UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MASTER

Design and Development of a Mental Illness Recognition System for Detecting Stress and Depression from Text

Nicolás López Cano
2021

TRABAJO DE FIN DE MASTER

Título:	Diseño y desarrollo de un sistema de reconocimiento de trastornos mentales para detectar estrés y depresión a partir de un texto
Título (inglés):	Design and Development of a Mental Illness Recognition System for Detecting Stress and Depression from Text
Autor:	Nicolás López Cano
Tutor:	Sergio Muñoz López
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

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

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MASTER

Design and Development of a Mental Illness Recognition System for Detecting Stress and Depression from Text

Julio 2021

Resumen

La salud mental se está convirtiendo en un punto crucial de nuestra sociedad. La depresión es el trastorno mental más frecuente del mundo y su incidencia aumenta cada año. Análogamente, el estrés está cada vez más presente y de forma más permanente en nuestro día a día. Esto perjudica a los que lo padecen tanto en su salud, como a nivel social o laboral.

El objetivo de este proyecto consiste en ayudar en una de las tareas cruciales para la mejora de la salud mental, la detección precoz. El proceso de diagnóstico actual requiere que el paciente acuda al médico para realizar los cuestionarios necesarios o hacerse análisis. Esto puede ser un problema dado el estigma social que existe en torno a los trastornos mentales y a las personas que lo padecen. No todos los pacientes que deberían, acuden al médico ya sea por miedo, vergüenza o desconocimiento.

Por todo lo expuesto, se ha desarrollado un sistema de reconocimiento de depresión y estrés en texto, basado en tecnologías de aprendizaje automático y procesado de lenguaje natural. El sistema está formado por un clasificador, encargado de hacer la predicción de los trastornos mentales y un plugin de Mattermost, el cual permite que el texto escrito por los usuarios en una herramienta de comunicación mediante texto, sea evaluado por el clasificador.

Esto permite detectar antes situaciones de estrés laboral o de depresión de algún trabajador. Además, en caso de que el clasificador detecte algún signo, permite enviar un mensaje de forma automática por el canal donde está el usuario para recomendarle que se tome un descanso, con el objetivo de que reduzca su estrés. Un sistema que sea capaz de detectar signos de estos trastornos mentales en texto escrito puede ser de ayuda tanto para los pacientes, como para los médicos.

Viendo el potencial del aprendizaje automático en el ámbito de la medicina, sistemas como el realizado en este proyecto pueden suponer avances para la sociedad y mejorar la calidad de vida de las personas que pudieran padecer estos trastornos.

Palabras clave:depresión, estrés, detección precoz, aprendizaje automático, clasificador, Mattermost.

Abstract

Mental health is becoming a crucial issue in our society. Depression is the most common mental illness in the world and its incidence is increasing every year. Similarly, stress is becoming increasingly present and permanent in our daily lives. This is detrimental to those who suffer from it, both in terms of their health, as well as socially and at work.

This project aims to help in one of the crucial tasks for the improvement of mental health, early detection. The current diagnostic process requires the patient to go to the doctor for the necessary questionnaires or tests. This can be a problem given the social stigma surrounding mental illness and the people who suffer from it. Not all patients who should go to the doctor either out of fear, embarrassment, or ignorance.

For all these reasons, we have developed a system for the recognition of depression and stress in text, based on machine learning and natural language processing technologies. The system consists of a classifier, in charge of predicting mental illnesses, and a Mattermost plugin, which allows the text written by users in a text-based communication tool to be evaluated by the classifier.

This makes it possible to detect situations of work stress or depression of a worker earlier. In addition, if the classifier detects any sign, it can automatically send a message through the channel where the user is to recommend that he take a break to reduce his stress. A system that can detect signs of these mental illnesses in written text can be of help to both patients and doctors.

Seeing the potential of machine learning in the field of medicine, systems such as the one developed in this project can represent advances for society and improve the quality of life of people who may suffer from these disorders.

Keywords: depression, stress, early detection, machine learning, classifier, Mattermost.

Agradecimientos

Gracias a todos los que me han acompañado en este doble viaje y han permitido que pueda hacer frente a este proyecto.

En especial a Julia por estar siempre ahí, sea hasta la hora que sea.

También a Sergio por sus cientos de correcciones y siempre tratar de ayudar con el mejor humor.

Contents

List of Figures

CHAPTER 1

Introduction

This first chapter will introduce the context of the project, including a brief description of all different parts of the project, as well as the project objectives. Finally, the structure of the document will be detailed.

1.1 Context

In recent years, mental health has become extremely important. This is due to the increased normalization and the attempt to remove the stigma of these disorders, as well as the increased incidence. Mental illnesses are a group of different conditions that can affect mood, behavior, thinking, or all of them. They are usually caused by various biological, social, and psychological factors [1]. The most common disorders [2] are depression, both in the form of major depression and bipolar disorder; and stress. About 19% [1] of US adults suffer from some form of mental illness.

Symptoms vary depending on the disorder that produces them, but some examples are: feelings of sadness, altered concentration, mood swings, extreme feelings of anger, worry and tiredness, social withdrawal, and suicidal tendencies [3].

One of the disorders for which the incidence has increased the most since a few decades ago [4] is depression. Depression [5] is a mental disorder that affects mood so that feelings of sadness and low mood directly affect daily life. It has a higher incidence in women and people under 45 years of age, hence the young population is a risk group. It may be caused by a genetic component, psychosocial factors, biological factors, substance abuse, or certain organic diseases, such as cancer or Parkinson's disease [5]. Of particular alarm are the figures of depression in recent years in university students, since they reach over 50% [6].

The other disorder that we will focus on is stress. Stress has become an unavoidable factor in everyone's life, whether at the academic, work, or family level. Stress is the body's reaction to a stimulus that demands a reaction with more intensity than usual [7] . It can be physical, mental, or emotional, leading to feelings of mild upset, anger, frustration, anxiety, or even depression. On a physical level, it alters blood pressure, heart rate, and hormone levels such as cortisol or glucose [8].

In addition, given the current situation in which we live with the pandemic of Covid-19, mental disorders have increased considerably both by confinement, bewilderment, and the virus itself [9]. Both depression and stress are highly influenced by living situations and Covid-19 has caused their numbers to increase markedly [10].

In view of the importance of these illnesses in the current society, we now turn to one of their main problems, detection [11]. There is no objective test, as there may be for other illnesses, which is why most of them are diagnosed using tests and many interviews with the doctor. Although the validity of these methods has been demonstrated, a more objective test based on information obtained from the individual patient could facilitate diagnosis.

A field in which much progress is being made and could solve this problem is affective computing [12]. Affective Computing aims to develop intelligent systems capable of endowing a computer with the ability to recognize, interpret and process human emotions. It

can be used for all kinds of purposes, either to see what people think of a certain political candidate, to detect terrorist texts, as for issues related to the mental health of people. It can help in the analysis of tests and questionnaires, and tools such as the ones in this project that can predict by text can help in this task.

It is undeniable the increase of interest in these techniques and the high number of classifiers with great results that are obtained in all the feelings, which makes it a current technology, with endless possibilities to which it is predicted to continue growing and optimizing in the coming years.

1.2 Motivation

Having seen the context, we can understand the importance of mental disorders in our society today. Given the growing incidence of mental illness in the young population, social networks, as a way by which they communicate and express themselves, are an essential data and information point.

And it is precisely from social networks that we can obtain a large source of data on this issue. Thanks to developed systems such as the one in this project, we can improve one of the most critical parts of these diseases: detection. This system could help identifying in an economical and early way these mental disorders. Another strong point is that the sufferer can have an alert to go to the doctor, which is otherwise not easy to obtain. This is a social problem since this type of disorder has a certain social stigma that makes it more difficult to talk about them and for people to recognize them.

In addition, by doing it on a communication platform used mainly by companies, this can allow monitoring the stress levels of employees to see how happy they are in the company and prevent them from leaving to other companies or improve certain aspects. Having employees with poor mental health can affect productivity [13], as it is related to higher absenteeism and lower ability to concentrate. In addition, employees with better mental health are more motivated and more likely to stay with the company.

The system could complement other types of communication tools by making the detection of these mental illnesses easier in different areas of life.

1.3 Project goals

The main objective of the project is the development of a system capable of detecting mental illness in real-time. For this purpose, the objective is to create a classifier that detects signs of depression and stress. For this, it will be necessary to train a model with depression and stress datasets. These datasets, obtained from social networks, allow the classifier to detect

signs of these mental illnesses in text.

To make the system as practical as possible, this classifier will be integrated into a plugin developed for Mattermost. The goal is that the text written by users using the Mattermost channel will be the text that the classifier will evaluate. In this way, the system would be complete.

To meet these objectives, the following tasks must be carried out:

- Study the state of the art of Natural Language Processing, Machine Learning Algorithms, and diagnostic techniques for depression and stress.
- Make a dataset search to have a large database in which there are data from different social networks with different formats.
- Design and develop a classifier that detects signs of mental illness by performing the following steps:
  - Perform a study of the most relevant features of the datasets.
  - Make the preprocessing, so that the text is the optimal one to train the model.
  - Define the model architecture and train it with the datasets.
- Design and develop a plugin for Mattermost, able to read the text written by users and respond to them.
- Combine the plugin and the classifier so that the input text of the classifier is the one written by the user.

