics from the input image represented in matrix format should be extracted [77]. This filter may have a defined size up to the maximum size of the image. The filter

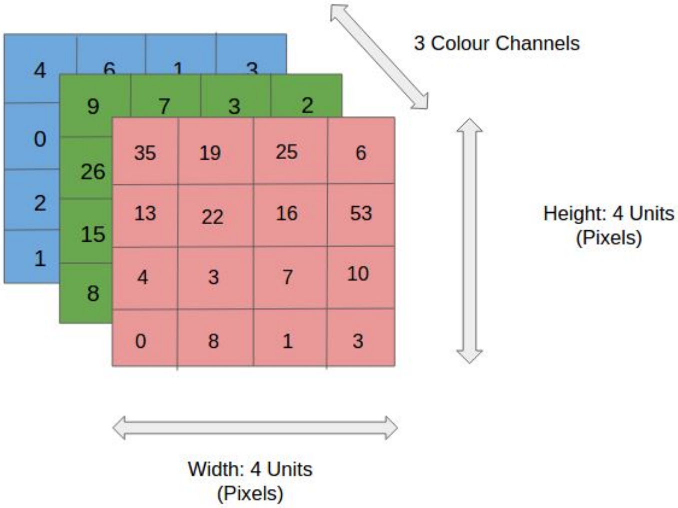


Figure 2.13: Channels of a CNN input [78]

can be initialized with random values, where a dot-product operation will be made with the elements of the input matrix and the resulting value will be stored in a feature map whose size will be according to the size of the filter and the stride used. The stride is the amount of nodes in the matrix that will be skipped on each pass of the filter. The resulting size of the feature map of an image, for example 7x7, will be 5x5, with a stride of 1, or 3x3 with a stride of 2, using a kernel of size 3x3. Figure 2.14 demonstrates the route followed by a 3x3 kernel in an image, or matrix, of size 7x7 and stride 1.

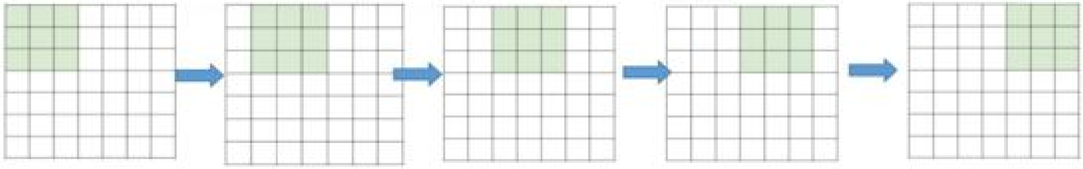
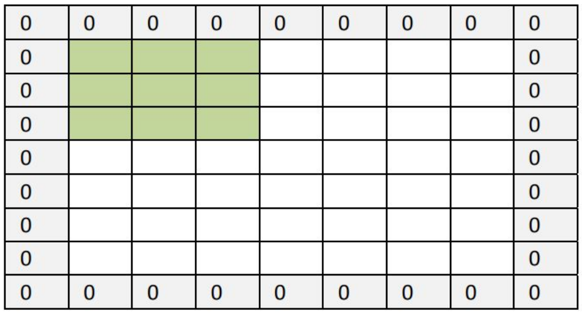


Figure 2.14: Filter route with stride 1 [79]

Depending on the image size, filter and stride, important information may be lost at the edge of the input image during the dot-product operation. To mitigate this problem values can be inserted at the border of the image. This technique is called padding. One approach for padding that may be adopted is the zero-padding that fills the image border with zeros and does not alter the characteristics of the image but the size, as the padding values will be multiplied with that of the filter and result in zeros [79]. Figure 2.15 shows how the zero-padding technique.



0	0	0	0	0	0	0	0	0
0	🟩	🟩	🟩					0
0	🟩	🟩	🟩					0
0	🟩	🟩	🟩					0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Figure 2.15: Zero Padding [79]

In the pooling layer a down-sampling of the representation will be done with the intention of reducing the complexity generated in the filters. Among the different pooling methods, the most common is Max-pooling, where the largest value within the given filtered region will be used, with the most common filter being 2x2 [79]. Figure 2.16 shows a representation of a Max-pooling operation, where for the pink 2x2 input region the maximum value is 6, the green is 8, the orange is 3 and the blue is 4.

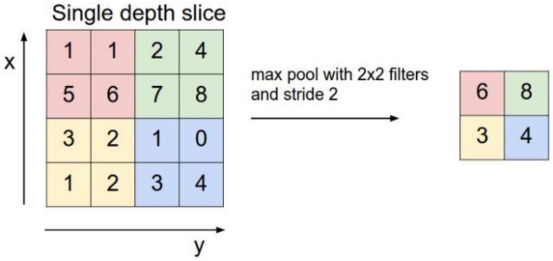


Figure 2.16: Max Pooling [80]

The fully connected layer is not so different from a traditional neural network. Thus each neuron will be connected to every node in the previous and subsequent layers [79].

2.6 Advances in Natural Language Processing Models

2.7 Transformers

Transformer model uses sequence to sequence (seq2seq) and self-attention mechanism to obtain features of each word in relation to the others in the sentence. It is based on the paper "*Attention is all you need*" [81], published by a group of researchers from Google. The Transformer model, unlike other models that deal with Natural Language Processing, does not make use of recurrence or convolution [82], it relies almost entirely on the attention mechanism [83].

The Transformer model has an Encoder-Decoder structure where the Encoder maps the input sequence X to another sequence Z. Then the decoder generates an output sequence of each element, and can make use of the previous elements in addition to support the generation of the output sentence [81]. Figure 2.17 presents the overall architecture of a Transformer model.

On the left side of the picture, the input side, is presented the encoder architecture, where the first step is to pass the whole sentence through an Embedding layer. After Embedding the sentences, the next step is insert information about the position into the Embedding, since the Transformer Model does not use recurrence. This can be achieved by using sine and cosine functions, as presented on Equations **??** and **??** respectively. All the even index on the input vector will be applied to a function using sine and the odds to a cosine.

The step after Positional Encoding pass the result encoding to a Multihead Attention, that is composed of multiple Scaled Dot-Product Attention and permits the model to handle information from other positions in different representation subspaces that wouldn't be possible using a single attention head. The Scaled Dot-Product Attention is used to get the context vector that is a context vector is a weighted average of the encoder's annotations.

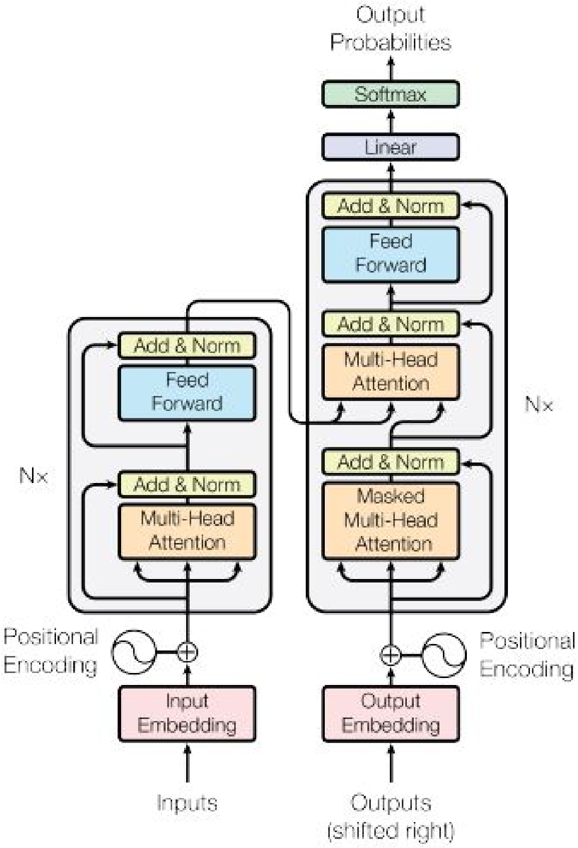


Figure 2.17: Transformer Architecture [81]

The next step is to pass the result through a Normalization layer [84] that will be added with the original embedded and positioned sentence. After that, it will pass through a fully connected Feed Forward layer and then Normalize the result, adding the data obtained on the previous Normalization.

