UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT AND EVALUATION OF A SARCASM DETECTION ALGORITHM BASED ON MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES

Julián Amigó Francés

2019

TRABAJO DE FIN DE GRADO

Título:	Desarrollo y Evaluación de un Algoritmo de detección de Sarcasmo utilizando técnicas de Aprendizaje Automático y Procesamiento del Lenguaje Natural
Título (inglés):	DEVELOPMENT AND EVALUATION OF A SARCASM DETECTION ALGORITHM BASED ON MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES
Autor:	Julián Amigó Francés
Tutor:	Óscar Araque
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 GRADO

DEVELOPMENT AND EVALUATION OF A SARCASM DETECTION ALGORITHM BASED ON MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES

Enero 2019

Resumen

La inteligencia artificial está invadiendo nuestras vidas cada vez más. Muchísimas tecnologías están cargadas de Inteligencia Artificial (IA) con el fin de mejorar la calidad de un servicio a un usuario. Nuestro teléfono móvil está repleto de esta IA, mejorando así muchas opciones de búsqueda, facilitándonos el uso de la cámara o incluso permitiéndonos hablar con el dispositivo sin necesidad de tocarlo.

Estas comodidades software han ido evolucionando gracias a las empresas que lo desarrollan. Pero estas no solo usan la IA para el usuario final, también aprovechan esta tecnología en beneficio propio. Mediante el desarrollo de herramientas que usan Inteligencia Artificial son capaces de detectar el grado de satisfacción de un cliente frente a un servicio o producto. Es aquí cuando las empresas emplean el análisis de opiniones o de sentimientos.

Cuando un usuario transmite una idea a través de comentarios en foros u otras redes sociales, mediante el análisis de sentimientos o "Sentiment Analysis" del inglés, las empresas pueden mejorar sus servicios o productos finales. Estos sistemas de detección de sentimientos analizan los comentarios y sacan una categoría de sentimiento final. Sin embargo, muchas veces un usuario no siente lo que textualmente escribe, es más, muchas veces el sentimiento de una persona es totalmente lo contrario a lo que está escribiendo. Es en estos casos, sobretodo si se quiere transmitir un sentimiento negativo, cuando el usuario hace uso de la ironía o el sarcasmo.

El objetivo principal de este trabajo es analizar este tipo de comportamientos, desarrollando un sistema que sea capaz de detectar estos casos en los que el usuario transmite un sentimiento opuesto a lo que aparenta escribir.

Analizando los textos, podremos detectar sentimientos opuestos en un mismo comentario o signos de puntuación típicos con los que detectar cuando un comentario es sarcástico o no. Para desarrollar esta tarea, se emplea el lenguaje de Python, que nos facilita la utilización de librerías como Scikit-Learn con herramientas para técnicas de Procesamiento de Lenguaje Natural ("Natural Language Processing" (NLP) del inglés) y Aprendizaje Automático ("Machine Learning" (ML) del inglés).

Se desarrollará por lo tanto, un sistema que diferencie, con la mayor exactitud posible, cuando un comentario es sarcástico o no. Para la prueba y evaluación de este sistema, se

analizarán diferentes características extraídas de los textos con el fin de obtener cuáles de ellas dan un mejor resultado a nuestro modelo. El entrenamiento de este modelo con las características finalmente escogidas, se realizará con un algoritmo de clasificación.

Como fuente de datos se ha escogido dos conjuntos: el primero recoge comentarios del foro de "Reddit" y el segundo de las páginas web de "The Onion" y de "HuffPost". En ambos casos, los comentarios están ya etiquetados como sarcásticos o no sarcásticos, ayudándonos así en la fase de entrenamiento de nuestro modelo.

Palabras clave: Procesamiento de Lenguaje Natural, Aprendizaje Automático, Scikit-Learn, Python, Sarcasmo.

Abstract

Artificial intelligence is invading our lives more and more. Many technologies are loaded with Artificial Intelligence (AI) in order to improve the quality of a service to a user. Our mobile phone is full of this AI, thus improving many search options, making it easier for us to use the camera or even allowing us to talk to the device without touching it.

These software commodities have been evolving thanks to the companies that develop it. But they don't just use AI for the final user, they also take advantage of this technology for their own benefit. Through the development of tools that use Artificial Intelligence, they are able to detect the level of customer satisfaction about a service or a product. It is here when companies apply the analysis of opinions or feelings.

When a user conveys an idea through comments on forums or other social networks, with sentiment analysis, companies can improve their services or products. These sentiment detection systems analyze the comments and get a final sentiment category. However, many times a user doesn't feel what is written, what's more, many times a person's feeling is the complete opposite of what he or she is writing. It is in these cases, especially if you want to convey a negative feeling, when the user makes use of irony or sarcasm.

The main objective of this work is to analyse this type of behaviour, developing a system that is capable of detecting these cases in which the user transmits an opposite feeling to what he or she seems to write.

By analysing the texts, we will be able to detect opposite feelings in the same text or typical punctuation marks with which we can detect when a comment is sarcastic or not. To develop this task, Python language is used, which facilitates the use of libraries such as Scikit-Learn with tools for Natural Language Processing (NLP) and Machine Learning (ML) techniques.

A system will therefore be developed that differentiates, as accurately as possible, when a comment is sarcastic or not. For the testing and evaluation of this system, different features extracted from the texts will be analyzed in order to obtain which of them give a better result to our model. The training of this model with the final features chosen, will be carried out with a classification algorithm.

Two datasets have been chosen as data sources: the first collects comments from the "Reddit" forum and the second from the "The Onion" and "HuffPost" websites. In both

cases, the comments are already labeled as sarcastic or non-sarcastic, thus helping us in the training phase of our model.

Keywords: Natural Language Processing, Machine Learning, Scikit-Learn, Python, Sarcasm.

Agradecimientos

A mi tutor Óscar Araque, por la confianza depositada en mí para sacar este trabajo adelante y por estar disponible siempre que lo necesitaba. Ha sido un gran tutor.

A Carlos A. Iglesias, por las facilidades que me dio al yo tocar en la puerta de su despacho por primera vez, explicarme con detalle todo lo que el departamento contenía y abrirme un gran abanico de posibilidades.

A mis padres, por ayudarme a lo largo de todos estos años y confiar en mí en todo momento, y recordarme todo lo que era capaz de hacer.

A mi gran grupo formado en la universidad. Amistades que son para toda la vida.

A mis amigos de Madrid y de Tenerife por aguantarme.

Y a Carmen, por cogerme el teléfono todos los días y hacerme feliz, y sobretodo por siempre creer en mí.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

The term Big Data, nowadays, is already known almost everywhere in the world. When we talk about Big Data we mean huge amounts of data sets that have complex structure with the difficulties of storing, analyzing and visualizing for further results. In 2013 it was estimated that 5 exabytes of data were created by human. Currently, that amount is created in just two days [34].

However, the important thing is not the amount of data obtained, but what organizations do with this data. Machine Learning provide us those tools for data analysis. It is programming computers to optimize a performance criterion using example data or past experience.

Machine Learning (ML) [8, 25] is a branch of artificial intelligence that employs a set of algorithms that give computers and computer systems the necessary autonomy to learn from their mistakes and optimise their successes without human intervention.

Machine Learning is having an impact on all areas of our lives: social, work, medical, etc. It is present in our daily lives and the majority of the population still does not know it. A smartphone is currently full of AI (Artificial Intelligence), from the web browser to correction on keyboards, where they learn, among other functions, which are the words that the users uses most frequently. Netflix's algorithms use ML to give recommendations to the

users of the films they may like the most, depending on the films they have already seen on this entertainment platform.

In addition, it affects almost directly the jobs of citizens, as in some cases machines will do the job faster than people, thus eliminating the need for this job, that is, replacing the person by the machine. This may frighten society, as Forrester (company who works with business and technology leaders to develop customer-obsessed strategies that drive growth) warns, which says that by 2025, in the United States, the impact of ML and AI will reduce 7% of jobs [21]. On the other hand, in the last edition of the World Economic Forum in Davos, it was said that AI, robotics and nanotechnology would destroy around 5 million jobs in the next 4 years, generating a little more than 2 million new jobs.