1.4 Structure of this document

The remaining of this document is structured as follows:

Chapter 2. State of art

In this chapter, we will focus more on depression and stress, looking at their theoretical models and new ways of detection. In addition, we will discuss the enabling technologies of the project.

Chapter 3. Model

In this chapter, we will look in detail at the data sources that have been used in this project, as well as the techniques needed to be able to use these data. We will describe the model used by the classifier.

Chapter 4. Evaluation

In this chapter, we will compare the results of the classifier when trained using different datasets.

Chapter 5. System architecture

In this chapter, we will look at the complete system architecture. The part related to the Mattermost plugin will be explained in detail.

Chapter 6. Conclusions

In this last section are the conclusions and future work on which the project can be further developed.

CHAPTER 2

State of Art

In this section we will see the state of the art in relation to mental illnesses and tools that allow their detection. In addition, we will see the enabling technologies that have allowed the realization of this project.

2.1 Mental illness

We will begin the state of the art by looking in more detail at the functional models of mental disorders. Having seen the context, we can understand the importance of mental health. Every day it is given more importance and the impact it has in all areas of our lives, even influencing the work performance in all kinds of jobs.

To do this, we will look at a diagram explaining the functional model of mental health. This will allow us to understand from a more visual point of view, what factors affect mental health and what consequences they may have. Seeing this, we can extrapolate what affects poor mental health and leads to mental illness [14].

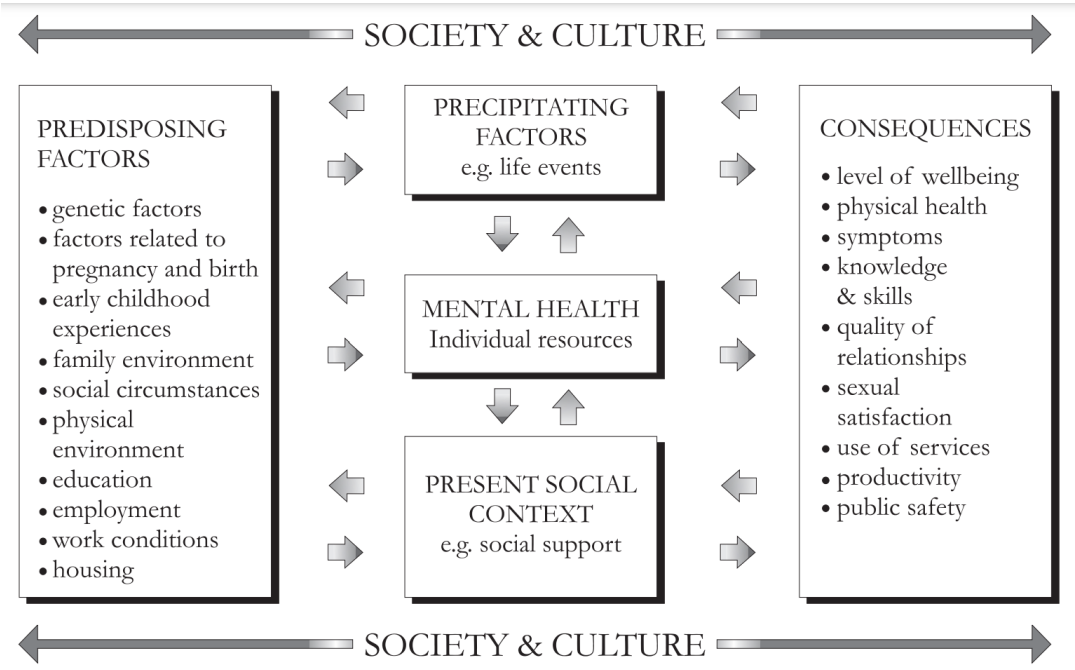

Figure 2.1: Functional model of mental health for promotion (modified from Hosman, oral communication 1997).

As can be seen in the diagram, different factors can condition mental health. The diagram is applicable for both depression and stress since both are mental health conditions and share these factors. All factors are related and are not isolated facts so that the same event does not necessarily affect two individuals from two different countries and ages in the same way.

On the one hand, we find the predisposing factors. These are specific to each individual, as they are related to childhood, education, or genetics. We also find the current social context of the individual, which is a fundamental part of the development of these mental disorders. For example, a bad work situation or divorce are highly stressful contextual

factors. Finally, we find the precipitating factor. As we have already said, this factor alone would not necessarily affect mental health, but when all factors are combined, it becomes the trigger for the disorder.

All this leads to the consequences. The consequences of developing any of these disorders affect all areas of the person. One of the most characteristic symptoms of both depression and stress is the effect on mood.

- **Depression.** In depression, the patient may have a feeling of sadness, apathy, disappointment, low energy, and existential crisis. To diagnose the disease, a series of symptoms must be met for two weeks, according to the DSM-V [15]. The ones we are most interested in are: depressed mood, decreased interest in daily activities, fatigue, feelings of worthlessness or recurrent thoughts of death or suicide. These are the most relevant symptoms to detect. We will now look at two examples of advances in the detection of this disorder through the application of machine learning techniques.

  And it is thanks to this affectation of mood that, together with affective computing, signs of these disorders can be identified.

  This technology allows to detect feelings from different sources, whether images [16], voice [17], facial recognition [18] or text. We will focus on text as it is the one used in our developed system.

  One of the great advances in recent years in the use of machine learning for the detection of mental illnesses is a model created by Canadian researchers in 2020 that detects depression [19]. It uses text to do so, and its main source of data is Twitter. One of the important advances of their software is that it demonstrates that there is a depressive language with its linguistic representation. This can be used for the early detection of depression. As future work, they propose to integrate the model with a bot to communicate with the elderly, to prevent them from loneliness, and to detect if they have depression.

  Another example is that carried out by a group of researchers at the University of Birmingham [20]. The idea was to use machine learning techniques to help in the detection of depression and psychosis. They gathered a group of people with these possible disorders and employing tests and machine learning techniques, they tried to detail with greater precision the symptoms of each of these disorders. Thanks to the techniques used, they were able to diagnose patients more accurately and differentiate those with depression from psychosis more effectively, seeing the importance of these tools in the world of mental health.

- **Stress.** In stress, on the other hand, there may be crises of anger, moodiness, rage,

anxiety, and irritability. Stress can be more complicated to detect as it can manifest itself in different ways. It usually deals with anguish at the triggering event, the feeling of not being in control of important things in life, nerves, feeling that everything is going wrong, and constant tiredness.

Regarding stress, we find that most advances try to detect stress through psychophysiological signals [21]. This includes different factors that can be measured from an electrocardiogram to body temperature and respiration.

Although no single algorithm has been chosen as the best, as research in this field is still ongoing, it has been possible to achieve an accuracy of over 80% with several algorithms [21].

Therefore, with the creation of this system, we want to contribute to the state of the art in the detection of mental illnesses utilizing machine learning techniques with our system. The system can predict signs of two different mental illnesses and is trained with information sources from different social networks, which makes it a breakthrough.

2.2 Text classification technologies

We will see below the technologies that have been necessary for the creation of the system.

2.2.1 Machine Learning Technologies

Machine learning has different definitions as we will now see, but a general definition is that it is a branch of Artificial Intelligence to make systems learn automatically. The first definition is from Arthur Samuel in 1959: "Field of study that gives computers the ability to learn without being explicitly programmed". Another definition that allows a better understanding of the concept of learning is that of Tom M Mitchell in 1997: "A computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at task in T, as measured by P, improves with E".

Having seen the definition, let us see the types of Machine Learning. There are 3 categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

- **Supervised Learning** In supervised learning, the goal is for our model to predict a value or label after training it with a lot of data so that it can generalize and determine or predict a value. That is to say, the model is trained with a dataset with the label that would correspond to each case and when the model receives new data, it is able to correctly predict the corresponding label. This would be a *Classification* problem.

The model must predict a discrete value, as a result of the training to which it has been subjected by looking at the characteristics of other values similar to the one to be predicted.

Another type of problem is the *Regression* problem, which is used to predict continuous rather than discrete values.

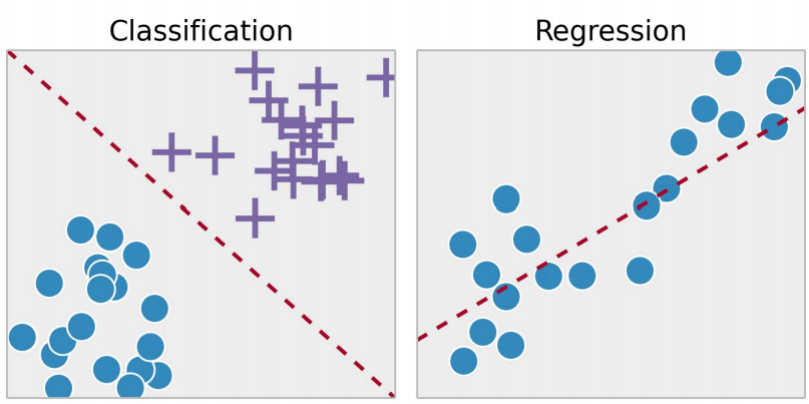

Figure 2.2: Supervised Learning problems.