The Decoder architecture runs in a similar manner to the encoder. It is fed with the outputs shifted one position to the right, in order to inhibit positions from stack attending to subsequent positions, which will pass also through the Embedding layer, Positional Encoding, Multihead Attention and Normalization. The difference is that before passing through the Feed Forward layer, it is added another Multihead Attention layer which is fed with the Encoder output and the Decoder result, then the information will pass through a Normalization layer, a Feed Forward layer and Normalization. The output of the Decoder passes through a Linear layer that can have the size of the vocabulary and a Softmax function [85] that will generate probabilities for each word.

2.8 Bidirectional Encoder Representations from Transformers

The Bidirectional Encoder Representations from Transformers is a recent approach developed by Google researchers. From the moment of its release, in 2018, this model stands as a state-of-art technique for tasks involving Natural Language Processing [86].

The technique involving BERT consists of a deeper pre-training of the Transformer model when trying to find the sentence context by processing the input in two directions, from the beginning to the end and from the end to the beginning, as opposed to other models that usually process the input only in one direction. During pre-training, the Masked Language Model (MLM) technique is used in which an input token is masked and the model aims to predict this token [87]. The bidirectional processing of the input helps in predicting the masked token, since it has a finer understanding of the sentence context. The Next Sentence Prediction (NSP) [88] technique is used along with these.

This technique uses sentence pair representations, where two input sentences separated by the token [SEP] will be provided. The goal is to find a relationship between sentences A and B. The sentences can be generated from a monolingual corpus where, when choosing each pair of sentences A and B, half of the time B will be the real sentence that succeeds A (labeled in this case as IsNext) and, in the other half being labeled as NotNext when the sentence B is not the one that actually succeeds A [89].

As shown in figure 2.18, each BERT input token is composed of the sum of token embedding, segment embedding and positional embedding.

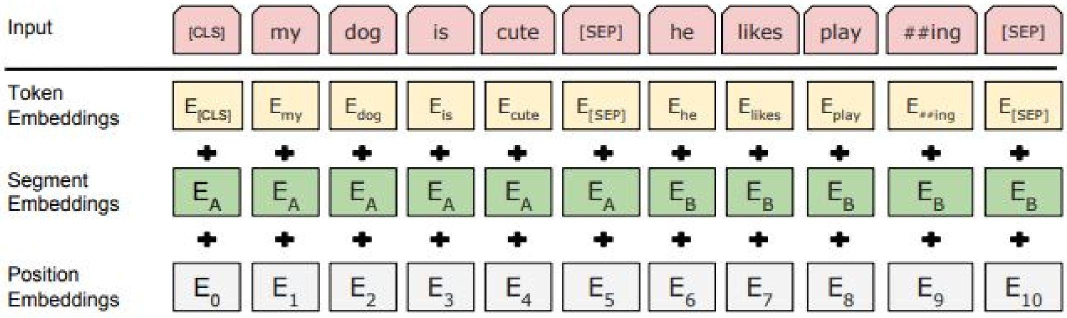


Figure 2.18: Bert input sentence representation [89]

Using the BERT model is typically done in two stages. The first one, discussed previously, is pre-training the model using unlabeled data. The second is the called fine-tuning, where the model should be initialized with parameters used in the pre-training stage and then fine-tuned during the training of the task to which it is being applied. Figure 2.19 shows a representation of both stages of the model, where the left side of the figure shows the pre-training stage using masked sentence pairs, [SEP] to separate sentence A from B and the classification token [CLS] at the beginning of every input [89].

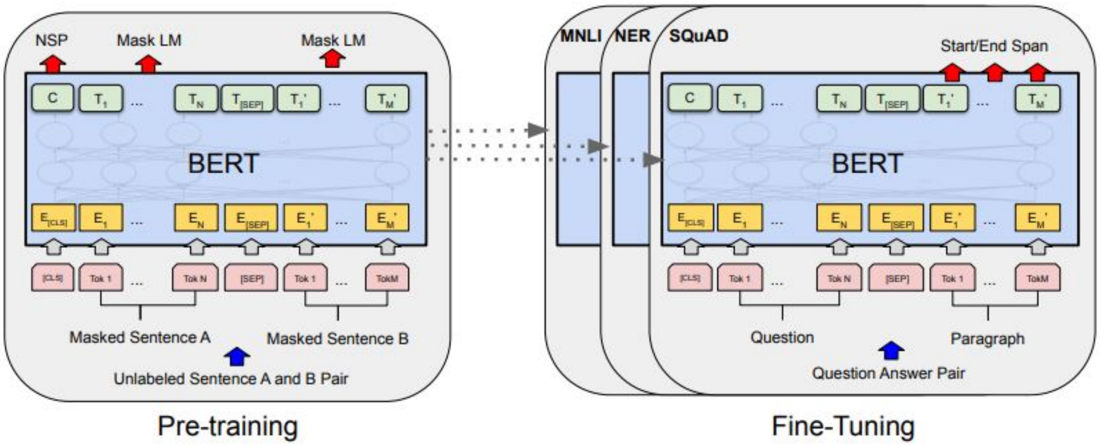


Figure 2.19: Bert pre-training and fine-tuning representation [89]

Chapter 3

Materials and Methods

This chapter will present the technologies and tools, dataset and procedures used in the development of this work.

3.1 Tools

To collect the dataset, implement the models and run the tests we used the following tools were used:

- Python[1]: a high-level, dynamically typed programming language that allows the creation of applications using object-oriented, imperative, functional or procedural paradigms;
- Tensorflow[2]: an open source library for numerical computation using computational graphs. This library was developed by the Google Brain team for machine learning and deep neural network research, but it can be used for a variety of purposes [90].
- Natural Language Toolkit - NLTK[3]: is a library for natural language processing. It is written in Python and it is useful for pre-processing texts because it provides a variety of functions for this purpose, like transforming words into tokens [91].

[1] Official website: `https://www.python.org/`

[2] `https://www.tensorflow.org/`

[3] `https://www.nltk.org/`

- Google Colab[4]: Google Colaboratory, also known as Colab is a research platform provided by Google that offers a Jupyter Notebook environment with support for Graphical Processing Unit (GPU) and Tensor Processing Unit (TPU) accelerator [92]. Colab is free to use and allows users to run their Machine Learning and Deep Learning models in an interactive development manner.

- scikit-learn[5]: an open-source Machine Learning tool, most of which was developed in Python. With scikit-learn it is possible to perform many different kinds of ML tasks, like classification, regression, clustering, using a huge variety of algorithms, like Random Forest, K-Means, SVM, Naive Bayes, among others, with a few lines of code and little complexity.

- Twarc[6]: a Python library tool used to obtain Twitter data. Each tweet is retrieved in a JSON format. With this tool is possible to collect users, trends, and hydrate tweet ids.

3.1.1 Hardware

To collect the dataset was used a Linux Virtual Machine instantiated in the cluster of Research Centre in Digitalization and Intelligent Robotics (CeDRI) at the Polytechnic Institute of Bragança - IPB. The virtual machine had the following specification: Ubuntu Mate OS, CPU AMD Epic Quad Core 64 bits 2400 MHz, 16GB RAM, GPU NVIDIA TU102 - GeForce RTX 2080 Ti.

To run the models was used the Google Colab whose specification was: Jupyter Notebook environment, CPU Intel(R) Xeon(R) CPU @ 2.30GHz, 12GB RAM, GPU Tesla K80.

In addition to the machines previously described to collect the dataset and run the models, was also used a personal machine to make the dataset labeling. The specifications

[4] https://colab.research.google.com/

[5] https://scikit-learn.org/stable/

[6] https://github.com/DocNow/twarc

for this machine was: Windows 10 PRO, CPU Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 64 bits, 12GB RAM.

3.2 Dataset

The dataset to train the models was obtained from a collection of Tweets made available by Lamsal[7] [93]. In Lamsal's work, millions of Tweets were collected to perform sentiment classification task regarding the virus. To collect the dataset several keywords related to the virus were used. Words such as COVID, COVID19, Coronavirus, virus, pandemic, and others.

The available dataset provided only the IDs of tweets, following Twitter's privacy policy. The "hydration" process, as it is called, is the process in which information regarding the tweet is retrieved from the Twitter platform using its ID. To "hydrate" the ids it waas used the Twarc tool.

After installing the Twarc tool, through the Package Installer for Python (PIP), it was necessary to register on the Twitter Developer platform[8], that provided access keys that later were inserted into the Twarc configurations and passed as parameters in the tweets requests. The data returned from Twarc requests was in JSON format, which brought a wide variety of information. However, as only the tweets themselves were needed, a filter was applied to the JSON object to get only the ID and the text of the tweet. The raw text of the tweets and its respective IDs were saved in a Comma Separated Values (CSV) file for labeling and pre-procesing. The process to request the tweets from the Twitter platform was the one that took the most time as it was put into sleep for around 5 minutes after a certain number of requests.