One of the most important fields to which AI applies is Natural Language Processing (NLP). This (combined with applied linguistics), as its name suggests, has as its objective the understanding of human language by machines, thus being able to carry out tasks such as automatic translation among others. The analysis of opinions and feelings is also a very important part of the NLP, which many companies use to capture the satisfaction of their customers with certain products. This is known by the term "Sentiment Analysis". Analyzing the comments of their clients in forums, social networks, etc, they can know more or less how a product has pleased a certain type of client, allowing them to act in negative situations with an improvement of the product. This is only one of the thousands of scenarios in which the analysis of feelings in comments can be useful.

But sometimes a person's opinion is not exactly what he is saying (in this case what he is writing), it is more, sometimes it means quite the opposite. The use of irony is very common to express an opinion, and more specifically, the use of sarcasm, to indicate that you completely disagree with something. In the area of sentiment analysis, sarcasm plays a important role as an interfering factor that can flip the polarity of a message [16]. To avoid this problem, this project will define a model capable of detecting (as accurately as possible) when a comment is sarcastic or not.

Currently, the automatic detection of irony in texts is an open topic, which is being addressed by different research groups. Irony and sarcasm are very similar terms and they are often confused. Actually, sarcasm is a kind of irony, so all instances of sarcasm are irony but not all instances of irony are sarcasm. An Irony, as defined by the University of Cambridge [5], is the use of words that are the opposite of what you mean, as a way of being funny , while a Sarcasm [6] is the use of remarks that clearly mean the opposite of what they say, made in order to hurt someone's feelings or to criticize something in a humorous way.

Sarcasm is easy to detect by tone of voice or expression, but not so easy when it comes to something in written form. This project is centered around this challenge. A simple phrase

like “It’s been amazing!” can mean it has been something very positive or something very negative. In this case, it would be necessary to analyze the context of this comment and see if it is something positive or negative and contrast it with the phrase we are analyzing. On the other hand, the use of exclamation marks or the use of several letters to show a prolonged syllable (for example: ”Sooorryyyy”), are other tricks to distinguish whether a comment is sarcastic or not.

Teaching a machine to understand how context can affect tone is a quite complicated task. There are other cases where with the simple analysis of the sentence, you can estimate if a comment is sarcastic or not: ”My flight is delayed. Great!”. Here, most humans could quickly interpret the person as being sarcastic because for most people the delay is a negative experience. The machine, compare this negative situation with the positive situation that is identified in ”Great!” and would identify this comment as sarcastic.

Other techniques could be applied to make it easier for the machine to detect that a comment is sarcastic. For example, looking if the speaker is usually sarcastic or not, if use an aggressive tone (these cases in particular, more sarcasms are used), identify the gender of the text (usually funny or satirical texts are more likely to include sarcasm), and so on.

1.2 Project goals

In essence, the main objective of this work will be to develop a system, maximixing its accuracy, to detect if a text is sarcastic or not, applicable to any field that requires knowing the exact opinion of a person and sarcasm may affect this knowledge. In order to achieve this objective, two datasets offered on the Internet for this type of task have been analysed, containing comments that have been described by humans as sarcastic and non-sarcastic.

Different preprocesses and various feature extractions will be evaluated, to then train an algorithm and choose the one that suits us best, i.e. the best possible precision result (all this will be explained in more detail in the following chapters).

The steps that will be followed in the project to achieve the objective are as follows:

- Preprocessing the dataset, creating a balanced training and testing dataset.
- Extraction of features to be analyzed by the classifier.
- Testing different algorithm and different types of preprocessing in order to use the one with higher accuracy.
- Evaluation of the final model.

1.3 Structure of this document

In this section we provide a brief explanation of the chapters included in this project. It will be structured as folows:

Chapter 1: On the one hand, this chapter describes the context of the main theme of the project and the importance of this issue today. On the other hand, the main objectives to be achieved will be explained.

Chapter 2: we will explain the technologies and libraries that have been necessary to implement for the proper functioning of the model. In addition, previous work related to this will be analyzed.

Chapter 3: different procedures used for the development of the model will be explained as well as the results of each one of them.

Chapter 4: the datasets used for model training will be explained, as well as the final methodology implemented.

Chapter 5: shows the results obtained in the project and elaborate a discussion about these results.

Chapter 6: discusses the conclusions drawn from this project, achieved goals and suggestions for a future work.

CHAPTER 2

Enabling Technologies

2.1 Introduction

This chapter will explain the technologies used for the development of our model. The main technologies to be highlighted are Machine Learning (ML) and Natural Language Processing (NLP). As we have explained in Section 1.1, both together are very useful when we want to get accurate information from text. First, we apply NLP to extract features from the text and then we apply ML algorithm to make a predictive model. Although the needs of NLP were clear from the very beginning, the strategy adopted by the research community was to tackle syntax first, for the applicability of machine learning techniques then [14].

Furthermore, we will talk about libraries like Numpy, Pandas or Scikit-learn (among others), all of them designed for the Python language. Scikit-learn includes several algorithms for classification, regression and group analysis. Pandas, which depends on Numpy, is used for data manipulation and analysis.

Finally, mention will be made of other previous works, related to the detection of irony or sarcasm.

2.2 Machine Learning

2.2.1 Introduction

There are many definitions that describe this cutting-edge technology today. One of them, which is quite interesting, is the one described in [33] which says that "Machine Learning consists of programming computers to optimise a performance criterion using example data or past experiences". As reflected in this book, this technology allows the computer to be able to improve its prediction based on example data and on how many times it has been trained, i.e. past experiences.

The main idea of the ML [19] is to transform input data into output data, depending on the task to be performed and the algorithm to be used. To differentiate these tasks we can separate these algorithms into three categories: Supervised Learning, Unsupervised Learning and Reinforcement Learning.

Within each of these categories we can differentiate several types of problems and they will be easier to solve in one category than in another. There are also numerous algorithms within each category, such as K-means and PCA for Unsupervised Learning, or Decision Tree and Logistic Regression for Supervised Learning.

2.2.2 Machine Learning Categories

Within each of the above-mentioned categories, which differ from the data point of view, there is another classification from the type of data output point of view. In this section, apart from talking about these three categories, we will also differentiate subcategories according to the task they perform.

- **Supervised Learning:** The term supervised refers to a set of samples, where the outputs (labels) corresponding to each of the input data are already known. The main objective of this category is to train a model from already labeled input data (that output data is known) in order to predict the corresponding output in new input data in the future [30]. A schematic representation can be seen in Figure 2.1.

  E-mail spam filtering, is a clear example where supervised algorithms are used. In this case, we can train a model with emails correctly labeled as spam or as not spam, to predict spam emails in the future and redirect them to an unwanted folder or directly to the trash. This filtering task, where discrete class labels are used, is also called a classification task. Another subcategory within supervised learning is regression. These subcategories are explained in more detail below.

  - *Classification:* The main objective of this type of supervised training is to predict

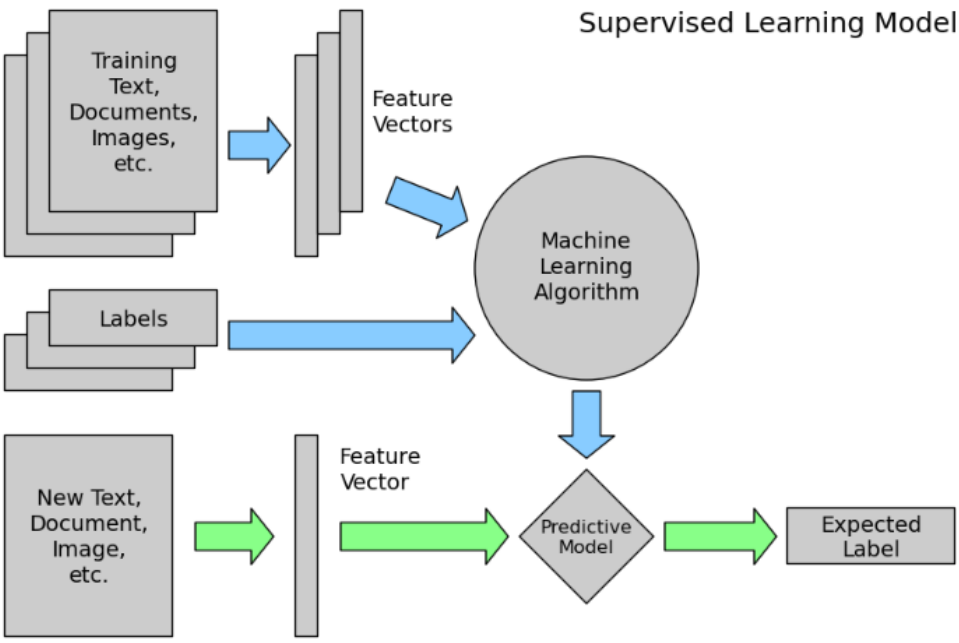

Figure 2.1: Supervised Machine Learning Model [15]

the labels of new instances based on previous observations. The previous example where spam mail was detected is a case of binary classification since only two tags are used: spam or non-spam. An example of multi-class classification could be: handwritten character recognition.