All this is done by algorithms and consists of two phases, training and testing. The dataset data is divided into two groups for this purpose, generally allocating 70% to training and 30% to testing, although these percentages may vary. Examples of algorithms: KNN or SVM for classification and Linear Regression or Logistic Regression for Regression.

- **Unsupervised Learning** This type of learning is based on trying to understand and obtain patterns from data of which there is no prior knowledge or label. The characteristic problem of this type of learning is Clustering. In it, we try to cluster the data to separate them. Data with similar characteristics are grouped so that when a new one is introduced, a decision is made as to which cluster it most resembles. An algorithm for this type of problem is K-means.

- **Reinforcement Learning** Reinforcement learning is based on rewarding good actions and punishing bad ones so that models learn from their own experiences and do not require large amounts of data. The reward or punishment is given based on a function to try to be as optimal as possible [22].

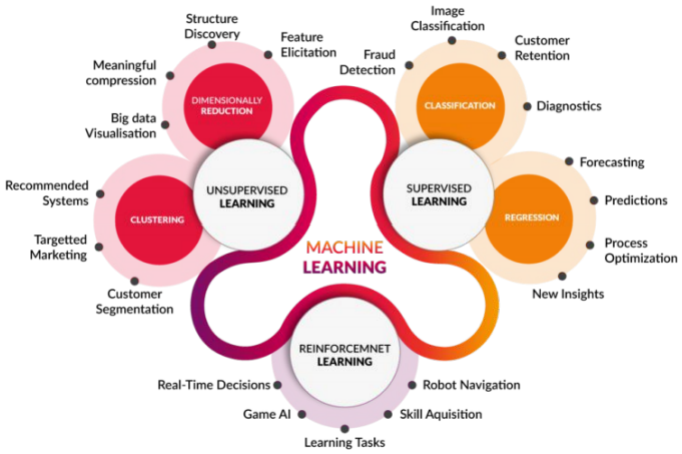

Figure 2.3: Machine Learning types

2.2.2 python

Python [23] is an interpreted programming language widely used in Machine Learning, Big Data, web development, among others. It was born in 1991 by Guido van Rossum. An interpreted language means that the code is translated and executed at the same time. It is also dynamically typed, which means that variables can take different values adapting to the code.

2.2.3 Python extensions for Machine learning

Once Python has been introduced, we move on to the libraries and modules that facilitate the analysis and use of data for machine learning.

Pandas. Pandas [24] is a Python library, which facilitates the connection between data analysis and Python, without the need to switch to a more specific language, such as R, providing a good structure for the data that allows the creation of a complete data analysis flow.

Some data structures that Pandas add are: Series, Dataframes, and Panels. Series are One-dimensional indexed arrays. Dataframes are similar to a database table, allowing to work in a more usual way. Panels are data structures with more than two dimensions.

The main features and functionalities provided by the pandas library are tools for reading and writing in different file formats and data grouping, data merging, and query operations, as well as time series analysis.

NumPy. NumPy [25] is based on a module called Numeric, which allows Python to do complex calculations, adding mathematical and vector capability to Python enabling operations on numeric data or arrays.

SciPy. SciPy [26] is a group of mathematical functions and algorithms built on top of NumPy. It allows to raise the functionality to compete with Matlab or R-Lab, creating data processing environments. A strong point is that it allows to develop very specialized programs in Python, that being such a versatile and used programming language is a very strong combination. For example, it adds optimization, linear algebra, ODEs resolution, and more. Its function is to allow the use of high-level commands to manipulate and display data.

NLTK. Natural Language Toolkit [27] is a library implemented in Python, which is used in Machine Learning for natural language analysis, is open source, and is available for different operating systems. It is a very powerful tool with many different internal modules, each with a specific function.

NLTK provides more than 50 corpus and lexical resources that allow classifying, tokenize, stemming, tagging, and the ability to classify, tokenize, stem, tag, and analyze [27].

Language processing task	NLTK modules	Functionality
Accessing corpora	nltk.corpus	Standardized interfaces to corpora and lexicons
String processing	nltk.tokenize, nltk.stem	Tokenizers, sentence tokenizers, stemmers
Collocation discovery	nltk.collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	nltk.tag	n-gram, backoff, Brill, HMM, TnT
Classification	nltk.classify, nltk.cluster	Decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	nltk.chunk	Regular expression, n-gram, named entity
Parsing	nltk.parse	Chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	nltk.sem, nltk.inference	Lambda calculus, first-order logic, model checking
Evaluation metrics	nltk.metrics	Precision, recall, agreement coefficients
Probability and estimation	nltk.probability	Frequency distributions, smoothed probability distributions
Applications	nltk.app, nltk.chat	Graphical concordancer, parsers, WordNet browser, chatbots

Figure 2.4: NLKT module functionalities

2.2.4 Transformers

Transformers [28] is an architecture with pretrained models for Natural Language Understanding and Natural Language Generation. It consists of more than 30 pretrained models in more than 100 languages [29].

It has become the state-of-the-art of NLP. Transformers is backed by the three most important deep learning libraries, Jax. PyTorch and TensorFlow

It is based on an Encoder and a Decoder. The first one analyzes the context of the input and the second one generates the output based on that. For all this to work, it is necessary to vectorize the words, that is to say, to make Word Embeddings. This transforms the words into a numerical representation, where the words that belong to similar contexts will be closer together [30].

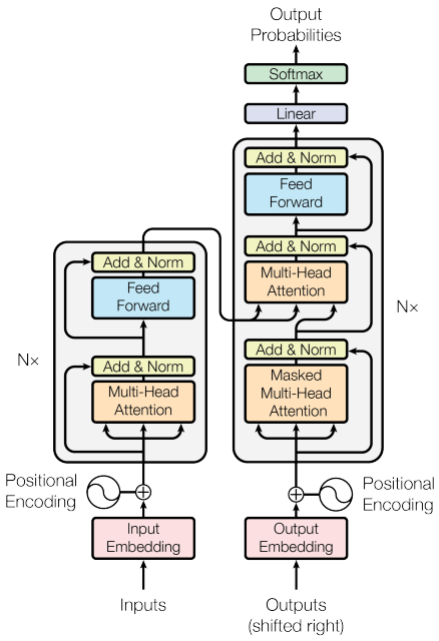

Figure 2.5: The Transformer - model architecture

Since the text flows simultaneously between the encoder and the decoder, it is necessary to include information about the position of each word in the vector.

A sinusoidal function is used for this purpose.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

This function was proposed since the process is deterministic, there is a unique value for each position of each word, and the range of values is consistent and independent of the length of the sequence.

To understand the relationship between words in a sequence, it employs a self-attention mechanism. Thanks to this, the reference used for the context becomes infinite, bounded by the computational power, uniquely. This is why you can use the entire context to identify the relationships between words.

This mechanism works based on the scalar product of [30] :

- **Q** The query that represents the vector of a word.
- **K** The keys which are all other words in the sequence.
- **V** The vector value of the word being processed at that time point.

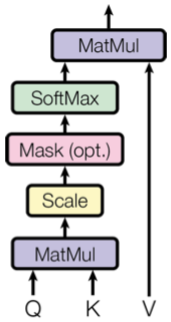

Figure 2.6: The Transformer - Scaled Dot-Product

The objective is to calculate the orthogonal projection of Q onto K. To do this, the scalar product of Q and K is transposed, divided by the root of the size of k. For this, we apply softmax so that the range of the weight is between 0 and 1 and, finally, we multiply by V to reduce the importance of irrelevant words.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

2.2.5 Mattermost

Mattermost [31] is an open-source proprietary SaaS messaging platform for enterprises. It provides an instant messaging service and allows the creation of public and private channels, where users exchange messages or files. It can also be complemented with other very useful external applications for companies such as Jira or GitHub, as well as with social networks. It is cross-platform and allows you to add plugins to extend its functionality.

2.2.5.1 Plugins

Plugins are an essential part of Mattermost, as they allow to complement and add new functionalities and integrate them in a simple way. Its basic components are a manifest, which can be defined in YAML or JSON, used to tell Mattermost what the plugin is, how to install and run it, and a server binary and/or a JavaScript Bundle. The server binary is a compiled program written in Go that runs as a Mattermost internal server process and interacts via RCP and the API. On the other hand, the JavaScript bundle runs in the Mattermost web/desktop apps. The client code is registered with the Mattermost code as part of the plugin reducer. Once this is done, plugins can be inserted and add components in different parts of the UI. The parts are compressed and a .tar.gz file is obtained and uploaded to Mattermost.

2.2.5.2 Bots

Bots are a essential part of Mattermost plugins. They allow establishing communication with users in an automated way. Bots access the Mattermost RESTful API.

Bots are administrator-created accounts, which are similar to regular user accounts except that they cannot be logged into, cannot create other bots, and do not count as users in channels.

To create a bot, once you have allowed the creation of bot accounts, from the Integrations menu access bot accounts. Here, the user can create a bot with the name, image, and description of his choice. In addition, the bot role can be selected. This can be Member or System admin, in the latter the bot can access all channels and direct messages, and in the other only those assigned to it. Once this is done, the bot token is obtained, which is an alphanumeric code that later will be introduced in our plugin for the correct connection.