The process of hydrating the tweets took approximately 3 months, between December/2020 and February/2021, due to the waiting time after a certain amount of requests to the Twitter platform and a loss of data in the meantime. Although the dataset was

[7] `https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset`

[8] `https://developer.twitter.com/`

collected at the end of 2020 and the beginning of 2021, it contained tweets only until August/2020 because of the available dataset.

3.2.1 Labeling

One of the challenges of this work was to find a dataset that was already labeled for the task of finding symptoms or self-report tweets of Covid-19, but until the implementation of this work is was not possible to find one. Therefore, it was necessary to label the data before the implementation of the models and the training. The dataset obtained from Lamsal[9] [93] contained only tweets with keywords referring to Covid-19, but no further annotations. Therefore the work of Sarker [94] was used as reference to label the tweets, using regular expressions and keywords to filter it.

At first, keywords like *Positive* were used in combination with one or more of the Covid-19 symptoms described in Section 2.1.2, provided by the WHO [14]. The tweets of the positive class were labeled with the value 1, while those of the negative class were labeled with 0. At first, tweets that contained the word *Negative* or *Tested Negative* were labeled as class Negative. However at the end of the labeling, the dataset was unbalanced, with a larger number of positive instances. To label the classes based on the symptoms, regular expressions were used.

To balance the dataset, more negative tweets were collected by selecting the ones that did not contained Covid-19 symptoms, *Tested Positive* or *Positive* keywords up to a quantity close to the number of positive tweets. Further manual verification was made in attempt to ensure that the data labels were correct and to minimize the errors in labeling.

3.2.2 Pre-processing

Before the training and testing phase, it was necessary to perform a cleaning on tweets to remove unknown and special characters like hashtag (#), that directs the user to a page with the same topic, @ to tag a user in a post and emoticons.

[9] `https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset`

In order to clean the dataset, we performed the following steps were performed:

- Emoticon removal: emoticons are expressed by a set of hexadecimal codes that represent a small figure on the text. A list of emoticons codes[10] that was used to remove them from the text; for example, the code U+1F642 represents a slightly smiling face;

- Text decapitalization: consisted in converting all the words in the sentence into the decapitalized form. This could avoid the problem of having two same words in the dictionary, for example, *Tomorrow -> tomorrow*. To decapitalize the words, the *String* library from Python was used;

- User tagging removal: the removal of the tags was done using regular expressions. Tagging users in tweets is made by adding a "@" character before the user name, or page. So in the regular expression the words starting with this character were removed, for example, *@someUser*.

- URL Removal: in several tweets it was possible to verify the presence of references to other pages through links. The removal of these links was also done using regular expressions to locate full URLs that were then removed.

- White Space merging/removal: This consisted of replacing multiple spacing with single spacing. The removal of the original content from the tweets left gaps of multiple spaces in the middle of the text. These gaps were replaced with single spaces. If it was at the end or the beginning of the text, it was simply removed instead of being replaced. To remove unnecessary spaces in the text, the *String* library from Python was used;

- Stop Words Removal: Stop Words are words that occur very frequently in the text but do not usually have much importance in the sentence context, and can cause a higher processing cost when passed on the classification models [95]. For this

[10] List of Emoticons codes: `https://www.alt-codes.net/smiley_alt_codes.php`

reason, words like *as, the, be, are,* etc, were removed from the sentences. NLTK has a list of Stop Words of the English language, which was used to remove the Stop Words from the dataset.

- Lematization: transforming inflected words into their canonical forms, for example, the plural of a noun into the singular form, or a conjugated verb into its infinitive form. WordNetLemmatizer is an NLTK library used to perform this part of the pre-processing, for example, *Plays, playing, plays -> play.*

3.2.3 Characteristics of the Tweets

The collected tweets included words related to covid terms. The dataset collected amounted to 22991 tweets; from this total, 9680 tweets were classified as negative and 13311 as positive. For the whole dataset, the 10 most frequent words in tweets are show in Figure 3.1.

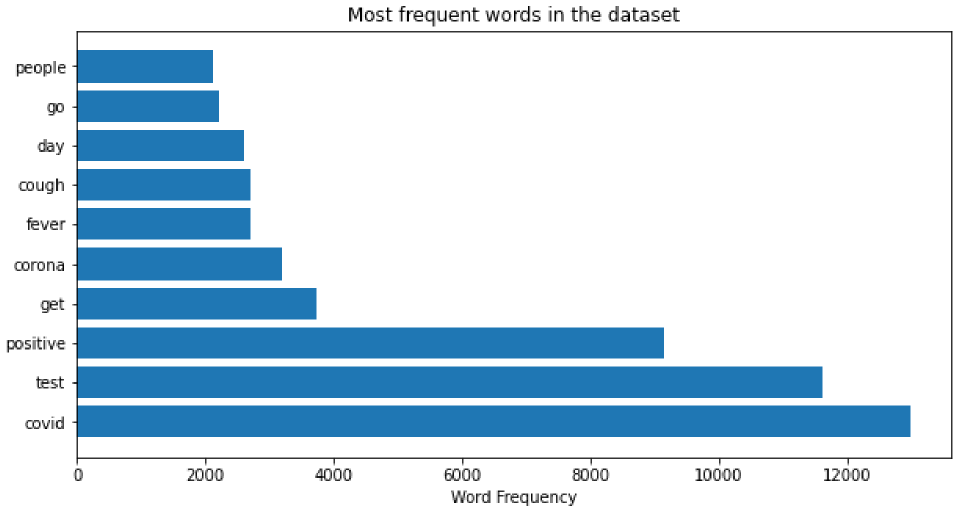


Figure 3.1: The 10 most frequent words in dataset

The Python library, for topic modelling in NLP, Gensim[11] has a feature that presents the most similar words to a given term. It computes the cosine similarity between a simple

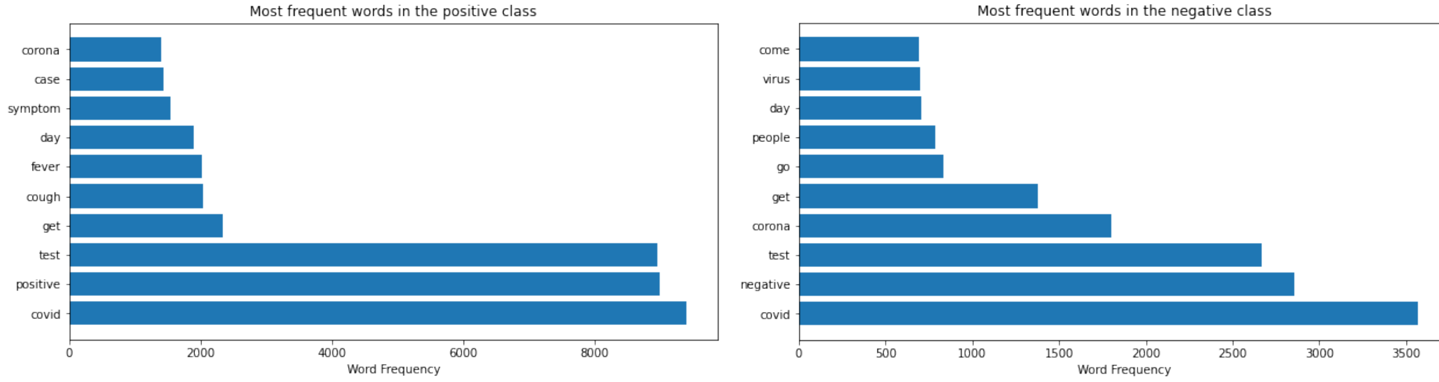
[11] Official website: `https://radimrehurek.com/gensim/`

mean of the projection weight vectors of the given word and the vectors for each word in the model. The words most similar to “Covid” are shown in Table 3.1.

Word	**Similarity**
rapid	0.7554
antibody	0.7390
inconclusive	0.7301
pcr	0.7294
corona	0.7160
retested	0.7048

Table 3.1: Similarity level with the term *covid*

The same is shown in Figures 3.2a and 3.2b for the classes labeled as positive and negative, respectively.



(a) Positive

(b) Negative

Figure 3.2: The 10 most frequent words for the positive and negative classes

The tweets were filtered based on symptoms that were declared by the persons that published the tweets. The 5 symptoms that appeared the most on the tweets were *fever, cough, breathe, taste and smell* as presented in Figure 3.3.