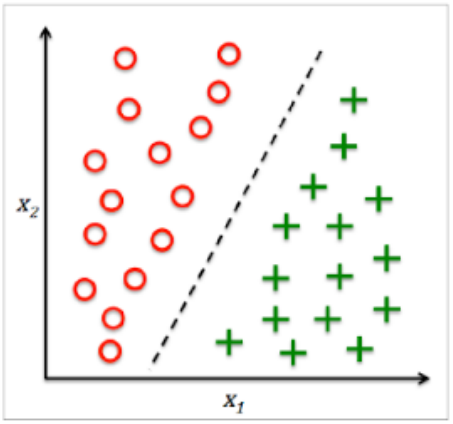

Figure 2.2: Graphical Representation of the Classification model

– *Regression:* The Regression model consists of the prediction on continuous variables. In this case, it allows you to make predictions from data by learning the relationship between features of your data and some observed, continuous-valued

response. Therefore, we are given a number of predictor variables and a continuous response variable (outcome), and we try to find a relationship between those variables that allows us to predict an outcome.

An example where we could use a Regression algorithm would be to predict the score of students. With the time you have spent to perform the test and its score, we can train a model that uses study time to predict other future scores.

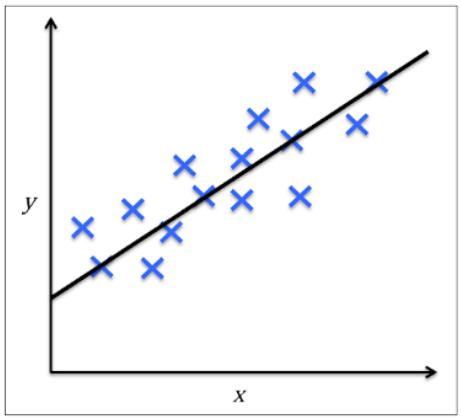

Figure 2.3: Graphical Representation of the Regression model

- **Unsupervised Learning:** In contrast to previous training, the training of Unsupervised learning models is done with data not previously labelled. Here we can explore the structure of our data without the need for a known outcome variable. Within this category, we can also differentiate by the type of tasks it performs, other subcategories. These include Clustering and Dimensionality Reduction.

  - *Clustering:* This technique allows us to organize a lot of information in groups without having any previous knowledge of the group to which they belong. Clustering is very useful to structure the information and look for significant relationships between the data. in the following image we see how with this method, we have been able to organize the information in three groups, depending on the similarities between the data, choosing in this case two important characteristics (x1, x2).

  - *Dimensionality Reduction:* This technique is commonly used to eliminate noise in information during pre-processing and to reduce information in smaller subspaces when the initial information provided has large dimensions. As a result, this technique facilitates, among other things, the classification, comprehension and visualization of high-dimensional data [23]

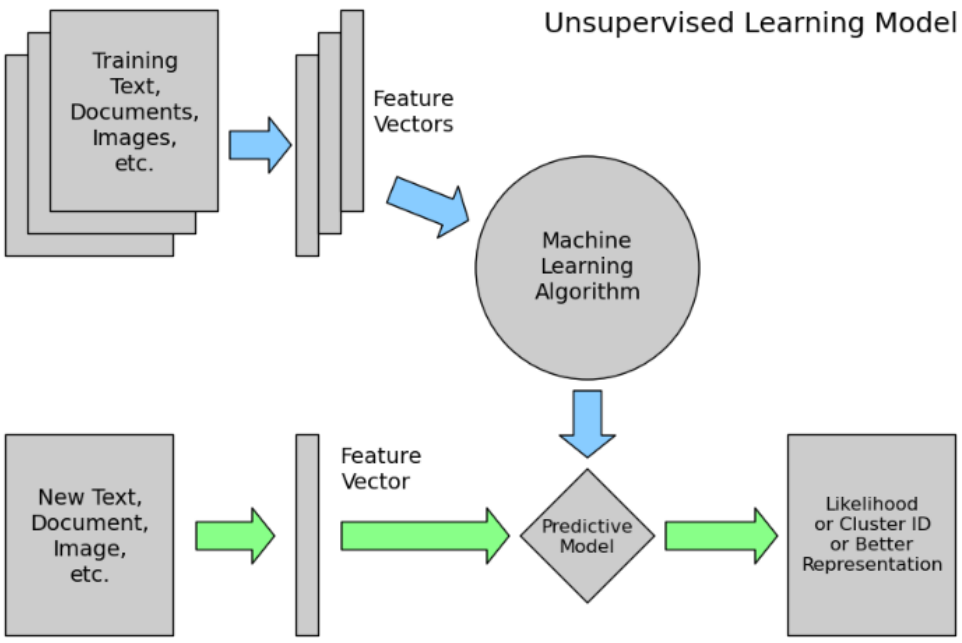

Figure 2.4: Unsupervised Machine Learning Model [15]

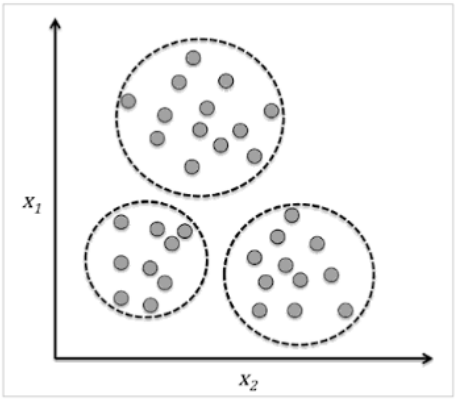

Figure 2.5: Graphical Representation of the Clustering Model

- **Reinforcement Learning:** The objective in this training is to develop a system that improves its performance thanks to interactions with the environment. The feedback that this system receives is not a label, but a reward function that specifies how well the action has been performed. A basic example could be a chess engine. In this case, a series of moves are decided depending on the environment (the board), and at the end of the game a reward is given to the system telling it if it has won or lost the game.

Figure 2.6: Reinforcement Learning model

2.2.3 Machine Learning Algorithms Examples

In this section, we will name some of the most important algorithms used to make Machine Learning, specifying if they are supervised or unsupervised algorithms.

- **Naïve Bayes Clasification:** Naive Bayes methods are a set of supervised learning algorithms based on the application of Bayer. This is a theorem with very non-sophisticated ("naive") independent suppositions between the labels or properties [28].

- **Decision Trees:** This Supervised Learning method is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance-event outcomes, resource costs, and utility. Take a look at the image to get a sense of how it looks like.They present an appearance like this:

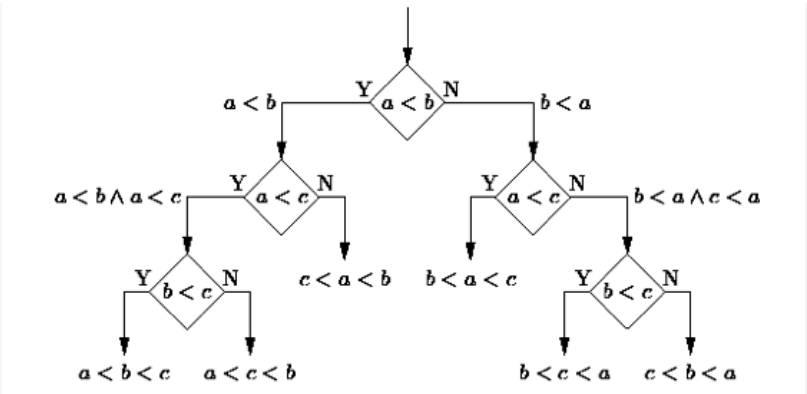

Figure 2.7: Decision Tree [29]

- **Logistic Regression:** This is another example of a Supervised Learning method. In this case, it is used to predict the outcome of a categorical variable (which can

adopt a limited number of categories) based on independent or predictive variables. In other words, it is a statistical method used to determine the contribution of various factors to a pair of results. An explanatory and easy-to-understand example might be as follows [26]:

A group of people who share similar demographic information and purchase products from a company. Modeling data to relate it to a particular outcome (such as the purchase of a product) will show how demographic information contributes to the likelihood that someone will buy that product.