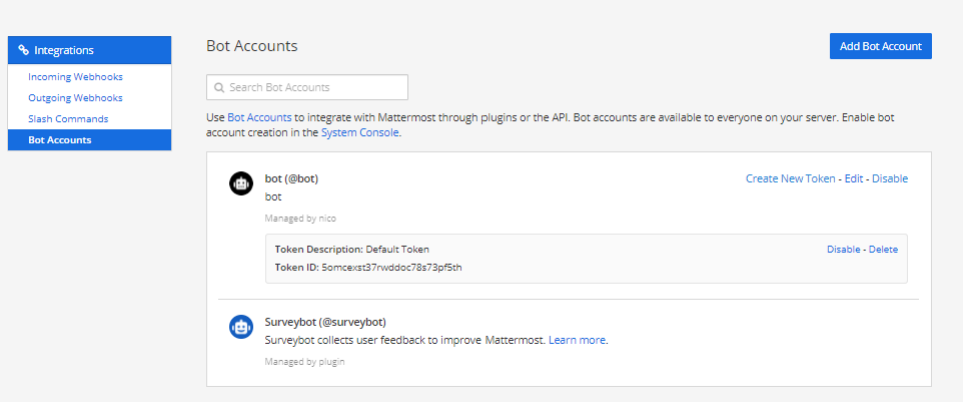

Figure 2.7: Bot account creation

The bots allow to post messages in any channel according to the configuration of the administrator, they allow avoiding that users are created for mechanical tasks, and thus to avoid a greater number of accounts to manage.

2.2.5.3 mmpy_bot

Mmpy_bot is a chat bot framework for Mattermost written in Python. It is easily installable and its code is available on GitHub. It works from Mattermost version 4.0.0 onwards. Once everything is installed, it allows you to configure the bots the way you want. This makes developing plugins for Mattermost much easier, since it is not necessary to use Go and it can be done in Python, a widely used language. Furthermore, thanks to the documentation and examples, it is very easy to create a basic plugin.

CHAPTER 3

Model

In this chapter, we will see everything related to the model used in the classifier. We will begin by looking at a detailed analysis of the data used to train the model. Then, we will explain the processes that need to be done on this data to train the model. Some parameters used in the model will be defined. Finally, we will describe how the model is described and what steps are performed when training and validating data.

3.1 Introduction

The classifier is the fundamental part of the Mental Illness Detection System or MIDS, since it is the one that allows us to predict whether the user could have signs of mental illness. We will start by looking at how to work with the data, how to build and train the model and get it to make predictions.

3.2 Data

Data are needed to train the model. Thanks to social networks, nowadays it is much easier to obtain datasets with thousands of data on a specific topic. In our case, we need datasets that provide us with information about mental illnesses, with a text and a label next to it that tells us if the text is associated with the disorder or not.

Since our classifier must predict both depression and stress, we will need at least one dataset of each, although the results may be similar with only one. Many of today's classifiers focus on a specific type of post, be it tweets or Reddit posts, which means that when confronted with actual text, the results may not be equally accurate, not being of the same length or with the same structure. Our datasets are from both Twitter and Reddit, with all text written in English.

Our dataset has a text column, where you find the text that the user wrote in the social network, and a label where 0 or 1 tells if that text is associated with a person with a mental illness. With this, we will train the model, so that it will be able to predict the label correctly.

In order to perform a detailed analysis of the datasets, the main characteristics of all of them will be shown in a table. In this way, they can be compared and defined.

	Dreaddit	**TensiStrength**	**Tweets Depression**	**Test Depression**
Number of samples	3545	6135	10314	40003
Mean number of words per sample	88.36	15.57	10.12	14.68
Non-stress/depression posts	1693	2530	8000	13255
Stress/depression posts	1852	3605	2314	26748
Label mean	0.522	0.587	0.224	0.669

Table 3.1: All dataset comparison.

3.2.1 Stress datasets

For the stress part, the model has been trained with two datasets.

- **Dreaddit.** Dreaddit is a dataset made for stress analysis [32], whose source are posts from the social network Reddit. It was created in 2019 and within it we find 3545 Reddit posts. Of these, more than half, 1852, have been classified as depression. We see in the label mean, that this is 52% of the samples, so the dataset is almost equally distributed between positive and negative. Moreover, being from Reddit, the maximum number of words and a half is higher than if it were from Twitter.

  In order to go deeper into the text found in this datset, a word cloud has been created, with the most frequent words in the dataset.This allows to have a quick and visual idea of the most common topics in the dataset. They are shown in the figure below.

Figure 3.1: Most frequent words in Dreaddit.

A word cloud assigns a different size to each word according to its frequency in the dataset. Thus, the words with the largest size are the most frequent ones. We see words like time, feel, anxiety, work that we could relate to stress.

- **Tensistrength** TensiStrength is an algorithm to detect the level of stress or relaxation in social web texts [33]. To do so, they use a dataset with tweets, evaluating their stress level. This dataset contains many more samples. Being tweets, we see that the average number of words per sample is considerably reduced, as well as the maximum number of words in a sample. There are more positive than negative tweets, with a ratio of 58.7%. Samples on Twitter tend to be shorter as more characters such as @

or exclamations are used than when writing a post. Hence, the datasets tend to be larger.

The following figure shows what a word cloud looks like.

Figure 3.2: Most frequent words in TensiStrength.

We see that words such as time appear with high frequency in both. We see in this certain words that are commonly negative, such as shit, hate, bad and others that are simply widespread, such as people or want.

3.2.2 Depression datasets

In the depression part, it was more difficult to find good sources of data, as the data were either confidential or created in an impractical way. Therefore, we have used two GitHub datasets that are with the necessary format and we have already worked with them in other projects obtaining good results. For the depression part, the datasets were:

- **Tweets Depression.** This is made up of a series of tweets related to depression and evaluated with a label, depending on whether the tweet is considered to have signs of depression or not [34].

  We see that it is made up of more than 10,000 tweets. Unlike the two previous ones, in this one there are more negative samples than positive ones. This is because this dataset uses tweets related to depression, but it can be either way. Therefore, many samples can talk about depression in an informative way or from a medical point of view, but they do not have to be written by users with this illness.

  Here are the most frequently occurring words.

Figure 3.3: Most frequent words in Tweets Depression.

It can be verified that the main one is depression, but it looks like words of all kinds, such as love or day.

- **Test Depression.** This dataset is also obtained from Github [35]. It is the largest of all, consisting of tweets containing negative sentiments.

  This dataset consists of 40,000 tweets. Of these, almost 70% have been related to depression disorder. Being tweets, we see that the length of the samples is still short.

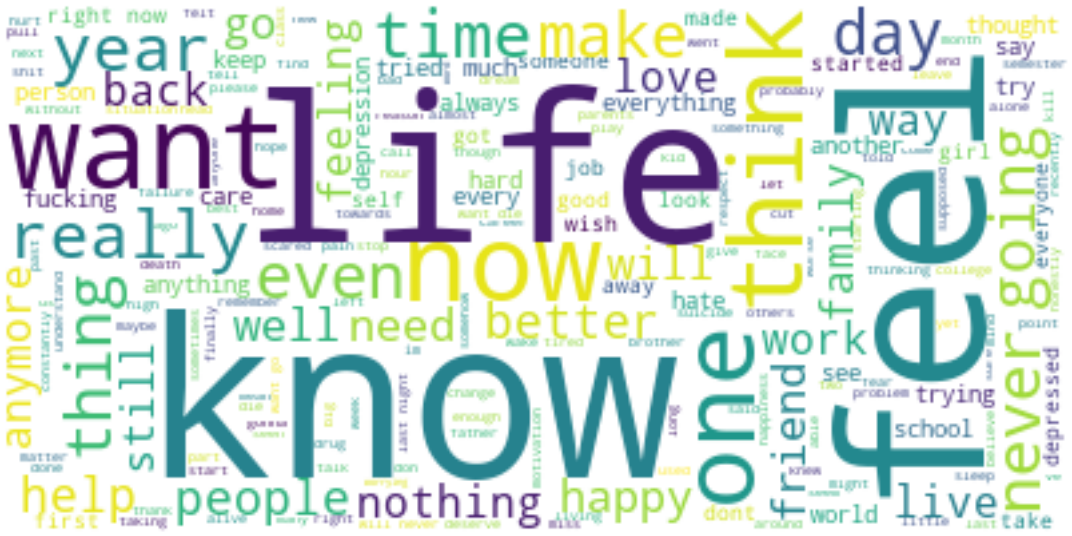

Figure 3.4: Most frequent words in Test Depression.

As for the most frequent words, the most frequent are life, feel, and know. These are filtered by negative sentiments, so many words could be in a negative form.

3.3 Preprocessing

Prior to data preprocessing, it is necessary to connect to a GPU environment. For this purpose, Google Colab will be used, which allows you to create a notebook and connect to a hosted environment.

Once connected to the environment and configured to use the GPU, the dataset is loaded using Pandas. Once loaded, it allows knowing a lot of information about it, such as how many rows it has, what type a certain column has, or to show the first five entries. The important part of the dataset for us, the text and labels, we will put them in a list each one.