Were also checked the bi-grams of the the entire dataset and of the positive and negative classes . Bi-grams are a sequence of two elements of a set of tokens. The tokens in this work represent words. The bi-grams frequency for the entire dataset are show in Figure **??**. According to Figure 3.4b, the most frequent bi-grams that appeared in the positive class were *(test,positive), (positive,covid), (test,covid), (covid,positive), etc.*

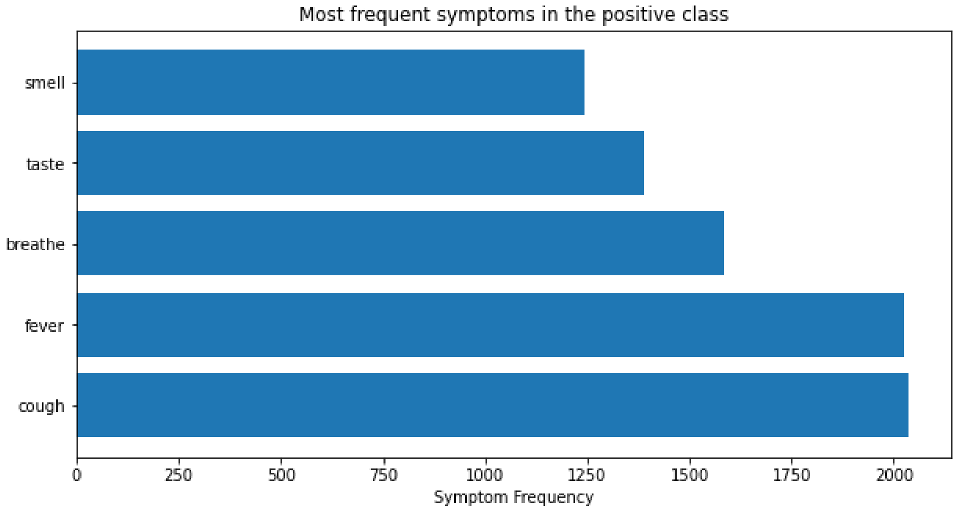


Figure 3.3: Symptoms that appeared the most in the positive class

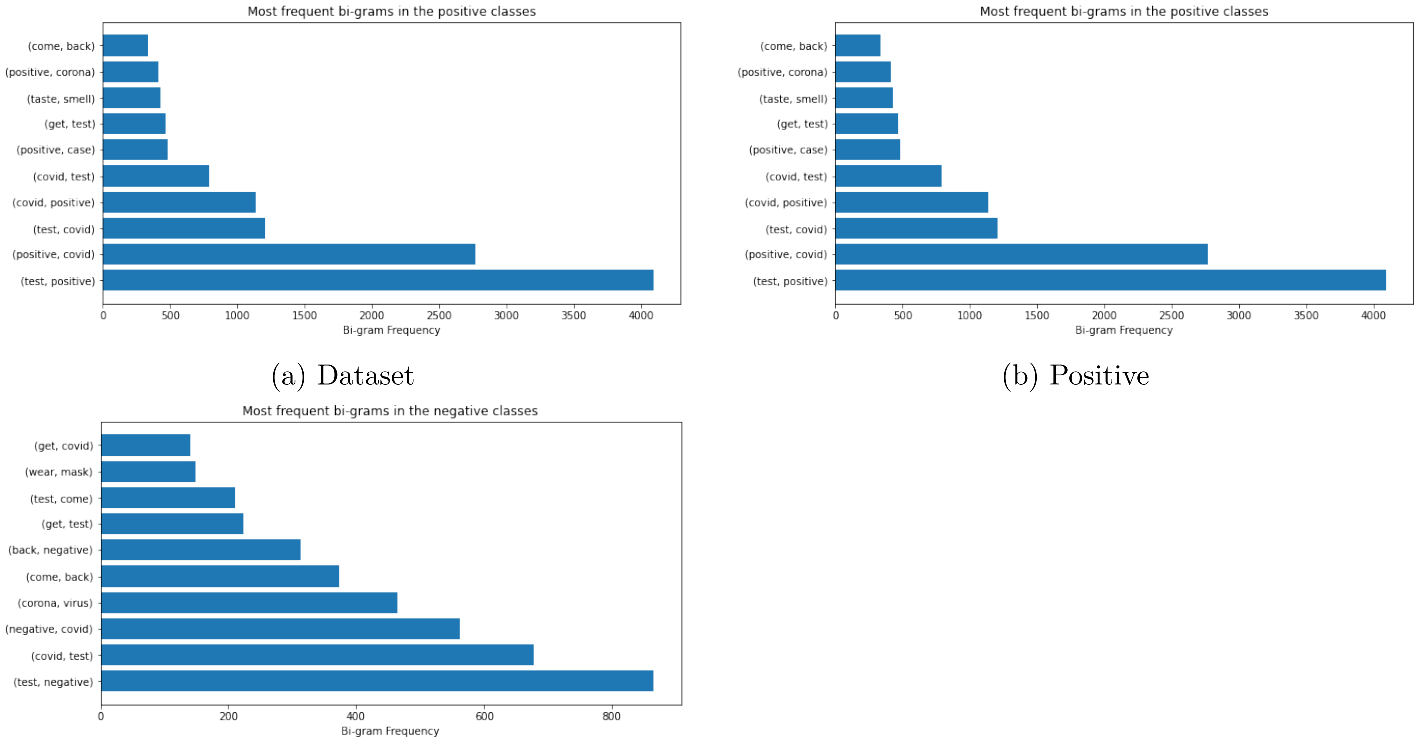
For the negative class, the most frequent bi-grams that appeared were *(test,negative), (covid,test), (negative,covid), (corona,virus), etc,* and are shown in Figure 3.4c.

3.3 Training

The dataset was divided in training and testing, using 70% and 30% of the original data, respectively. The dataset was used to train and test KNN, Naive Bayes, Decision Tree, Random Forest, SVM, MLP, CNN and BERT models.

In the paper done by Hsu [96], comparisons were made between 8 supervised models for text classification, including the models MLP, Naive Bayes, SVM, Decision Tree and Random Forest. For the same models, the hypeparameters that obtained the best results were also adopted. Using textual data, the hyperparameters configuration that performed best in MLP had hidden layers with 100 neurons and *activation=ReLU*. In SVM, the regularization parameter was set to $C = 4$ and the tolerance for stopping criterion was set to $tol = 1e - 04$. In Decision Tree it was used *criterion=entropy*, and in Random Forest was used $n\_estimators = 200$ and also *criterion=entropy*.

For CNN the configuration of the hyperparameters was according to the work of Krishnakumari [97] which compared different hyperparameters for document classification.



(a) Dataset

(b) Positive

(c) Negative

Figure 3.4: Most frequent bi-grams in the whole dataset and in the positive and negative instances

The performance values in the results varied according to the type of hyperparameter. The size of the filters were 3, 4 and 5, the dropout rate was 0.5, the feature map was 128 and the Softmax activation function was used. The architecture of the CNN is represented in Figure 3.5. In the Figure, the input is represented as a matrix of size 7x5 as the author used as an example. The size of the input matrix in the paper should be set as dxl where d is the dimension of the embedding and l the size of the sentence. For this work, we used embedding dimension of 128 was used.

The BERT model was obtained from a course available on the online educational platform Udemy [12], and taught by Professor Jones Granatyr [13] that made an implementation of the official BERT model described in Chapter 2. As the model implemented by Granatyr was used to question answering, some changes were made on the output in order to return only the value that should be the classification of the model.