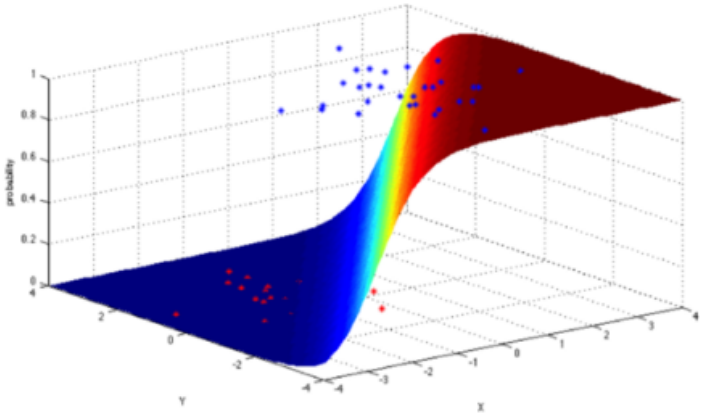

Figure 2.8: Logistic Regression

- **K-means:** This is an unsupervised algorithm, within the clustering subcategory which makes a grouping of data in a predefined number of clusters [17]. The main idea is to define k centroides (one for each group, cluster) and then assign each element of the database to the nearest centroid. Then, the new centroid of each group is recalculated to redistribute all the elements of the database according to the nearest new centroid.

- **PCA:** Also within unsupervised algorithms, PCA (Principal Component Analisys) performs a reduction from some initial variables to the minimum possible number of them. In other words, it introduces a smaller representation of the data set and finds a linear combination sequence of variables that explain the maximum variance and summarize most of the data information.

2.3 Scikit-Learn

First of all, the global package which this Scikit-Learn library belongs should be mentioned. The Spicy Ecosystem [4] is a collection of open source software for scientific computing in Python. This Ecosystem includes general and specialised tools for data management and computation. Within this ecosystem, the collection that provides algorithms and tools for Machine Learning is Scikit-Learn [28]. Scipy Ecosystem core package are the following:

- ***Python:*** a general-purpose programming language, powerful enough to write code for large applications.
- ***Numpy:*** the fundamental package for numerical computation. More details will be discussed below.
- ***Matplotlib:*** a popular plotting package, that provides publication-quality 2D plotting as well as a simply 3D plotting.

Some of the key packages provided by Scipy Ecosystem and used for this project are the following:

- ***Pandas:*** providing easy to use data structure. More details will be discussed below.
- ***IPython:*** a rich interactive interface, letting you quickly process data and test ideas.
- ***Jupyter Notebook:*** provides the functionalities of iPython and allows us an easy writing for the development and execution of the code.

2.4 Numpy

One of the most important modules in Python is Numpy [2]. It was designed to provide Python with calculation capabilities similar to other software such as MATLAB. Numpy is in charge of adding all the mathematical and vector capabilities to Python making it possible to operate with any numeric data or array.

Its main mission is to add mathematical and vector capability to Python enabling operation with numeric data or arrays.

2.5 Pandas

Pandas [3] is a python library for data analysis, which provides flexible data structures and allows you to work with them very efficiently. The two main Pandas structures are:

- **Series:** They are one-dimensional arrays with indexation (arrays with index or tagged), similar to dictionaries. They can be generated from dictionaries or lists.

- **Dataframe:** It is the basic data structure of this library, similar to relational database tables such as SQL. It is an ordered collection of columns with names and types, where a single row represents a single case and the columns represent particular attributes.

Pandas makes it easier for us to clean the raw data so that they are suitable for analysis. Pandas carries out important tasks, such as aligning data for their comparison, merging data sets, managing lost data, etc., but it has become a very important library for processing high-level data in Python (i.e. statistics). Therefore, as a summary to say that pandas provides us with the data structures and functions necessary for data analysis.

2.6 NLTK

As we have already explained in section 1.1, NLP (Natural Language Processing) is a way for computers to analyze, understand, and derive meaning from human language in a smart and useful way. This field focuses on research into computationally effective mechanisms for communication between people or between people and machines by means of natural languages.

In order to process the natural language there are many tools, one of them is NLTK Natural Language Toolkit [1] which has a collection of packages and Python objects very adapted for NLP tasks.

NLTK is a platform used for building Python programs that work with human language data for applying in statistical natural language processing. It is free, opensource, easy to use and well documented. This tool helps the computer to analysis, preprocess and understand the written text.

Here are some language processing tasks we can perform and the NLTK modules commissioned for them, with examples of funcionality:

Language processing task	NLTK modules	Functionality
Accessing corpora	corpus	standardized interfaces to corpora and lexicons
String processing	tokenize, stem	tokenizers, sentence tokenizers, stemmers
Collocation discovery	collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	tag	n-gram, backoff, Brill, HMM, TnT
Machine learning	classify, cluster, tbl	decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	chunk	regular expression, n-gram, named-entity
Parsing	parse, ccg	chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	sem, inference	lambda calculus, first-order logic, model checking
Evaluation metrics	metrics	precision, recall, agreement coefficients
Probability and estimation	probability	frequency distributions, smoothed probability distributions
Applications	app, chat	graphical concordancer, parsers, WordNet browser, chatbots
Linguistic fieldwork	toolbox	manipulate data in SIL Toolbox format

Figure 2.9: Language processing tasks and corresponding NLTK modules [36].

2.7 GSITK

GSITK [10] is a library on top of scikit-learn that facilitates the development of projects related to automatic learning of NLP. It uses numpy, pandas and related libraries to easy the development. This library handles datasets, features, classifiers and evaluation techniques, so writing an evaluation line is quick and easy.

Some of the tools developed in this library and used in the project are:

- *Word2VecFeatures:*

  Word embedding [24] is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Loosely speaking, they are vector representations of a particular word.

  Word2Vec [32] is a method to construct such an embedding; it is one of the most popular approaches that allows modeling words as vectors. It is based on the two following methods (both involving Neural Networks): Skip Gram and Common Bag Of Words (CBOW), to perform the computation of the distributed representations. Our objective is to have words with similar context occupy close spatial positions. Without going into much detail, the following is a neural network of the CBOW method:

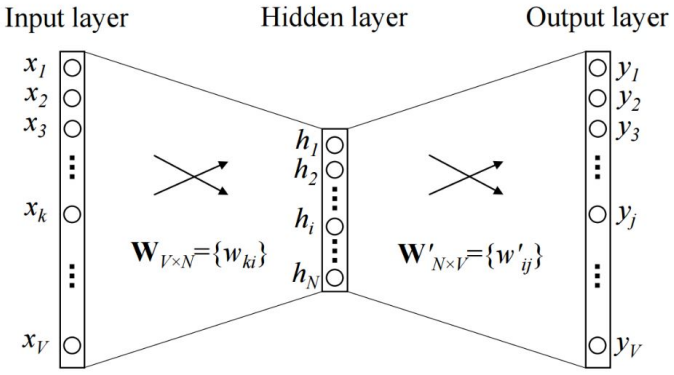

Figure 2.10: Simple CBOW model with only one word in the context

This method takes the context of each word as the input and tries to predict the word corresponding to the context. The hidden layer neurons just copy the weighted sum of inputs to the next layer. But, the above model used a single context word to predict the target. We can use multiple context words to do the same.