However, as already mentioned, the model is not trained with raw text, it needs to be cleaned and converted. The input text must be cleaned so that it is as optimal as possible to be processed in the best way. Cleaning the text consists in eliminating the part that is not essential information, such as punctuation marks, stopwords, etc. Stopwords are words that are meaningless by themselves. In general, they are adverbs, prepositions, conjunctions, and articles. By eliminating them, the amount of text to be processed is reduced and the relevance of all remaining words is increased.

3.3.1 Tokenizer

For this process, we need to tokenize the text. This means moving from text to single words separated by blanks. To do this, we will use Bert's tokenizer, BertTokenizer. Taking advantage of Bert's many features, we will use the encode_plus function, which will not only separate individual words, but also convert them directly into input_ids. This is a list of numbers that are the values of each word in Bert's vector field. In addition, it will also create the attention_mask, which indicates in which part of the input_ids list is the information, since each input can have a different length, the missing part is filled with zeros. On the attention_mask list, it is indicated which parts are zeros and must be ignored. The encode_plus function has certain options:

- (1) Tokenize the sentence. As we have seen, this means separating the text into individual words.
- (2) Prepend the '[CLS]' token to the start. This token is added at the beginning to inform that the task is to classify the sentence. In this way, Bert knows where each sentence starts and what is the task to be performed.
- (3) Append the '[SEP]' token to the end. This token is added to the end of each sentence to indicate the separation with the next one.

- (4) Map tokens to their IDs. In this step, we go from having words to having a list of their equivalents that will be used to train the model.
- (5) Pad or truncate the sentence to 'max_length.'As the phrases can have different lengths, pad takes care of filling with zeros up to max_lenght if the sentence is shorter, while truncate removes anything longer than this length. This way, they all have the same length.
- (6) Create attention masks for [PAD] tokens. As some sentences are filled with zeros, we need the attention_masks to know in which part there is information and which part is padding. Thus, nothing that is not information is processed.

Once this is done, the lists are transformed into tensors using torch. Once we have these tensors, we create a dataset formed by the three input_ids, attention_mask, labels. This dataset will be divided into the part to train and the part to validate. In our case, it will be 90% for training and 10% for validation.

3.3.2 Before Training/ Hyperparameters

An important concept that comes into play here is that of *Batch size*. This defines the number of samples to be propagated through the network. In our case, the batch size is 16, since with 32 there was not enough GPU in the hosted environment. The network will train with the first 16 samples and then take the next 16, and so on. This has the advantage that it does not collapse the memory by not training the data at once but in small batches. Recommended size is 16 or 32. The smaller the batch, the faster the model will be trained and the more likely it will fit in memory. On the other hand, the larger the batch, the more accurate it will be in estimating the gradient, but it will take longer and occupy more space.

Another important parameter is the *Epoch*. This is the interval from one update of the weights of an ML model with respect to a batch to the next update on the same observation or batch. The recommended values are between 2 and 4, in our model, it is 4, which is the standard value.

The learning rate is a hyperparameter to regulate the rate at which the model updates its cost function. We will use the most commonly used and recommended value, which is 2e-5.

3.4 Model

Once this is done, we load Bert's model. We are going to use BertForSequenceClassification.from_pretrained. This means that it is Bert Model transformer with a sequence

classification/regression head on top. That is, on top of the basic architecture of the model, we put a special sequence for the case where we want to make a prediction of a label.

This model is a neural network [36] . This is used in deep learning. A neural network is a machine learning model that is based on the human brain structure. It is a network of interconnected nodes that each performs a simple computation. The way the nodes interact is similar to that of neurons, hence the concept.

Some differences between machine learning and neural networks are that neural networks learn from their errors, they do not need human intervention to improve. The neural network can make decisions autonomously, while in a classical machine learning model it makes decisions based on what it has already learned by processing the data. Neural networks directly employ a large number of optimized algorithms.

Once the model is loaded, the batch size, and the number of epochs are assigned, the model training starts. A essential part of the models based on neural networks is the loss function. In these models, one of the objectives is to minimize the error in each iteration. This is done through the loss function. It is combined with an optimization function so that this function reduces the error at each iteration. In most classification problems, the function used is Cross-Entropy.

At each iteration, the model updates its values based on the loss function, with the values obtained from the gradients, the learning rate that has been selected, and other model settings. In addition, it can update the learning rate at each iteration.

Once it has trained the model with the training data, it validates with the remaining data of the dataset. In this case, the accuracy of the predictions made by the model is checked.

The training of the model and its validation is done once per epoch so that as the model learns more each time through the functions, we have seen, the model can improve from one epoch to another. This does not mean that it always does, because if the model is trained too much, overfitting can occur, which makes the validation results not very good, since the model learns too much from the input data but when it receives a new one, it is not very good at classifying it.

Once this step is finished, the model can be saved for use in other applications, such as the plugin that has been developed in this project.

In addition to the validation, it is convenient to do a test part, in which the model is tested with a part of the dataset that has not been trained or validated or with new sentences. In our case, we have tested it with part of the datasets and with sentences written by us by hand, to see if we could see in which sentences the predictions failed more. We concluded that the longer the sentence, the higher the accuracy of the model. In our best case, we have achieved an accuracy of we obtained 87% accuracy.

The part of testing with data that the model has not seen before is important to check that the model has really learned. If this were not done, the model could have *learned the data by heart*, instead of actually learning. Furthermore, by testing with sentences written without a defined format, such as a tweet, we tested the effectiveness of the system in a function similar to the one it will perform within the developed system.

In the next chapter, we will see the comparison between the results achieved by the different datasets.

CHAPTER 4

Evaluation

In this chapter, we will see the comparison between the results obtained with each dataset. For this purpose, some concepts related to the evaluation of results in machine learning models will be defined. Next, the results obtained with each one will be shown. Finally, some common conclusions will be drawn.

4.1 introduction

In this section, we will see the comparison and analyze the results obtained depending on the dataset used. To do this, we will look at the results for each epoch of each dataset and compare the graphs obtained.

To understand how a model is evaluated, we must first define some concepts. These are the possible results we have with the predictions.

- **True Positive (TP).** A true positive is one in which the label has been predicted to be positive and is correct. In our case, it would be to label a text as having signs of mental disorders and actually having them.
- **True Negative (TN).** A true negative is the case where the model is correct but predicts a negative label, i.e. it does not detect signs and there are no signs.
- **False Positive (FP).** A false positive is where our system predicts a positive in a case that was negative. That is, it detects signs of mental disorders in a text where there are none.
- **False Positive (FN).** A false negative is the one that our system predicts as negative but is actually positive. The text has signs of mental illness, but the system does not detect them and predicts a negative.

Once this has been seen, it will be explained how they are combined to generate these metrics that allow the evaluation of the models. We will define the most common metrics for the evaluation of the results. These are: precision, accuracy, recall, and f1-score.

- **Precision.** Precision measures the number of true positives among the total number of positive predictions of the model.

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

- **Accuracy.** Accuracy measures the proportion of correct predictions among all samples tested.

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

- **Recall.** Recall is used to measure the accuracy of the model for classifying true positives. It measures the percentage of true positives that are correctly classified.

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

- **F1-score.** It is the mean between precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

To compare the results, tables have been obtained in the training of the model. These show the data for each epoch of Training Loss, Validation loss, Validation accuracy, and the time taken to train and validate.

4.2 Stress datasets

- **Dreaddit.** We will see for each dataset the results of training and validating the model, the loss function both training and validating, and finally, the results of testing with a small part of the dataset that the model has not seen before.

epoch	Trainig Loss	Validation Loss	Validation accuracy	Training Time	Validation Time
1	0.49	0.40	0.77	0:05:03	0:00:12
2	0.32	0.39	0.80	0:05:03	0:00:12
3	0.18	0.46	0.85	0:05:03	0:00:12
4	0.11	0.55	0.84	0:05:03	0:00:12

Table 4.1: Dreaddit results.

In this case, we see how the training function decreases, while the validation accuracy increases. The loss function of validation does not follow the same dynamics as that of the downward training.

In the graph, we can see how the training graph goes down while the validation graph does not. This is because the model focuses on the training function rather than the validation function, which focuses more on the accuracy of the predictions. The objective of the model is to keep the loss training function as small as possible.