[12] https://www.udemy.com/

[13] https://iaexpert.academy/teacher/jones-granatyr/

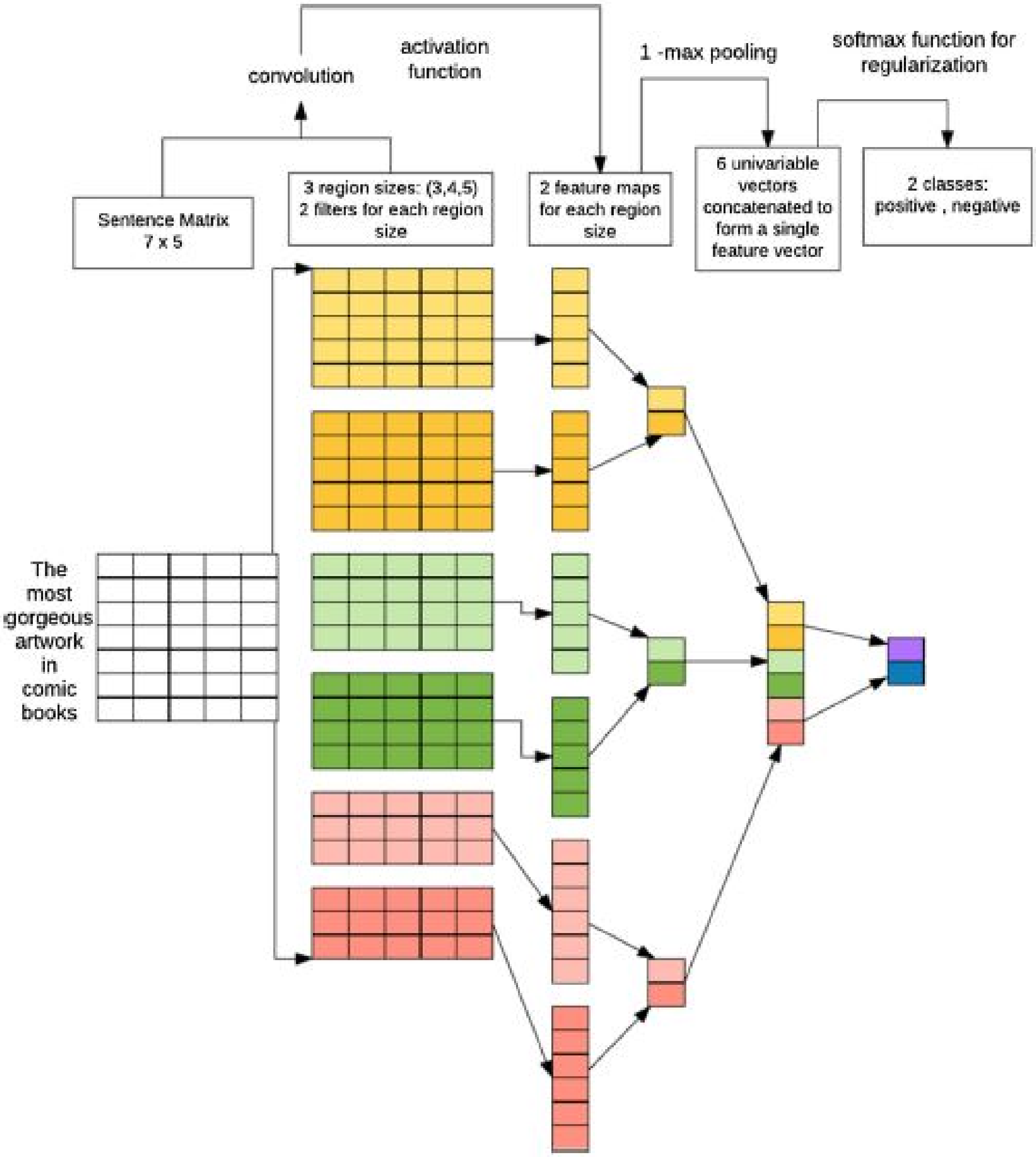

convolution
activation function
1 -max pooling
softmax function for regularization
Sentence Matrix 7 x 5
3 region sizes: (3,4,5) 2 filters for each region size
2 feature maps for each region size
6 univariable vectors concatenated to form a single feature vector
2 classes: positive , negative
The most gorgeous artwork in comic books

Figure 3.5: Proposed CNN architecture [97]

Chapter 4

Results and Discussion

This chapter will present the results of the tests performed according to what was described in Chapter 3 with a discussion about the results.

It is important to mention that the computational cost was low for this amount of data and did not present much difference between the models. The time spent in the training was less than 4 minutes for all the models.

4.1 Analysis of Results

To evaluate the results, confusion matrices were obtained from each model after the tests. A confusion matrix shows the classification frequencies for each class of the model, presenting the amount of True Positives, False Positives, True Negatives and False Negatives, whose meaning are in the following:

- True Positive is when the model classifies an instance as positive and the real class is positive;

- False Positive is the classification of an instance as positive but its real class is negative;

- True Negative is the classification of an instance as negative and the real class is negative;

- False Negative is the classification of an instance as negative but its real class is positive.

In addition to this, were calculated and compared the accuracy, precision, recall and F-Measure of all models tested. The meaning of these metrics is as follows:

- Precision quantifies the number of positive class predictions that actually belong to the positive class. To calculate the precision of the model, the following formula is used: $Precision = \frac{TruePositives}{(TruePositives+FalsePositives)}$.
- Recall quantifies the number of positive class predictions made out of all positive examples in the dataset. To calculate the recall of the model, the following formula is used: $Recall = \frac{TruePositives}{(TruePositives+FalseNegatives)}$.
- F-Measure provides a single score that balances both the concerns of precision and recall in one number. To calculate the F-Measure of the model, the following formula is used: F-Measure $= \frac{(2*Precision*Recall)}{(Precision+Recall)}$.
- Accuracy presents an overall performance of the model, among all instances, how many of them were correctly classified. To calculate the accuracy of the model, it is used the following formula: $Accuracy = \frac{TruePositive+TrueNegative}{TruePositive+TrueNegative+FalsePositive+Falsenegative}$

4.2 Confusion Matrices

The results obtained with the tests performed for each model on the dataset are presented in Table 4.1, where TP, TN, FP and FN are the short form of True-Positive, True-Negative, False-Positive and False-Negative respectively. In this table it is possible to observe that KNN has the greatest number of incorrect classifications of all the models when classifying as False-Positive. However, Naive Bayes has greatest amount of instances classified as False-Negative. When both the False-Positive and False-Negative are summed in all the models these two present the greatest number of incorrect classification.

Model	**TP**	**TN**	**FP**	**FN**
KNN	1709	2592	**2285**	312
Naive Bayes	3614	1833	380	**1071**
Decision Tree	3732	2675	262	229
Random Forest	**3801**	2706	193	198
SVM	3695	2726	299	178
MLP	3641	2609	353	295
CNN	3721	2693	273	211
BERT	3699	**2734**	295	170

Table 4.1: Confusion matrices table of all models

Table 4.1 also shows that regarding the correct classifications, Random Forest has the greatest number of correct predictions when classifying the instances as Positive. Although this model presents a high number of hits in the True-Negative classification, compared to the other models, BERT presents best number of correct prediction for the negative instances. Nevertheless, the models with the best results for both the True-Positive and True-Negative classification do not have a huge difference between them.

As KNN and Naive Bayes have the worst results in FP and FN, respectively, they also present the worst results when classifying correctly the instances: KNN has the smallest number of TP of all models while Naive Bayes presents the worst number of TN.

4.3 Metrics

In Table 4.1 we presented the values of the confusion matrices of all models. From these initial values we have already pointed out some observations about the models that presented the best results in number of correct and incorrect classifications. Based on the values of Table 4.1, it was possible to calculate the Precision, Recall, F-Measure and Accuracy of the models in order to have more measures to evaluate their performance.

All the metric values calculated for each model are presented in Table 4.2 in percentage format.

Model	Precision	Recall	F-Measure	Accuracy
KNN	84.6%	42.8%	56.8%	62.3%
Naive Bayes	77.1%	90.4%	83.2%	79.0%
Decision Tree	94.2%	93.4%	93.8%	92.9%
Random Forest	95.0%	**95.1%**	**95.2%**	**94.3%**
SVM	95.4%	92.5%	93.9%	93.1%
MLP	92.5%	91.2%	91.9%	90.6%
CNN	94.6%	93.2%	93.9%	93.0%
BERT	**95.6%**	92.6%	94.0%	93.3%

Table 4.2: Metrics of all models tested

4.3.1 Precision

Precision deals with the amount of variation that arises from a set of measurements performed. The more precise a measurement, the smaller the variability between the values obtained. Therefore, the precision of the analyzed models will not check the overall correctness of each one, which should be measured by the accuracy, but rather the degree of variation and correctness in the positive class.

According to Table 4.2 it is possible to see that BERT, SVM and Random Forest are the models that present the highest precision in the tests, with BERT having the maximum (95.6%). This suggests that there is not a large variation in the test results for the models with the highest precision, and that these are the models that present the greatest correctness in classifying the instances of the positive class. It is also possible to verify that the Naive Bayes algorithm is the one that present the lowest precision among all the models tested. This means, that of all the instances classified as positive, it behaved the worst, obtaining a precision of only 77.1%.