We have seen how word representations are generated using the context words. But, we can use the target word (whose representation we want to generate) to predict

the context and in the process, we produce the representations. In this case we use Skip-Gram model as we show below:

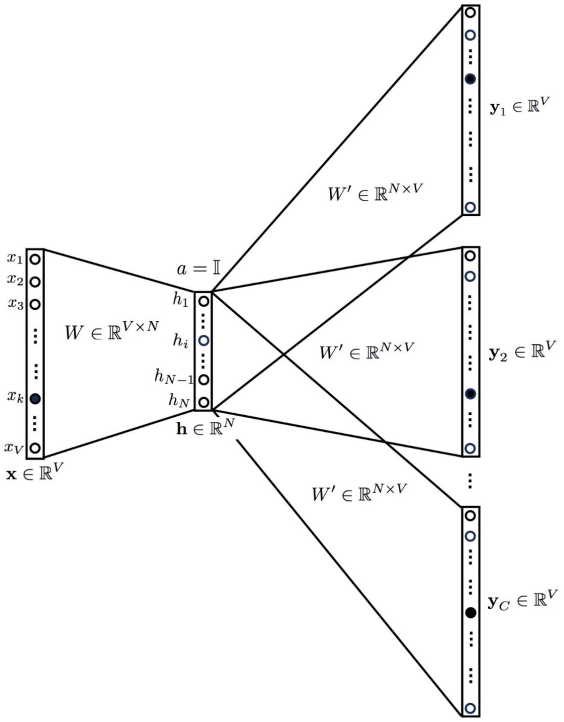

Figure 2.11: Skip-Gram model

Both have their own advantages and disadvantages. Skip Gram works well with small amount of data and is found to represent rare words well. On the other hand, CBOW is faster and has better representations for more frequent words.

- *SIMON features extractor:*

  As we have already seen in "word2Vec", a very useful technique is to represent words in the form of a fixed vector. In this case, with SIMON feature extractor [11], we can represent these words but adding one more functionality: as we will see below, subjective information of feelings is included in the proposed model. In general, word embeddings contains semantic and syntactic information, but they do not contain specific sentimental information, as no sentiment-related signal has been included in the training process. Therefore, additional information sources must be included in the feature extraction process. A sentiment lexicon vocabulary is required.

  What this tool proposes is the representation of a certain word, which can be outside the vocabulary of the lexicon, through a projection to a set of sentimental words extracted from a sentimental lexicon. A certain word is represented by its similarities with the selection of words from the lexicon.

As we can see in the following image , two words are taken from the sentimental lexicon: good and bad. Between these two, there is a two-dimensional space. The values of that space correspond to the semantic similarity of each word with the words of the lexicon.

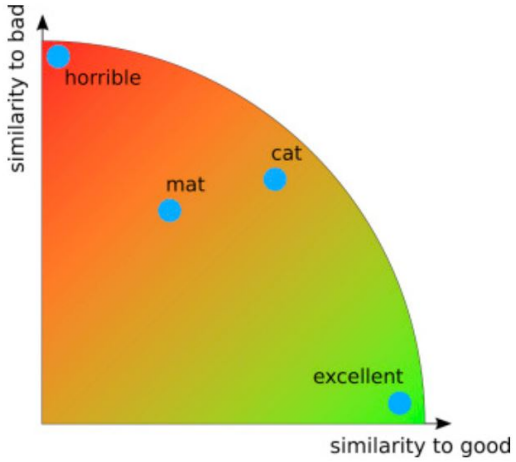

Figure 2.12: Conceptual diagram of word projection over a lexicon formed only by the set of words (good, bad)

The proposed method for extracting similarity features can be expressed as an algorithm. We call this method SIMilarity-based sentiment projectiON (SIMON). Similarities are computed against a selection of lexicon words, and a max function is applied column-wise, obtaining a feature vector.

2.8 Related Work

In this chapter we will talk about other works related to this topic of Sarcasm detection. In this way we will have a broader vision of how other Sarcasm detection models have been carried out and how their evaluations have been, in order to make comparisons with our model.

The development of our proyect is based on two papers: the one provided by the SARC paper [22], in which no model has been developed (it is a corpus, not a method for the detection of sarcasms) and the one provided by [27]. The model here will be proposed is made with two datasets that this two papers have provided us.

The SARC (A Large Self-Annotated Corpus for Sarcasm) corpus propose good baselines for model development. It presents different datasets to analyse and mentions many works related to the detection of irony or sarcasm that should be highlighted. For the development of this project, the following works mentioned in the corpus have been analyzed:

1. **Sparse, Contextually Informed Models for Irony Detection [37]**: This paper developed by Byron C. Wallace (University of Texas, Austin) and Do Kook Choe and Eugene Charniak (Brown University) propose a rather novel strategy that takes into account the context of the comments. They thus purpose for verbal irony classification, combining noun on phrases and sentiment extracted from comments with the forum type to which they were posted. For the development of this task they decide to take the comments published in Reddit [38].

   They decide to take this platform because here there are user communities focused on specific topics, where those users (subreddits) within each community differ in the side to which they belong. For example, progressive or conservative subreddits, or atheism and Christianity subreddits. In this way, the main objective here will be to recognize the verbal irony in the comments published in those forums, automatically differentiating whether the user who posts a comment intends it to be ironic or not.

   The following features to be analyzed are highlighted: "bag-of-NNP" (NNP is Noun Phrase, it consist of a pronoun or noun, which is called the head, and any dependent words before or after), characteristics that indicate whether the inferred feeling for a given phrase was positive or negative; interaction characteristics, (for example, if a given phrase mentions ObamaCare and was published in a conservative subreddit); and also three-way interactions, crossing feelings with NNPs and subreddits. Numerous performances are performed with different datasets (politics, religion, etc.) bringing out the most outstanding characteristics. They conclude that contextual analysis is very important for the detection of irony.

2. **A multidimensional approach for detecting irony in Twitter [31]**: In this case, the corpus elaborated by Antonio Reyes, Paolo Rosso and Tony Veale, use the social network Twitter as a source for the search of comments for its dataset. For the elaboration and organization of the data set, they extract from this social network 40.000 tweets, where 10,000 are labeled as ironic and the remaining 30.000 as non-ironic. To perform this labeling, the first group of 10.000 tweets is made with comments that contain #irony hastag. Subsequently, another 3 groups of 10,000 tweets each are made, which will be labeled as non-sarcastic and will be tweets extracted from the #humor #politics and #education hastags.

   They propose a model that is organized according to four types of conceptual feature: signatures, unexpectedness, style, and emotional scenarios. Each feature, save for unexpectedness, is represented with three dimensions; unexpectedness is represented with just two dimensions.

- *Signatures:* concerning pointedness, counter-factuality, and temporal compression. This feature is focused on exploring irony in terms of specific textual markers or signatures.
- *Unexpectedness:* concerning temporal imbalance and contextual imbalance. They note that surprise is a key component of irony, and even goes as far as claiming that unexpectedness underlies all ironic situations. As a consequence, they conceive the unexpectedness feature as a means to capture contextual imbalances in an ironic text.
- *Style:* as captured by character-grams (c-grams), skip-grams (s-grams), and polarity skip-grams (ps-grams). They emphasize that the concept of style refers textual elements that express relatively stable features of how a text is appreciated, and which might thus allow us to recognize stylistic factors that are suggestive of irony.
- *Emotional scenarios:* concerning activation, imagery, and pleasantness. This feature attempts to characterize irony in terms of elements which symbolize abstractions such as overall sentiments, attitudes, feelings and moods, in order to define a schema of favorable and unfavorable contexts for the expression of irony.

They evaluate the model in two ways: (1) by considering the appropriateness or representativeness of different patterns to irony detection; and (2) by considering the empirical performance of the model on a tweet classification task.

As will be commented in the following chapters, when evaluating a model a final score is obtained according to the true positives, false negatives, false positive and true negatives obtained. In this way, the following parameters are formed: Recall, Precission and F1-Score. Those are explained in 4.1.

Finally, the highest F-Measure (F1-score) obtained is a score of 0.768 in the balanced distribution.

3. **Contextualized Sarcasm Detection on Twitter[12]**: This paper is made by David Bamman and Noah A. Smith (from School of Computer Science Carnegie Mellon University). As already indicated in the title, they use twitter as a source to pick up the comments to analyze. As in many other Twitter-based studies, they use the hastag #sarcatic or #sarcasm to search for sarcastic comments more easily. In this case, they also take into account the context of the comment, but go beyond simply studying the thread of the comment; they even look at the relationship of the people who exchange these sarcastic comments. They realize that sarcasm is used more when two people know each other. Otherwise, the use of hastag #sarcasm is more frequent.

Therefore, they present here a series of experiment, reasoning about features derived not only from the local context of the message itself (as in past work), but also using information about the author, their relationship to their audience and the immediate communicative context they both share.