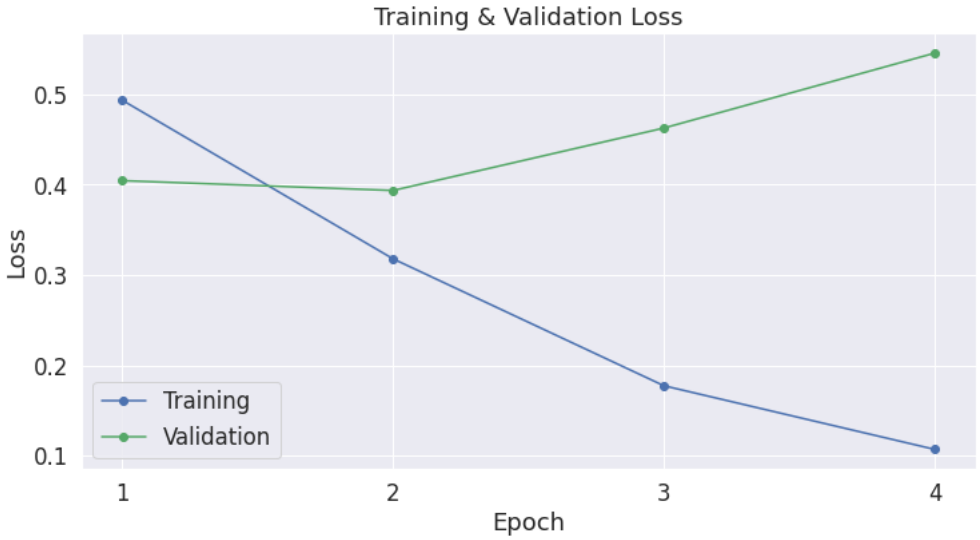

Figure 4.1: Training vs validation loss with dreaddit dataset.

We will now take a look at the results of the test with the Dreaddit dataset.

	Precision	**Recall**	**F1-score**
0	0.83	0.75	0.78
1	0.78	0.85	0.82
accuracy			0.80
macro avg	0.80	0.80	0.80
weighted avg	0.80	0.80	0.80

Table 4.2: Dreaddit test results.

We see that the accuracy is 80% which is a good figure and the training has been satisfactory.

- **TensiStrength.**

  In this case, we see that the training is faster. This is because tweets are shorter than Reddit posts, so the model trains faster. The training loss function starts higher but decreases similarly to the previous one. In the first epoch, the validation accuracy was higher than in the following epochs, this may be due to the samples used or that as already mentioned, there was some overfitting.

epoch	Trainig Loss	Validation Loss	Validation accuracy	Training Time	Validation Time
1	0.57	0.43	0.82	0:00:32	0:00:01
2	0.40	0.40	0.81	0:00:32	0:00:01
3	0.29	0.43	0.81	0:00:32	0:00:01
4	0.21	0.47	0.81	0:00:32	0:00:01

Table 4.3: TensiStrenght results.

We see that the curves are very similar in both datasets.

Figure 4.2: Training vs validation loss with TensiStrength dataset.

When training with this dataset, we see that it has considerably more accuracy in predicting positive than negative samples. This may be due to the format of each sample, with the positive samples being longer than the negative ones, which tells us how good Bert's model is.

	Precision	**Recall**	**F1-score**
0	0.75	0.89	0.81
1	0.86	0.70	0.77
accuracy			0.79
macro avg	0.80	0.79	0.79
weighted avg	0.80	0.79	0.79

Table 4.4: TensiStrength test results.

4.3 Depression datasets

- **Tweets Depression.**

  In this one, the validation function is flatter than in the others, which indicates that the samples may be more homogeneous. It obtains an accuracy of about 80%, in a similar way.

epoch	**Trainig Loss**	**Validation Loss**	**Validation accuracy**	**Training Time**	**Validation Time**
1	0.50	0.40	0.82	0:01:45	0:00:01
2	0.38	0.40	0.81	0:01:46	0:00:01
3	0.29	0.45	0.80	0:01:46	0:00:01
4	0.17	0.42	0.81	0:01:46	0:00:01

Table 4.5: Tweets Depression results.

We see that the training loss function continues to improve with each epoch, going from 0.5 to less than 0.2, while the validation function remains between 0.4 and 0.45.

We can see that in this case, we obtain a slightly lower precision than the previous ones, but it is still good. Since there are many more negative samples than positive ones, the model predicts much better the negative ones since it has been trained more with them.

Figure 4.3: Training vs validation loss with Tweets Depression dataset.

	Precision	**Recall**	**F1-score**
0	0.87	0.81	0.83
1	0.70	0.80	0.75
accuracy			0.77
macro avg	0.79	0.80	0.77
weighted avg	0.79	0.80	0.77

Table 4.6: Tweets Depression test results.

- **Test Depression.** In the last one, we see how the accuracy improves in each epoch.

epoch	**Trainig Loss**	**Validation Loss**	**Validation accuracy**	**Training Time**	**Validation Time**
1	0.40	0.46	0.77	0:03:05	0:00:22
2	0.32	0.39	0.79	0:03:05	0:00:22
3	0.25	0.40	0.81	0:03:05	0:00:22
4	0.22	0.52	0.80	0:03:05	0:00:22

Table 4.7: Test Depression results.

The loss function curve is similar to the previous ones.

Figure 4.4: Training vs validation loss with Test Depression dataset.

We obtain an accuracy of 80% which is the average accuracy of the model.

	Precision	**Recall**	**F1-score**
0	0.82	0.81	0.81
1	0.80	0.81	0.80
accuracy			0.80
macro avg	0.81	0.81	0.80
weighted avg	0.81	0.81	0.80

Table 4.8: Dreaddit test results.

4.4 Conclusions

Given the above, we can say that the model will make predictions with good accuracy. With all datasets that have been tested, similar results are obtained, all-around 80%, which is a good value considering that there are datasets from different social networks.

To see it in more detail, the confusion matrix generated with the Dreaddit dataset is

shown below. The confusion matrix is a useful tool, which allows to evaluate the performance of a model in a quick way. On the upper part would be the actual values and on the lateral axis, the predictions.

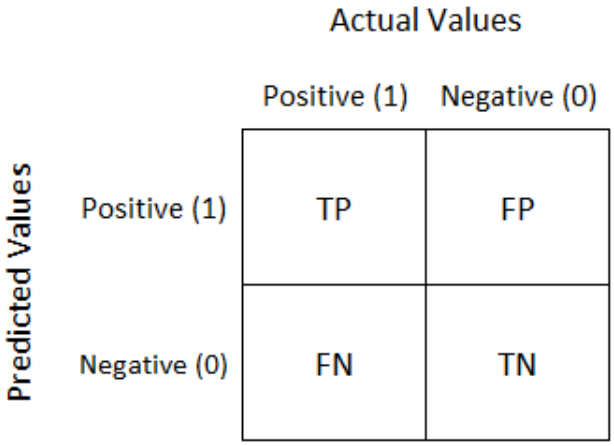

Figure 4.5: Comfusion Matrix.

In this one, we see that most of the samples are on the main diagonal, which is the one with correct preconditions. We see that 2813 samples have been classified correctly and 732 have been classified incorrectly, which is about 80% of the samples.

	Positive	**Negative**
Positive	1463	343
Negative	389	1350

Table 4.9: Dreaddit confusion matrix.

With all that we have seen, we can affirm that the classifier will perform its function correctly before the two mental disorders and with different types of input data.

CHAPTER 5

Architecture

In this chapter, we will look at the overall architecture of the system. Each module that makes up the system will be detailed.

5.1 Introduction

In this section, we will see what the overall system architecture looks like. We will define the modules of which the system is composed and we will delve into how the Mattermost plugin has been developed.

5.2 Architecture

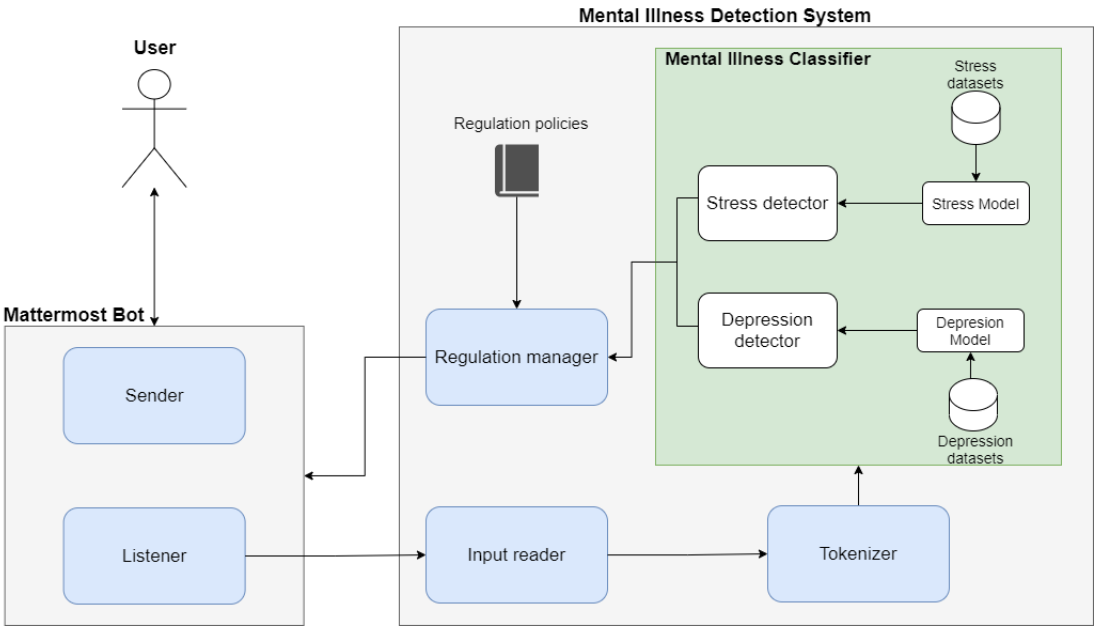

Figure 5.1: System architecture.