4.3.2 Recall

Recall deals with the frequency that the model finds instances of a class. This measure will verify when an instance is actually of the class being observed.

Of the models analyzed in test, and arranged in Table 4.2 shows also the recall of all models tested. The one with the highest recall was the Random Forest algorithm which achieved 95.1% (of the instances that this algorithm classified as positive 95.1% were expected to belong to this class). On the other hand, the KNN algorithm is the one that presents the worst result, with only 42.8%.

In the KNN algorithm, similar data tends to be concentrated in the same region of the data scatter space. In the dataset, is used some tweets were found to have reports of Covid-19 symptoms but normally used to jokes or sarcasms. These tweets were classified in the dataset as negative. In this case, the algorithm may have grouped words that represent a Covid-19 symptom to classify the instances as positive class.

4.3.3 F-Measure

It is represented as the harmonic mean of the Precision and Recall, seeking to bring a number that indicates the overall quality of a model. F-Measure values can range from 0.0 to 1.0 which indicates the perfection of the precision and recall results.

Among the models analyzed in the tests, the one that presents the highest F-Measure is Random Forest (95.2%), which indicates a better balance between precision and recall. On the other hand, the model that presents a much lower result than the others is KNN, which obtained an F-Measure of only 42.8%. This result is due to the imbalance between the accuracy of this model (which presented a value of 84.6%), and its recall (that obtained 42.8%), while in the other models the difference between them was not so high.

4.3.4 Accuracy

Accuracy gives an overall result of how the model performed during testing. It typically serves to answer the question of what is the overall success rate of the model. In this case, Table 4.2 presents also the result of the accuracy of all models. It is possible to verify that the model that exhibits the highest accuracy of all those tested is Random Forest with 94.3%, and the one with obtained the lowest accuracy is KNN with just 62.3%.

It is also possible to note that BERT, SVM and CNN have very close results, all within the 93% accuracy range, with a slight superiority for the BERT model, which is the second best model.

4.4 Best Results

For the dataset used, and taking into consideration the relatively small size, the Random Forest algorithm was the one that presented the best results among all the models analyzed in this investigation. BERT and CNN models are more robust than Random Forest, but may suffer from over-fitting when the training data is too small, as exemplified by Pasupa [98], where CNN models with multiple hidden layers were compared with those with shallower layers and also SVM. It was concluded that CNN models with fewer hidden layers and SVM performed better than those with deeper layers for a small dataset.

Although Random Forest performed better than BERT, CNN and SVM, they all showed metrics very close to each other and could switch their order of performance ranking if the dataset were larger or the hyperparameters of the models were modified.

Chapter 5

Conclusions and Future works

The goal of this work was to analyze and compare machine learning and deep learning models for the classification of text, more specifically Covid-19 related tweets, where the content of the text presented indications of symptoms of the disease or self-reported contamination. After the end of the tests it was possible to achieve the general objective of studying and applying machine learning models to identify symptoms and text with self-diagnostic content in social networks, as well as the the specific objectives listed in Chapter 1.

This work was important for understanding how different models work for text classification in a small dataset and also for the analysis and identification of symptoms of diseases that may appear in social media texts.

Unlike what might be expected, the result of the tests proved to be a surprise, as Random Forest managed to outperform models that currently pose as state-of-the-art in Natural Language Processing tasks such as BERT [86], or even CNN.

5.1 Challenges during the project

The biggest challenge in performing the work was obtaining and labeling the dataset. The dataset from which the tweets for training and testing were obtained contained only texts that referenced Covid-19, as explained in Chapter 3, but in the process of filtering the relevant tweets, some were not adequately filtered, which required a further finer filtering.

In addition, requesting tweets from the internet took more time than expected, as after a number of requests it was necessary to wait for a period of time for the tool to work again.

Also, the cleaning of the tweets was a challenge: some emoticons did not have the hexadecimal codes available to allow automated removal, and so this had to be done manually.

5.2 Future Directions

In this work, different hyperparameters for each model were not tested, which could improve the performance of the algorithms and bring better practical results. Therefore, for future work, there is the possibility of testing the models in this work, together or individually, with different types of hyperparameters to verify which combinations of hyperparameters provide the best results.

For this work, the collected dataset had a small size. Probably more data was generated and is available since the dataset was collected. With more data available the dataset could be increased and further analysis be made.

One approach used by Pasupa [98] in the treatment of emoticons was the replacement with the respective sentiment from a dictionary of emoticons. This kind of approach can be checked during the cleaning of the dataset and analyzed to check if provides any improvement in the results.

Another possibility for future work is to use other information that is returned by Twitter in the “hydration” process. During the data request, various information is returned in a JSON format for each tweet, and within this information is the region from where the tweet was posted. With this, it is possible to make a study of the relationship between regions and certain diseases or symptoms.

Bibliography

[1] M. A. Al-Garadi, Y.-C. Yang, S. Lakamana, and A. Sarker, "A text classification approach for the automatic detection of twitter posts containing self-reported covid-19 symptoms", 2020.

[2] G. Eysenbach, "Infodemiology and infoveillance: Framework for an emerging set of public health informatics methods to analyze search, communication and publication behavior on the internet", *Journal of medical Internet research*, vol. 11, no. 1, e11, 2009.

[3] M. A. Al-Garadi, M. S. Khan, K. D. Varathan, G. Mujtaba, and A. M. Al-Kabsi, "Using online social networks to track a pandemic: A systematic review", *Journal of biomedical informatics*, vol. 62, pp. 1–11, 2016.

[4] P. Kathiravan, S. Makila, H. Prasanna, and P. Vimala, "Over View- The Machine Translation in NLP", vol. 2, no. 7, 2016.

[5] J. Carter, V. Saunders, and V. A. Saunders, *Virology: principles and applications.* John Wiley & Sons, 2007.

[6] A. M. King, M. J. Adams, E. B. Carstens, and E. J. Lefkowitz, "Virus taxonomy", *Ninth report of the International Committee on Taxonomy of Viruses*, pp. 486–487, 2012.

[7] L. Bruslind, *The viruses.* [Online]. Available: `https://open.oregonstate.education/generalmicrobiology/chapter/the-viruses/`.

[8] W. Levinson, P. Chin-Hong, E. A. Joyce, J. Nussbaum, and B. Schwartz, *Review of medical microbiology and immunology*. McGraw-Hill Medical Estados Unidos, 2008.

[9] J. T. Patton, *Segmented double-stranded RNA viruses: structure and molecular biology*. Horizon Scientific Press, 2008.

[10] J. A. Levy, *The retroviridae*. Springer Science & Business Media, 2013.

[11] L. D. Kramer, *Visão geral dos vírus - doenças infecciosas*. [Online]. Available: `https://www.msdmanuals.com/pt-pt/profissional/doen%C3%A7as-infecciosas/v%C3%ADrus/vis%C3%A3o-geral-dos-v%C3%ADrus`.

[12] T. Singhal, "A review of coronavirus disease-2019 (COVID-19)", *The Indian Journal of Pediatrics*, vol. 87, no. 4, pp. 281–286, Apr. 1, 2020, ISSN: 0973-7693. DOI: `10.1007/s12098-020-03263-6`. [Online]. Available: `https://doi.org/10.1007/s12098-020-03263-6` (visited on 03/06/2021).

[13] G. Li, Y. Fan, Y. Lai, T. Han, Z. Li, P. Zhou, P. Pan, W. Wang, D. Hu, X. Liu, Q. Zhang, and J. Wu, "Coronavirus infections and immune responses", *Journal of Medical Virology*, vol. 92, no. 4, pp. 424–432, Apr. 2020, ISSN: 0146-6615. DOI: `10.1002/jmv.25685`. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7166547/` (visited on 03/06/2021).

[14] W. H. Organization, *Coronavirus*. [Online]. Available: `https://www.who.int/health-topics/coronavirus#tab=tab_3`.

[15] *Vacinacao - covid-19*. [Online]. Available: `https://covid19.min-saude.pt/vacinacao/`.

[16] M. van Algemene Zaken, *Order of vaccination for people who do not work in healthcare*. [Online]. Available: `https://www.government.nl/topics/coronavirus-covid-19/dutch-vaccination-programme/order-of-vaccination-against-coronavirus/order-of-vaccination-for-people-who-do-not-work-in-healthcare`.

[17] *Vacina já: Governo do estado de são paulo.* [Online]. Available: `https://vacinaja.sp.gov.br/`.