In order to get these sarcastic comments and form the data set to be studied, they filter those comments that contain the untilgs already mentioned and that have at least three words each. Furthermore, to explore the influence of the communicative context, we further subsample this set to include only tweets that are responses to another tweet. The total dataset is evenly balanced at 19.534 tweets. They divide the model into 4 main features: those scoped only over the immediate tweet being predicted (Tweet Features); those that reason over the author of that tweet, including historical data by that author (Author Features); those that reason over the addressee of the tweet, including historical data for that individual and the author's history of interaction with them (Audience Features); and features that consider the interaction between the tweet being predicted and the tweet that it is responding to (Response Features).

Finally, combining the features, if all the characteristics are included together, the best performance is obtained with 85.1%. They realize that most of these gains come simply from adding author's information.

CHAPTER 3

Proposed Model

3.1 Introduction

In this chapter we will talk about the technical development of the project, showing and explaining all the techniques that have been used throughout the development of the model. In chapter 4, all the cases studied and the choice of the best of them will be explained. The final result will be detailed in chapter 5.

Firstly, it will be detail the "Preprocessing" task that has been implemented for cleaning the texts of the dataset. This first process is indispensable in this type of work as it prepares the text for the subsequent feature extraction and Machine Learning implementation.

Next, it will be explained how the features extraction have been carried out, which try to obtain the greatest amount of information from the texts in order to be able to apply the Machine Learning algorithm. Some examples of these features extraction are the assignment of grammatical categories to the words of the texts (POS) or obtaining similar subjects in some parts of the data.

Finally, the classifiers that have been tested to obtain the final score of the model will be detailed.

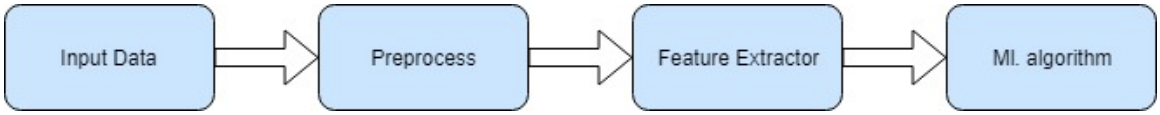

Figure 3.1: Model Development Phases

3.2 Preprocess

The Preprocessing phase, consists on cleaning the text for a greater understanding by the machine before the feature extraction. This cleaning is achieved by eliminating irrelevant words, punctuation marks or other characters that are not necessary for subsequent training. However, some steps can be eliminated. For example, sometimes, the model obtains a greater precision if the punctuation marks are not removed, as we will see. Before doing this steps, it is always necessary to "tokenize" the text. Given a character sequence, "Tokenization" is the task of chopping it up into pieces, called tokens. To do this we use the 'sent_tokenize' and 'word_tokenize' methods provided by the NLTK library. This paper [18] shows us a good work about the importance of a good preprocessing in order to obtain a final accuracy.

Once this Tokenization has been done, other preprocessing techniques can be used (all of them made thanks to the methods provided by the NLTK library). Here are some of the ones that have been tested for this project.

3.2.1 Lemmatization

This task makes it possible to unify terms that convey the same idea. It consists on replacing each one with its "lemma". By definition, a "lemma" represents all the words derived from it, but it does not necessarily have to be formed by the same root (for example, the lemma of the word "best" is "good" and the lemma of the verb "eat" is "eat"). Other examples: the lemma of the words "am", "are", "is" would be "be"; or the lemma of the words "car", "cars","car's","cars'" would be "car". WordNetLemmatizer is a method example in NLTK library for lemmatization.

3.2.2 Stemming

Another way to unify terms is to obtain the root (or stem) of the terms. To do this we eliminate the endings of each word, so that all are represented by a small part of them. For example, words "automates" and "automatic" would be transformed into "automat"; or words "love", "loving", "lovingly" and "loved" would be transformed into "love". There are several available stemmers in NLTK library, like PorterStemmer or lancaster.

3.2.3 Stop Word Removal

It consists on removing the stopwords in lower case. In this case NLTK provides us with a list of Stopwords in English that will remove those terms that will not add value for feature extraction. Some of these stopwords are the following: i, me, my, myself, what, that, these, etc.

3.2.4 Punctuation Removal

It consists on eliminating all punctuation marks that may not be necessary for subsequent training. By writing this line "string.punctuation" scikit-learn give us a string of all punctuation marks like: ! , " , #, $, % , (, =, ?, @, etc. Then we can search those characters in the texts from the data and remove all of them if it is necessary.

3.3 Feature Extraction

During this section, we will analyze different methods of feature extraction. This task is very important for the classification of the texts since the characteristics (or features) that we select are the ones that will be used later to train the classifier. The selection of features varies depending on the type of documents we are dealing with. Certain features may not provide valuable information to the classifier or it may even confuse it in some cases, but others may be very useful. Our goal is to find as many features with relevant information as possible. But we must be careful because, as variables are added to the model, the risk of our classifier making generalization errors increases.

As we shall see below, any feature extraction requires a subsequent vectorization, which, as its name indicates, it is the process to transform into a vector the information obtained (in this case, words). Each vectorization tool has its own representation that allows us to present the data in a processable format for the learning algorithm of the machine that we will see in the following sections.

Now we are going to comment on different feature extraction methods that we have used in our study.

3.3.1 Lexical Stats

With the "sent_tokenize" method provided by NLTK library, this feature counts the total number of sentences in the document. The final features extracted are combined in a tuple containing the total length of the document with the total number of sentences in that document.

3.3.2 TF-IDF

To know how relevant a word is to a document in a collection we use the vectorization TF-IDF (Term Frequency - Inverse Document Frequency), it is a feature extractor provided by Scikit-learn library. It is an algebraic model that represents textual information as vectors, being its components the importance of a term. The objective is to model documents in a vector space ignoring the order of the words but focusing on the information about the occurrences of the words. The tf-idf value increases proportionally with the number of times a word appears in the document, but it is compensated by the frequency of the word in the document collection (in the complete corpus), which allows handling the fact that some words appear more frequently than others. We cannot forget that some existing words do not provide any information, they are called stop words and they are mainly prepositions, pronouns and articles. As we have already seen in section 3.2 these words need to be preprocessed so that our learning algorithm does not consider them.

3.3.3 Ngram

A substring of 'n' consecutive elements in a given sequence is called n-gram. In our case, it refers to the grouping of 'n' words of a given text. It is one of the most relevant aspects in the feature extraction because it allows the machine to understand a context rather than individual words, increasing the probability of obtaining better classification results. The most commonly used cases of n-grams are n=1,2,3; unigrams, bigrams and trigrams, in that order. Therefore, we will be able to see the groupings of one, two or three words that are repeated throughout the corpus, studying their weight within the corpus. This is possible thanks to "CountVectorizer" function provided also by the Scikit-learn library [28], giving us a vector as a result. We can customize this function as follows:

- *stop_words*: We can remove the stop words if it has not been done previously in the preprocessing phase.
- *ngram_range*: This param indicates the range of the n-grams for a diagram, biggram, trigram, etc.
- *min_df*: ignores terms that have a document frequency (presence in % of documents) strictly lower than the given threshold. For example, Min_df=0.66 requires that a term appears in 66% of the documents for it to be considered part of the vocabulary.
- *max_df*: ignores terms that have a document frequency strictly upper than the given threshold.

- *max_features*: it allows us to build a vocabulary that only considers the top max_features ordered by term frequency across the corpus.

3.3.4 POS

Known as Part Of Speech (POS) [39]. It is the process of assigning a grammatical category to a word. It consists of classifying each term of a text according to its context and the function it performs: verb, adjective, adverb, determinant, etc.