This is the general architecture of the system. It is composed of two blocks: the Mattermost bot and the Mental Illness Detection System or MIDS, which includes the Mental Illness Classifier. We will now define each one in detail.

5.3 Mattermost bot

The bot is the fundamental module of the project for communication between the user and MIDS. The main function of this module is listening to the text, which is written by the user, and taking any action automatically like sending a message.

The user connects to his Mattermost channel from his usual device. In this channel, the bot will have already been introduced, so that whatever this user writes will be heard by the bot. The user can see that the bot has been added to the channel, but not interact with it.

The bot will continue to listen silently until the Regulation Manager informs the bot that the system has detected a few possible signs of mental illness, so the bot will have to reply. In this case, the bot will send a message that will appear to the user in his channel as a response to his message. In this way, the user can know that a sign of stress or depression has been detected and in which message it has been.

It is built on the framework of mmpy_bot, which allows it to have been created in Python, which facilitates the integration of the classifier.

It consists of two parts, the Listener and the Sender.

Listener. Listener is in charge of listening to what the users write through the channel. This is done employing mmpy's listen_to function, which works with regular expressions. Which means that it can be configured to listen only to phrases that begin with certain words or from certain users, in particular In our case, as we want to listen to everything the users write we use .*. This text is saved thanks to the Message function and passed to the Mental Illness Detection System.

Sender. The Sender is in charge once it has received the alert from the MIDS of replying to the message from the user that caused the notification to arrive. To reply to messages, the bot uses the reply_to function of mmpy_bot. With it, the bot can answer messages to any user as if it were another user, but in an automated way.

5.4 Mental Illness Detection System

The Mental Illness Detection system or MIDS is in charge of processing the text and communicating to the bot the action to take.

The bot listens to everything written by the user and passes it to the MIDS through the Input Reader. Since the module does not understand the text itself but needs it to be vectorized, it passes it to the tokenizer. Once it is done, it can be sent to the submodule inside, which is the Mental Illness Classifier. When the answer is obtained, it is sent to the Regulation manager, and depending on what it is and the regulatory policies, it gives the bot the order of what to do.

Input Reader. The input Reader receives the text that the bot listens to. This is done thanks to the message function of mmpy. The bot stores everything it listens to in a variable which is what the Input reader gets and sends to the tokenizer.

Tokenizer. This part is essential for the Mental Illness Classifier to work properly. The system uses BERT's tokenizer, which is BertTokenizer. Once the tokenizer is created with the encode_plus function, it is passed from the text to the necessary vectors. Once you have it, it is converted into tensors by torch.cat as it is the necessary input for the model to make its prediction. And this is what it puts on the Mental Illness Classifier so that it makes the

prediction.

Regulation manager. Once the classifier has obtained the prediction, the Regulation manager is in charge of giving instructions to the bot, depending on whether it has received '0' or '1' and the regulation policies that have been established. These regulatory policies have been created at the same time as the plugin.

- **0 or negative.** In case the classifier does not detect mental illness, it answers with a '0'. In this case, the Regulation manager tells the bot to continue listening.In addition, if for a period of time no sign is detected in any message, the Regulation Manager will empty its counter and send a congratulatory message via chat. The usual time for this is a full working day.

- **1 or positive.** In case the classifier does detect a mental illness, a Regulation Manager counter will be incremented, so that if a certain number of messages with signs of mental disorders are detected over a period of time, it will alert the bot. In this way it is possible to control the level of stress at which it acts. If there is a case of depression, it will also be detected since most of the messages sent by the user with depression will be detected by the MIDS, causing the warning to be triggered. The response that the bot will give in case the threshold is reached, will be *Take a break!* since it has been considered that most of the messages that will be detected in a company chat environment are of stress, which could be reduced with a break.

5.4.1 Mental Illness Classifier

This module is in charge of the main part of the system, which is to detect mental illnesses. For this purpose, the models have been trained using specific datasets for each disorder. Once these have been sufficiently trained and good results have been obtained, the classifier is ready. When a tokenized text arrives, the model will make its prediction about mental illnesses. In case it detects it, it will send a '1' to the Regulation manager. Otherwise a '0' will be returned. With this, the system can already predict signs of mental disorders.

Given the modular architecture of the system, if any module were to be changed, this would not affect the rest. The Mattermost Bot could be replaced by a bot on any text-based communication platform. The Mental Illness Classifier could have other different disorders and the system would still work correctly.

5.5 Case Study

This section will show an example of how the system works following the defined architecture.

The user connects to his Mattermost account. This can be done from any device, as it can be used via the application or web browser.

The bot is located on the main channel on which he speaks. Therefore, everything that this user writes through this channel, the bot will listen to it and send it to the MIDS. This receives the text and tokenizes it to pass it to the Mental Illness Classifier. Here, the prediction is made by the model, and the Mental Illness Classifier sends the response to the Regulation Manager.

In case the messages written by the user are not detected as having signs of mental illness, the user will not notice anything. In case he/she writes a few messages that are detected, the bot will reply to the message to take a break, to lower the stress. If a user receives too many warnings, an interview with human resources may take place to evaluate the user's mental health status.

CHAPTER 6

Conclusions

This chapter will describe the achieved goals done by the master thesis following some the key points developed in the project and the conclusions extracted.

6.1 Achieved Goals

In this chapter, we will review everything that has been done in the project and how the proposed objectives have been completed. To do so, we will describe the key tasks that have been carried out to complete the project. The achieved goals for this project are the following ones:

- **Create a mental illness classifier.** This was the first part of the project and to see how it was completed, we will see how these tasks were carried out.

  - *Dataset collection and analysis.* The first part was to do a research on the current classifiers and obtain datasets to train our own. The classifier has been trained with two types of datasets, for stress and for depression.
  - *Preprocessing of the dataset and obtaining of the most important features.* Once the dataset is obtained, it is processed so the model can be trained by them, keeping only the part that is most useful to us. For the two types of datsets, it was necessary to do a preprocessing since many of them contain symbols, such as @ on Twitter, or words that do not provide meaning.
  - *Create a model using BERT, optimize it, and evaluate it.* We create and train the model as seen. It is tested with different datasets to see that it works correctly.
  - *Make the model able to predict correctly when the input is a text written by the user.* To check that it has developed correctly, sentences simulating the real operation have been handwritten to check if the model was able to predict correctly.

- **Develop a tool to include the model.**In this case, we selected Mattermost as the messaging application since it is the one used in the department and is widely used at the enterprise level. In this case, the tasks were:

  - *Study and analysis of existing plugins.* To begin with, an analysis of the existing Mattermost plugins has been carried out. In particular, focusing on those that are bots.
  - *Create a bot and add it to the channels.* Now it was necessary to create the bot that would listen to what the users write and mount it in the system deployed locally. This was done following the Mattermost documentation and using mmpy_bot as the framework to make it easier.

Once all the above is done, the two parts are combined and the model is added to the bot, so that it can predict if the user suffers from any disorder and acts accordingly, so that the objectives proposed for this project were completed.

6.2 Conclusion

The main objective of this project was to develop a system capable of predicting mental illnesses and responding in real-time. This is done employing a data input that is written by the user. Even having obtained a good classifier for the two mental illnesses by training the model with only one type of datset, we have preferred to improve it by training it with datsets characteristic of each disorder. The information has been obtained from social networks such as Twitter or Reddit, with a binary classification that indicates whether that tweet or post is associated with any of our mental disorders.

From here we can see, once again, the great importance of social networks, both for obtaining data in this type of work, as well as for everyone to express themselves with a level of freedom that allows classifiers of mental illnesses to be made.

The idea of integrating the classifier into a plugin of a text communication application opens up a wide variety of opportunities. Whether for companies, to be able to check the mental health of their employees and try to generate the best possible working environment, or in an environment such as universities, where it could be implemented in the internal mail, checking the status of students, which we have already seen that they are a highly vulnerable group to suffer from this type of disorder.

Therefore, this project consists of an important formative part, but it can also be something that helps people's daily life.

6.3 Future work

In this section we will see possible lines in which the project could be continued to make it more complete.

- **Add more data to train the model.** Increasing the number of datasets with which the model is trained can improve its accuracy and achieve the most realistic results possible.
- **Add more mental illnesses.** Other mental illnesses could be predicted to make the tool more complete.
- **Develop the tool for other communication platforms.** The tool could be developed for other types of applications such as Telegram or Discord, which have similar

functionality to Mattermost, but are not for work, but rather for play. This could help to detect these disorders in specific channels.

- **Implement more methods for data entry.** The text classifier could be complemented with an emotion detector either by video or voice. Thus, in case of any alarm, a response from the tool would be produced.

APPENDIX A

Ethical, economic, social and environmental aspects

A.1 introduction.

In this annex, we will see the impact of our project on ethical, economic, social, and environmental aspects.

A.2 Description of relevant impacts related to the project.

Since we deal with health-related issues, the ethical and social aspects are the most important. You have to make sure that you comply with the GDPR at all times, as a problem with the confidentiality of this data could be critical. For this reason, all datasets are pseudonymized so that you do not know whose data is being published.