[18] X. Shen, H. Tang, C. McDanal, K. Wagh, W. Fischer, J. Theiler, H. Yoon, D. Li, B. F. Haynes, K. O. Sanders, *et al.*, "Sars-cov-2 variant b. 1.1. 7 is susceptible to neutralizing antibodies elicited by ancestral spike vaccines", *Cell Host & Microbe*, 2021.

[19] A. Fontanet, B. Autran, B. Lina, M. P. Kieny, S. S. A. Karim, and D. Sridhar, "Sars-cov-2 variants and ending the covid-19 pandemic", *The Lancet*, 2021.

[20] N. G. Davies, S. Abbott, R. C. Barnard, C. I. Jarvis, A. J. Kucharski, J. D. Munday, C. A. Pearson, T. W. Russell, D. C. Tully, A. D. Washburne, *et al.*, "Estimated transmissibility and impact of sars-cov-2 lineage b. 1.1. 7 in england", *Science*, 2021.

[21] G. Iacobucci, "Covid-19: New uk variant may be linked to increased death rate, early data indicate", *bmj*, vol. 372, p. 230, 2021.

[22] J. W. Tang, O. T. Toovey, K. N. Harvey, and D. D. Hui, "Introduction of the south african sars-cov-2 variant 501y. v2 into the uk", *The Journal of infection*, 2021.

[23] C. K. Wibmer, F. Ayres, T. Hermanus, M. Madzivhandila, P. Kgagudi, B. Oosthuysen, B. E. Lambson, T. de Oliveira, M. Vermeulen, K. van der Berg, *et al.*, "Sars-cov-2 501y. v2 escapes neutralization by south african covid-19 donor plasma", *Nature medicine*, pp. 1–4, 2021.

[24] A. Lopez-Rincon, C. Perez-Romero, A. Tonda, L. Mendoza-Maldonado, E. Claassen, J. Garssen, and A. D. Kraneveld, "Design of specific primer sets for the detection of b. 1.1. 7, b. 1.351 and p. 1 sars-cov-2 variants using deep learning", *bioRxiv*, 2021.

[25] G. A. Poland, I. G. Ovsyannikova, S. N. Crooke, *et al.*, "Sars-cov-2 vaccine development: Current status", in *Mayo Clinic Proceedings*, Elsevier, 2020.

[26] F. Amanat and F. Krammer, "Sars-cov-2 vaccines: Status report", *Immunity*, vol. 52, no. 4, pp. 583–589, 2020.

[27] W. H. Organization, *Coronavirus disease (covid-19): Vaccines*, Oct. 2020. [Online]. Available: https://www.who.int/news-room/q-a-detail/coronavirus-disease-(covid-19)-vaccines?adgroupsurvey=%7Badgroupsurvey%7D&gclid=CjwKCAiAhbeCBhBcEiwAkv2cY_ij4pce_wCZuKfU4o-bkMJpkPyMPjSUH0b93ZVumjv8q0WY4H7CtxoCKHk BwE.

[28] Aadhityan A, "A novel method for implementing artificial intelligence, cloud and internet of things in robots", in *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2015, pp. 1–4. DOI: 10.1109/ICIIECS.2015.7193238.

[29] R. J. Schalkoff, *Artificial intelligence: an engineering approach*. McGraw-Hill New York, 1990.

[30] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[31] M. Campbell, A. Hoane, and F.-h. Hsu, "Deep blue", *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, 2002, ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(01)00129-1. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370201001291.

[32] F.-Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, "Where does alphago go: From church-turing thesis to alphago thesis and beyond", *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 2, pp. 113–120, 2016.

[33] V. Kaul, S. Enslin, and S. A. Gross, "The history of artificial intelligence in medicine", *Gastrointestinal endoscopy*, 2020.

[34] M. VAZIRGIANNIS, "Artificial intelligence–challenges for the future", 2020.

[35] J. Rodrigues. (2017). O que é o processamento de línguagem natural, [Online]. Available: https://medium.com/botsbrasil/o-que-%C3%A9-o-processamento-de-linguagem-natural-49ece9371cff.

[36] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very deep convolutional networks for text classification”, *arXiv preprint arXiv:1606.01781*, 2016.

[37] Y. Wu, H. Mao, and Z. Yi, “Audio classification using attention-augmented convolutional neural network”, *Knowledge-Based Systems*, vol. 161, pp. 90–100, 2018.

[38] X. Zhang and Y. LeCun, “Which encoding is the best for text classification in chinese, english, japanese and korean?”, *arXiv preprint arXiv:1708.02657*, 2017.

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

[40] T. M. Mitchell, *Machine Learning*, 1st ed. USA: McGraw-Hill, Inc., 1997, ISBN: 0070428077.

[41] H. Honda, M. Facure, and P. Yaohao, *Os três tipos de aprendizado de máquina*, Jul. 2017. [Online]. Available: `https://lamfo-unb.github.io/2017/07/27/tres-tipos-am/`.

[42] J. C. d. Silva, *Algoritmos de aprendizagem de máquina: Qual deles escolher?*, Mar. 2018. [Online]. Available: `https://medium.com/machina-sapiens/algoritmos-de-aprendizagem-de-m%C3%A1quina-qual-deles-escolher-67040ad68737`.

[43] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning”, in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 157–163.

[44] Y. Wang and Z.-O. Wang, “A fast knn algorithm for text categorization”, in *2007 International Conference on Machine Learning and Cybernetics*, IEEE, vol. 6, 2007, pp. 3436–3441.

[45] I. Jose, *Knn (k-nearest neighbors)*, Sep. 2018. [Online]. Available: `https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e`.

[46] J. Joyce, “Bayes’ theorem”, in *Stanford Encyclopedia of Philosophy*, 2008.

[47] L. Breiman, “Random forests”, *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[48] P. O. Gislason, J. A. Benediktsson, and J. R. Sveinsson, “Random forests for land cover classification”, *Pattern recognition letters*, vol. 27, no. 4, pp. 294–300, 2006.

[49] R. Gandhi, *Support vector machine - introduction to machine learning algorithms*, Jul. 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[50] C. Huang, L. Davis, and J. Townshend, “An assessment of support vector machines for land cover classification”, *International Journal of remote sensing*, vol. 23, no. 4, pp. 725–749, 2002.

[51] O. Chapelle, P. Haffner, and V. N. Vapnik, “Support vector machines for histogram-based image classification”, *IEEE transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[52] D. Mahmoodi, H. Marvi, M. Taghizadeh, A. Soleimani, F. Razzazi, and M. Mahmoodi, “Age estimation based on speech features and support vector machine”, Jul. 2011. DOI: 10.1109/CEEC.2011.5995826.

[53] D. J. S. Ribeiro, “Support vector machines na previsão do comportamento de uma etar”, PhD thesis, 2012.

[54] G. Palm, “Warren McCulloch and walter pitts: A logical calculus of the ideas immanent in nervous activity”, in *Brain Theory*, G. Palm and A. Aertsen, Eds., Berlin, Heidelberg: Springer, 1986, pp. 229–230, ISBN: 978-3-642-70911-1. DOI: 10.1007/978-3-642-70911-1_14.

[55] T. H. Abraham, “(physio) logical circuits: The intellectual origins of the mcculloch–pitts neural networks”, *Journal of the History of the Behavioral Sciences*, vol. 38, no. 1, pp. 3–25, 2002.

[56] G. L. Shaw, “Donald hebb: The organization of behavior”, in *Brain Theory*, G. Palm and A. Aertsen, Eds., Berlin, Heidelberg: Springer, 1986, pp. 231–233, ISBN: 978-3-642-70911-1. DOI: 10.1007/978-3-642-70911-1_15.

[57] C. Van Der Malsburg, “Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms”, in *Brain Theory*, G. Palm and A. Aertsen,

Eds., Berlin, Heidelberg: Springer, 1986, pp. 245–248, ISBN: 978-3-642-70911-1. DOI: `10.1007/978-3-642-70911-1_20`.

[58] M. Minsky and S. Papert, "An introduction to computational geometry", *Cambridge tiass., HIT*, 1969.

[59] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition", *Computer*, vol. 21, no. 3, pp. 25–39, 1988.

[60] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation", *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[61] I. M. Quintanilha, "End-to-end speech recognition applied to brazilian portuguese using deep learning", *Ph. D. dissertation, MSc dissertation*, 2017.

[62] T. W. Rauber, "Redes neurais artificiais", *Universidade Federal do Espírito Santo*, 2005.