Tag	Meaning	English Examples
ADJ	adjective	new, good, high, special, big, local
ADP	adposition	on, of, at, with, by, into, under
ADV	adverb	really, already, still, early, now
CONJ	conjunction	and, or, but, if, while, although
DET	determiner, article	the, a, some, most, every, no, which
NOUN	noun	year, home, costs, time, Africa
NUM	numeral	twenty-four, fourth, 1991, 14:24
PRT	particle	at, on, out, over per, that, up, with
PRON	pronoun	he, their, her, its, my, I, us
VERB	verb	is, say, told, given, playing, would
.	punctuation marks	. , ; !
X	other	ersatz, esprit, dunno, gr8, univeristy

Figure 3.2: POS Tagging examples table [15]

This technique is very useful in sentiment analysis (which affects our case study) because it allows us to differentiate words that are written the same but have different meanings. Thanks to the NLTK library, we can perform this task. After preprocessing the text, the assignment is made to each of the words in the text of the labels shown in table 3.2. After this labelling, the total elements of each category is counted and divided by the total elements of all categories. In this way, we get a weight for each category of POS Tagging. Finally, the result is a dictionaries list of the input dataset which, thanks to the DictVerorizer function

from Scikit-Learn library, allows us to transform these lists of feature value mappings into vectors, an actionable format for the Machine Learning algorithm.

3.3.5 LDA

Latent Dirichlet Allocation (LDA) is defined as a model that generates topics based on the frequency of occurrence of words in texts. This model starts with preprocessing the input text, creating a matrix containing the word count, and then identifying a number of topics according to the previous word specification per topic and total number of topics. The final result consists on a list of topics which in turn contains a list of words associated with the weight corresponding to the relationship with that topic. This is a great advantage in broadly identifying the issues covered by a document.

3.3.6 Word2vec

The emergence of deep learning and the availability of large databases have led to the emergence of new classification methods such as Word2Vec. As already mentioned in the previous chapter, it is one of the most popular technique to learn "word embeddings" (vector representations of a particular word) using shallow neural network. The words or phrases of the vocabulary are linked to vectors of fixed length to represent each word of the corpus. Traditional representation methods often ignore context information or word order in texts. However, this representation makes these limitations disappear, since the vectors generated by the neural network have a good semantic and syntactic sense. Semantic in the sense that the word "iOS" will be close to "Android". And syntactic since related words such as "king" and "queen" will be at a similar distance as "man" and "woman".

In other words, thanks to this task we can locate in a vectorial space the different groups of words that the tool has formed from the corpus, organizing these words both semantically and syntactically in a quite optimal way.

In order to be able to make this representation of word vectors of the texts, we have used the GSITK library with the Word2VecFeatures() method, already explained in Section 2.7. This vector representation technique is based on this paper [10].

3.3.7 SIMON

Thanks to the function provided by the GSITK library, we can use the SIMON feature extractor function to vectorially represent a word. It uses the "word2vec" function to vectorially represent the words of the sentiment lexicon. Then, thanks to the algorithm provided by SIMON, we look for similarities between the words of the lexicon and the words of the input data, creating a new vector for this input word. The technical aspects

have been explained in Section 2.7. In order to use this tool, as we saw in that Section 2.7 it is necessary to use a "Sentiment Lexicon". "Lexicon" is a vocabulary, a list of words, a dictionary. In other words, it is a collection of information about the words of a language about the lexical categories to which they belong.

3.4 Classifier

The classifier we have used to train our model is the Logistic Regression. As we explained in Section 2.2.3, this is a supervised learning algorithm. It is a statistical method used to determine the contribution of various attributes to a pair of results.

We can describe 3 types of logistic regression:

- *Binary Logistic Regression:* The categorical response has only 2 possible outcomes. Sarcastic or non-sarcastic.

- *Multinomial Logistic Regression:* Three or more categories without ordering.

- *Ordinal Logistic Regression:* Three or more categories with ordering.

Therefore, it is suitable for our case study where the output is "0" or "1" only, more specifically, Binary Logistic Regression [20].

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed to describe properties of population growth in ecology. It's a curve that can take any real number between '0' and '1', but never exactly at those limits. It follows this ecuation:

$$\frac{1}{1 + e^{-value}} \tag{3.1}$$

Where 'value' is the actual numerical value that you want to transform.

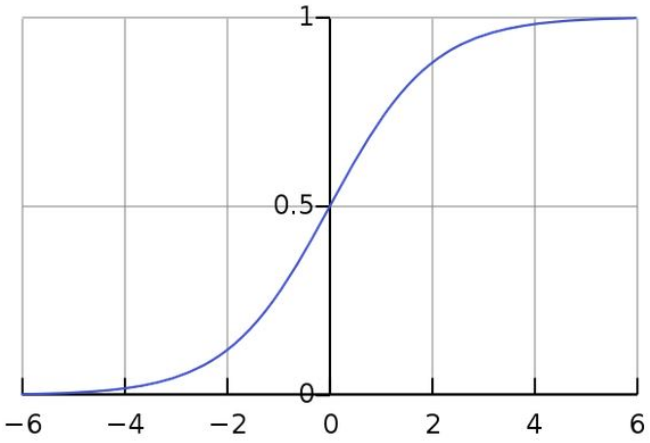

Figure 3.3: Logistic function

Now that we know what the logistic function is, let's see how it is used in logistic regression. Below there is an example logistic regression equation:

$$y = \frac{e^{b_0+(b_1\times x)}}{1+e^{b_0+(b_1\times x)}} \tag{3.2}$$

Where: x: Input values; b_0, b_1: coefficient values; y: output values.

Each column in your input data has an associated b coefficient. The coefficients of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation. It is enough to say that a minimization algorithm is used to optimize the best values for the coefficients for your training data.

These are the most required things to keep in mind when deciding if we want to use the Logistic Regression model:

- *Binary output variable:* The categorical response has only 2 possible outcomes. Sarcastic or non-sarcastic.
- *Remove Noise:* Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.
- *Fail to Converge:* It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse.

CHAPTER 4

Materials and Methods

4.1 Introduction

In this chapter we are going to describe how our model has been developed. It will show all the Feature Extraction combinations that have been tested as well as the different techniques used for preprocessing justifying the final choice of those we have chosen.

Furthermore, datasets that have been used to elaborate the project will be commented, as well as the way in which it has been decided to organise it for further analysis. Before we begin, we must mention some important concepts that have not been discussed and their understanding is required to proceed with the explanation of this chapter.

- ***Pipeline:*** A pipeline in NLP is a chain of independent modules, each one taking as an input the output of the module before it. To automate the workflow a Pipeline will be implemented. This will group all the functionalities that we want to add to the model: from the transformation of the input data in a suitable format to the application of the Machine Learning algorithms. The features to be obtained are joined using the FeatureUnion function, a method provided by the Scikit-Learn library. This class concatenates the results obtained from multiple transformers. It applies the list of transformers to the input data in parallel and then concatenates the results obtained. Each transformation has two following parameters: "BaseEstimator", which adjusts the estimator parameters; and "TransformerMixin", which allows us to adjust and

transform the data to obtain the required characteristics.

In the next section, we will see different pipelines that have been tested throughout the project, finally choosing the one that has had better metrics.

- ***Metrics:*** To evaluate this classification task, those are the metrics tools provided by the Scikit-Learn library that have been used:

  – Confusion Matrix: A confusion matrix is a summary of prediction results on a classification problem. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It consists of four parameters: True Positives(TP), True Negatives (TN), False Positives (FP) and False Negatives(FN). The representation will be as follows:

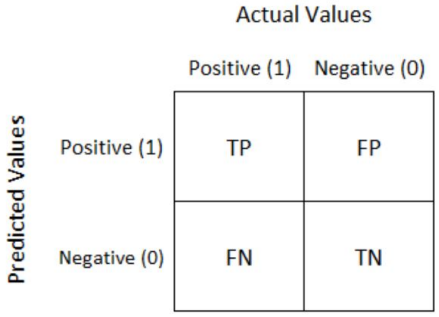

Figure 4.1: Confusion Matrix Table

  – Precision: Precision looks at the ratio of correct positive observations. It is calculated as the division of true positives into the sum of true positives and false positives.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

  – Recall: Recall is also known as sensitivity or true positive rate. It's the ratio of correctly predicted positive events. It is calculated as the division of true positives into the sum of true positives and false positives.

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

  – F1-Score: The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at '1' and worst at '0'.

$$F1Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4.3}$$

- ***Cross Validation:*** Although we have not used this technique in our complete model, we have used Cross Validation to search for the n-gram that best fits the Machine Learning algorithm we are going to use. It is not good practice to use this technique with such large datasets and with many feature extraction to test. Cross-Validation is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. It consists of the division of the input data into K subsets and the subsequent training in the K-1 subset elements. This is because the last subset is used for testing. This operation will be repeated K iterations. In other words, one round of cross-validation involves partitioning a sample of data into complementary subsets (K subsets) performing the analysis on one subset (the training set), and validating the analysis on the other subset (testing set) and then do this iteration the same number of times as the number of subsets, so every subset is tested on at least once.