On the other hand, projects like this can have a great social impact, since they can help people with these mental illnesses and even the general population, improving the detection of these.

And this is where the economic aspect comes in, since a system like this could be of interest to any company, since employees with poor mental health are less productive, as we have already seen. So, a system capable of preventing this and encouraging them to rest at

critical moments could improve their effectiveness so as to increase the overall effectiveness of the company without the need to hire new employees.

APPENDIX B

Economic budget

This appendix details an adequate budget for the project.

B.1 Physical resources

This project was carried out on a computer with an i7 processor 8th generation, 16 GB of RAM, a 100 GB SSD hard disk and an NVIDIA GeForce GTX 1050 Ti graphic card. Approximately, the cost of this laptop is 1600€.

On the software side, everything used has been open-source, so the cost is zero.

B.2 Human resources

This project has taken approximately 450 hours. This has been done for 5 months, so it would be 50 hours per month. If we estimate an average of 10 euros per hour for an graduate telecommunication engineer, it results 4500 euros, 900 euros per month.

So the total would be: 4500+1600 = 6100€.

Bibliography

[1] What is mental illness, https://www.psychiatry.org/patients-families/what-is-mental-illness, 13 de mayo de 2021.

[2] Teresa L. Scheid and Tony N. Brown. *A Handbook for the Study of Mental Health.* CAMBRIDGE UNIVERSITY PRESS, 2 edition, 2010.

[3] Richard J McNally. *What is mental illness?* Harvard University Press, 2012.

[4] Fernando Cardila Fernández, África Martos Martínez, Ana Belén Barragán Martín, Mª Del Carmen Pérez-Fuentes, Mª Del Mar Molero Jurado, and José Jesús Gázquez Linares. Prevalencia de la depresión en españa: Análisis de los últimos 15 años. *European Journal of Investigation in Health, Psychology and Education*, 5(2):267, 2015.

[5] Depression definition from World Health Organization, https://www.who.int/news-room/fact-sheets/detail/depression, 13 de mayo de 2021.

[6] Naiara Ozamiz-Etxebarria, Maria Dosil-Santamaria, Maitane Picaza-Gorrochategui, and Nahia Idoiaga-Mondragon. Niveles de estrés, ansiedad y depresión en la primera fase del brote del covid-19 en una muestra recogida en el norte de españa. *Cadernos de Saúde Pública*, 36(4), 2020.

[7] Definicion de estres NCI, https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/estres, 14 de mayo de 2021.

[8] El estrés y su salud: medlineplus enciclopedia médica, https://medlineplus.gov/spanish/ency/article/003211.htm, 14 de mayo de 2021.

[9] Candy Palomino-Oré and Jeff Huarcaya-Victoria. Trastornos por estrés debido a la cuarentena durante la pandemia por la covid-19. *Horizonte México (Lima)*, 20, 10 2020.

[10] Humberto Nicolini. Depresión y ansiedad en los tiempos de la pandemia de covid-19, 13 de mayo de 2021.

[11] La detección precoz, fundamental en depresión, https://www.redaccionmedica.com/secciones/industria/la-deteccion-precoz-fundamental-en-depresion-3872, 14 de mayo de 2021.

[12] Isabelle Hupont. *Affective computing: emotional facial sensing and multimodal fusion.* PhD thesis, Universidad de Zaragoza, 2010.

[13] Melisa Bubonya, Deborah A Cobb-Clark, and Mark Wooden. Mental health and productivity at work: Does what you do matter? *Labour economics*, 46:150–165, 2017.

[14] Eero Lahtinen, Ville Lehtinen, Eero Riikonen, and Juha Ahonen. Framework for promoting mental health in europe. pages 33–35, 1999.

[15] James Morrison. *DSM-5 Guía para el diagnóstico clínico.* Editorial El Manual Moderno, 2015.

[16] Kehan Li. *Identificación de emociones a través del procesado digital de imágenes.* PhD thesis, 2019.

[17] Mario López Rodríguez. Identificación de características sonoras en enfermedades neurodegenerativas para detección precoz. 2019.

[18] Blanca Fanny Chalén Pacay and Juan Carlos Camacho Girón. *Desarrollar un prototipo de reconocimiento facial basado en Machine Learning para detectar estado de Somnolencia en conductores de una cooperativa de transporte.* PhD thesis, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas ..., 2020.

[19] Nawshad Farruque, Osmar Zaiane, and Randy Goebel. Augmenting semantic representation of depressive language: From forums to microblogs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 359–375. Springer, 2019.

[20] Nikolaos Koutsouleris, Dominic B. Dwyer, Franziska Degenhardt, Carlo Maj, Maria Fernanda Urquijo-Castro, Rachele Sanfelici, David Popovic, Oemer Oeztuerk, Shalaila S. Haas, Johanna Weiske, Anne Ruef, Lana Kambeitz-Ilankovic, Linda A. Antonucci, Susanne Neufang, Christian Schmidt-Kraepelin, Stephan Ruhrmann, Nora Penzel, Joseph Kambeitz, Theresa K. Haidl, Marlene Rosen, Katharine Chisholm, Anita Riecher-Rössler, Laura Egloff, André Schmidt, Christina Andreou, Jarmo Hietala, Timo Schirmer, Georg Romer, Petra Walger, Maurizia Franscini, Nina Traber-Walker, Benno G. Schimmelmann, Rahel Flückiger, Chantal Michel, Wulf Rössler, Oleg Borisov, Peter M. Krawitz, Karsten Heekeren, Roman Buechler, Christos Pantelis, Peter Falkai, Raimo K. R. Salokangas, Rebekka Lencer, Alessandro Bertolino, Stefan Borgwardt, Markus Noethen, Paolo Brambilla, Stephen J. Wood, Rachel Upthegrove, Frauke Schultze-Lutter, Anastasia Theodoridou, Eva Meisenzahl, and PRONIA Consortium. Multimodal Machine Learning Workflows for Prediction of Psychosis in Patients With Clinical High-Risk Syndromes and Recent-Onset Depression. *JAMA Psychiatry*, 78(2):195–209, 02 2021.

[21] Elena Smets, Pierluigi Casale, Ulf Großekathöfer, Bishal Lamichhane, Walter De Raedt, Katleen Bogaerts, Ilse Van Diest, and Chris Van Hoof. Comparison of machine learning techniques for psychophysiological stress detection. In *International Symposium on Pervasive Computing Paradigms for Mental Health*, pages 13–22. Springer, 2015.

[22] Introduccion al-machine learning, `http://smartbasegroup.com/introduccion-al-machine-learning/`, 16 de mayo de 2021.

[23] Python, `https://www.python.org/`, 16 de mayo de 2021.

[24] Pandas documentation, https://pandas.pydata.org/docs/user_guide/index.html, 16 de mayo de 2021.

[25] Numpy doc, https://numpy.org/doc/stable/user/whatisnumpy.html, 16 de mayo de 2021.

[26] SciPy doc, https://docs.scipy.org/doc/scipy/reference/tutorial/general.html, 16 de mayo de 2021.

[27] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly, 2011.

[28] Huggingface Transformers, https://huggingface.co/transformers/index.html, 18 de mayo de 2021.

[29] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[31] Mattermost documentation, https://docs.mattermost.com/overview /product.htmlabout-the-mattermost-open-source-project, 16 de mayo de 2021.

[32] Elsbeth Turcan and Kathy McKeown. Dreaddit: A Reddit dataset for stress analysis in social media. In *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*, pages 97–107, Hong Kong, November 2019. Association for Computational Linguistics.

[33] Mike Thelwall. Tensistrength: Stress and relaxation magnitude detection for social media texts. *Information Processing & Management*, 53(1):106–121, 2017.

[34] Depression_detection_using_machine_learning_2021, https://github.com /patilgirish815/$Depression_Detection_Using_Machine_Learning$, $12 de junio de 2021$.

[35] gulshan0297-depressiondataset, https://github.com/Gulshan0297/DepressionDataset, 12 de junio de 2021.

[36] Common loss functions in machine learning, https://towardsdatascience.com/common-loss-functi 05 de junio de 2021.

[37] Elizabeth Cristina Sosa Garcia. *Qué es el estrés ocupacional, enfermedades derivadas y reconocidas por la legislación Colombiana*. Revista CES Salud Publica, 2021.

[38] Jorge Pla Vidal. Depresión: Causas, síntomas y tratamiento. clínica universidad de navarra, https://www.cun.es/enfermedades-tratamientos/enfermedades/depresion, 13 de mayo de 2021.

[39] Cómo se diagnostica la depresion, https://www.msdsalud.es/cuidar-en/depresion/informacion-basi 14 de mayo de 2021.

[40] test de detección del nivel de estrés, https://www.teknon.es/es/pruebas-diagnosticas/laboratorio-analisis-clinicos/test-det 14 de mayo de 2021.

[41] Senthil Sriramprakash, Vadana D Prasanna, and OV Ramana Murthy. Stress detection in working people. *Procedia computer science*, 115:359–366, 2017.

[42] Mattermost plugin documentation, https://developers.mattermost.com/contribute/server/plugins 16 de mayo de 2021.

[43] Mattermost bots documentation, https://docs.mattermost.com/deployment/bots.html, 16 de mayo de 2021.