[63] S. Haykin, *Redes neurais: princípios e prática*. Bookman Editora, 2007.

[64] T. M. Leite. (2018). Redes neurais, perceptron multicamadas e o algoritmo backpropagation, [Online]. Available: `https://medium.com/ensina-ai/redes-neurais-perceptron-multicamadas-e-o-algoritmo-backpropagation-eaf89778f5b8` (visited on 05/10/2018).

[65] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm", in *Proceedings of the IEEE international conference on neural networks*, San Francisco, vol. 1993, 1993, pp. 586–591.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[67] G. S. d. S. Gomes and T. B. Ludermir, "Redes neurais artificiais com funções de ativação complemento log-log e probit para aproximar funções na presença de observações extremas", *Revista da Sociedade Brasileira de Redes Neurais (SBRN)*, vol. 6, no. 2, pp. 142–153, 2008.

[68] H. Demuth, M. Beale, and M. Hagan, “Neural network toolbox”, *For Use with MATLAB. The MathWorks Inc*, vol. 2000, 1992.

[69] B. Karlik and A. V. Olgac, “Performance analysis of various activation functions in generalized mlp architectures of neural networks”, *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.

[70] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks”, in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[71] J. Schmidt-Hieber *et al.*, “Nonparametric regression using deep neural networks with relu activation function”, *Annals of Statistics*, vol. 48, no. 4, pp. 1875–1897, 2020.

[72] S. Sharma, *Activation functions in neural networks*, Feb. 2019. [Online]. Available: `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`.

[73] D. S. Academy. (2019). Deep learning book, [Online]. Available: `http://deeplearningbook.com.br/algoritmo-backpropagation-parte-2-treinamento-de-redes-neurais/`.

[74] A. L. Koerich, “Aprendizagem de máquina”, PhD thesis, Pontificia Universidade Católica do Paraná, 1999.

[75] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences”, *arXiv preprint arXiv:1404.2188*, 2014.

[76] M. Z. Amin and N. Nadeem, “Convolutional neural network: Text classification model for open domain question answering system”, *arXiv preprint arXiv:1809.02479*, 2018.

[77] K. O’Shea and R. Nash, “An introduction to convolutional neural networks”, *arXiv preprint arXiv:1511.08458*, 2015.

[78] G. Alves, *Entendendo redes convolucionais (cnns)*, Dec. 2018. [Online]. Available: `https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184`.

[79] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network", in *2017 International Conference on Engineering and Technology (ICET)*, Ieee, 2017, pp. 1–6.

[80] A. M. Clappis, *Uma introdução as redes neurais convolucionais utilizando o keras*, Jul. 2019. [Online]. Available: `https://medium.com/data-hackers/uma-introdu%C3%A7%C3%A3o-as-redes-neurais-convolucionais-utilizando-o-keras-41ee8dcc033e`.

[81] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need", *arXiv preprint arXiv:1706.03762*, 2017.

[82] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", pp. 1–15, Sep. 2014. arXiv: `1409.0473`. [Online]. Available: `http://arxiv.org/abs/1409.0473`.

[83] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 5884–5888.

[84] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization", *arXiv preprint arXiv:1607.06450*, 2016.

[85] O. Press and L. Wolf, "Using the output embedding to improve language models", *arXiv preprint arXiv:1608.05859*, 2016.

[86] *Open sourcing bert: State-of-the-art pre-training for natural language processing*, Nov. 2018. [Online]. Available: `https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html`.

[87] W. L. Taylor, "“cloze procedure”: A new tool for measuring readability", *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[88] Y. Jernite, S. R. Bowman, and D. A. Sontag, "Discourse-based objectives for fast unsupervised sentence representation learning", *CoRR*, vol. abs/1705.00557, 2017. arXiv: `1705.00557`. [Online]. Available: `http://arxiv.org/abs/1705.00557`.

[89] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[90] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning", in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[91] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[92] E. Bisong, "Google colaboratory", in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, 2019, pp. 59–64.

[93] R. Lamsal, *Coronavirus (covid-19) tweets dataset*, 2020. DOI: `10.21227/781w-ef42`. [Online]. Available: `https://dx.doi.org/10.21227/781w-ef42`.

[94] A. Sarker, S. Lakamana, W. Hogg-Bremer, A. Xie, M. A. Al-Garadi, and Y.-C. Yang, "Self-reported covid-19 symptoms on twitter: An analysis and a research resource", *Journal of the American Medical Informatics Association*, vol. 27, no. 8, pp. 1310–1315, 2020.

[95] W. J. Wilbur and K. Sirotkin, "The automatic identification of stop words", *Journal of information science*, vol. 18, no. 1, pp. 45–55, 1992.

[96] B.-M. Hsu, "Comparison of supervised classification models on textual data", *Mathematics*, vol. 8, no. 5, p. 851, 2020.

[97] K. Krishnakumari, E. Sivasankar, and S. Radhakrishnan, "Hyperparameter tuning in convolutional neural networks for domain adaptation in sentiment classification (htcnn-dasc)", *Soft Computing*, vol. 24, no. 5, pp. 3511–3527, 2020.

[98] K. Pasupa and W. Sunhem, "A comparison between shallow and deep architecture classifiers on small dataset", in *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, IEEE, 2016, pp. 1–6.

Appendix A

Implemented Code

Code implemented in Python to filter tweets by symptoms and remove emoticons using regular expressions.

```
import re
import os
import pandas as pd

def get_list_of_files(folder_path='./'):
    list_of_files=[]
    for _, _, files in os.walk(folder_path):
        list_of_files.extend(files)
    return [l for l in list_of_files if '.txt' in l]

def deEmojify(text):
    regrex_pattern = re.compile(pattern = "["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u”\U00012702−\U000127B0”  # Dingbats
        u”\U000124C2−\U0001F251”  # Enclosed characters
        u”\U0001F30D−\U0001F567”  # Other Aditional Symbols
        u”\U0001F681−\U0001F6C5”  # transport & map symbols
        u”\U0001F600−\U0001F636”  # Additional Emoticons
        u”\U00010000−\U0001FFFF”  # Additional Emoticons

                           ”]+”, flags = re.UNICODE)
    return regrex_pattern.sub(r’’,text)

def get_txt_file(file_name, path=’./’):
    if ’.txt’ not in file_name:
        file_name = file_name.strip() + ’.txt’
    file_to_open = path.strip()+file_name
    with open(file_to_open, ’r’, encoding=’utf8’) as source:
        tweets = source.read()
    tweets_list = [i.strip() for i in tweets.split(’\n’)]
    tweets_with_id = []
    for t in tweets_list:
        t = t.replace(’[’, ’’).lstrip()
        t = t.replace(’]’, ’’).rstrip()
        id_tweet = t[:19]
        tweet_itself = t[20:]
        tweets_with_id.append([id_tweet, tweet_itself])
    return tweets_with_id

def re_text(texts):
    symptoms = [’cough’, ’fever’, ’breath’, ’breathing’, ’taste’, ’diarrhea
    pronoumns=[’i’, ’we’, ’us’, ’my’, ’her’, ’his’]
    annotated_positive_texts = []
    annotated_negative_texts = []
    for tweet in texts:
        tweet_id = tweet[0]
        tweet_text = deEmojify(tweet[-1])
        if re.findall('positive', tweet_text.lower()):
            for s in symptoms:
                if s in tweet_text:
                    for p in pronoumns:
                        if p in tweet_text and [tweet_id, tweet_text, '1'] n
                            annotated_positive_texts.append([tweet_id, twee
        else:
            annotated_negative_texts.append([tweet_id, tweet_text, '0'])
    return annotated_positive_texts, annotated_negative_texts



def save_csv(list_file, file_name, path='./'):
    columns = ['ids', 'text', 'annotation']
    dt = pd.DataFrame(list_file, columns = columns)
    dt  = dt.drop_duplicates(subset='text', keep='first')
    file_name = file_name + '.csv'
    dt.to_csv(file_name, sep='\t', columns=columns, index=False)
    print('File␣saved')


path_to_save = './annotated/'
```

```
for file_item in get_list_of_files('./hydrated_json'):
    print(file_item)
    texts = get_txt_file(file_item, './hydrated_json/')
    pos, neg = re_text(texts)
    name = file_item.replace('.txt','').strip()
    save_csv(pos, 'positive_'+file_item, path_to_save)
    save_csv(neg, 'negative_'+file_item, path_to_save)
```