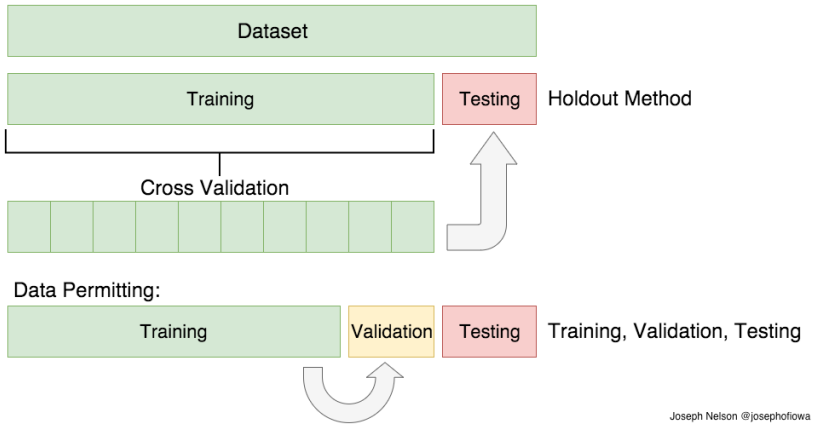

Figure 4.2: Scheme of the Cross Validation method [13]

- ***GridSearch:*** This method (provided by Scikit-Learn) makes an exhaustive search on the values of the parameters specified for an estimator. In our case, we have used this function to search the optimal n-gram for our model. Making variations with the parameters "ngram_range" and "ngram_maxFeature", which we already explained in Section 3.3.3 , we chose the "n-gram" that best suits the algorithm we are going to train with, LogisticRegresion. This function uses Cross Validation to search for the combination with the best result.

  In the next section, the final n-gram chosen will be specified, that is, the one that the GridSearch function has found as the best score.

4.2 Datasets

As already mentioned in section 2.8, two datasets have been used in this project, thus being able to choose the one that gives us the greatest accuracy to our model.

The first one that has been used is the one provided by the SARC corpus [7], which obtains the comments from the Reddit page. This corpus has 1,3M of sarcastics statements, allowing to learn in both balanced and unbalanced label regimes. All these comments have been tagged by the author himself.

For our study case, it has been decided to choose a balanced training, where there are the same number of sarcastic and non-sarcastic comments. Therefore, we were given a first .json format file with all the comments organized by id, also specifying other relevant information that could be useful for the development of the model; such as the name of the author who has written each comment, the subject of the comment, the creation date of the comment, etc. On the other hand, they also gave us another .csv format file, with a list of conversations from Reddit forum, specifying the id of the comment within the conversation. Each row of the file refers to a conversation thread, each of them separated in 3 groups. The first group contains the ids with the post of comments. The second group contains the ids with the answers to the post of the first group, where an id is referred to a sarcastic comment and the other id is referred to a non-sarcastic comment. The third group contains a label for each answer of the second group, "1" means that it is sarcastic and "0" that is not.

The model developed has not taken into account the study of the context of the comment. This is why only the comments that are tagged, those from the second group, have been extracted from these files; all tagged as sarcastic or non-sarcastic. Finally, a total of 13.668 comments (balanced) were obtained for the training phase of the model and 3.406 for the test phase.

	id_comment	comment	sarcastic
0	c205da7	TIL, voluntary slavery by starving people who'...	1
1	c201mb5	Welcome to china	0
2	c21btxl	That'd be great if only Ron Paul wasn't A CHRI...	1
3	c21bxjw	What's the Vegas Line on this one?	0
4	c21hz0p	But, but...lowering taxes creates jobs!	1

Figure 4.3: Piece of Dataframe example

For a better data organization, thanks to the Pandas library, we can use the Dataframes to prepare our input in tables as shown in Figure 4.3.

- *id_comment:* the id that refers to a comment.
- *comment:* the text that will be analyzed by the model.
- *sarcastic:* the label that says if a comment is sarcastic or not. "1" indicates that it is, and "0" indicates that it is not.

The other parameters provided by the corpus have not been introduced in the dataframe to be analysed as they will not be taken into account for the development of the model. These parameters have not been included because, as previously mentioned, our model has been developed without taking into account other parameters external to the comment itself: the context, the author who writes it, etc. This is done in order to simplify the model.

The evaluation has been expanded by means of an additional dataset with which we can measure the effectiveness of our model. This one gave us more tagged comments than the SARC corpus and did not provide context information, making it more suitable for the model developed. This is the dataset provided by the paper New-Headlines-Dataset-For-Sarcasm-Detection [27] which obtains the sarcastic comments from the headlines of "The Onion", and all the non-sarcastic comments from "Huffpost" headlines. This dataset has 21.368 comments for the training phase and 5.340 for the test phase. Something good about this dataset is that "The Onion" has as its only purpose to publish sarcastic news, so you get a high quality label, with much less noise than if you got comments from Twitter.

This dataset has been structured in the dataframe in the same way as the first SARC corpus, with the same labeling format: "1" sarcastic and "0" non-sarcastic.

Once the data have been organised in the right way, the texts are preprocessed, as we have seen in the previous section 3.2

4.3 Evaluation methodology

To explain the evaluation of our model, as mentioned above, two different datasets have been used. We will divide this section into two sub-sections explaining in each one of them a diferente dataset. In this section we will only show the different tests performed. The justification and discussion of the values obtained will be carried out in the following chapter 5. Our model has only been trained with one Machine Learning Algorithm: Logistic Regression. As we have already explained in Section 3.4, this clasiffier is the most suitable for our task.

Due to the outputs of this model will be only “0” or “1” (binary output), this has been the optimal algorithm for this type of work. In addition, tests have been made at the beginning with other algorithms to contrast this idea and effectively Logistic Regression was the most appropriate. Therefore, what we want to focus on here is the search for the most important features extraction for the development of the model.

4.3.1 SARC Corpus

In this first evaluation with this dataset, the following tests were performed. It should be noted that after several tests working with the preprocessing in the different Feature Extraction, we realized that applying the four preprocessing techniques we saw in section 3.2 (Stemming, Lemmatization, Stop Word Removal, Punctuation Removal) did not help to get better metrics. Therefore, the preprocessing that is done in each of them is simply the first step of ”Tokenization”, always essential for any NLP work, removing other 4 preprocessing techinques. On the other hand, as we also commented in the previous section, thanks to the GridSearch() method we were able to obtain the N-gram that gave us the best result for the classifier with which we were going to train. The parameters that best gave us results were: n_grama_range=(1,2) and max_features=50.000. The first parameter indicates that it takes subsequences of 1 and 2 words, i.e. it makes unigrams and bigrams at the same time.

Pipeline	Lexical Stats	TF-IDF	N-grams	POS	LDA	W2Vec
1	X	X	X	X	X	X
2		X	X	X	X	X
3		X	X		X	X
4	X	X	X		X	X
5		X			X	X
6		X	X			X
7		X	X			X
8		X				X
9		X				

Table 4.1: Pipelines made with SARC corpus

As we can see in the table above, 9 training and testing tests were carried out with 9 different pipelines. Each row represents a pipeline with its respective Feature Extraction marked with an "X" if it contains it. Many variations were made, even tested in one pipeline with all feature extraction and another only with one of them. However, not only it was tested a pipeline with one feature extraction, as we see in pipeline 9 (only TF-IDF), it was also tested with the rest of features individually. This information is not shown here because it was not considered relevant. In this case, no test was performed with SIMON feature extraction. As we will see below, this came up later to improve the results with the second dataset.

As we will comment in Section 6.2, we were adding feature extractor to the model for this first dataset, but no improvement were made, always with the same results. Adding also this SIMON feature extractor wouldn't give any extra information.

In the next chapter we will discuss about the results obtained with this dataset. But, due to the fact no good metrics were obtained, we train our model with another dataset, as we are going to explain below.

4.3.2 New Headlines Dataset

It was decided to search for another dataset due to dissatisfaction with the final result of the previous dataset. In this case, this dataset contains many more tagged texts than the previous dataset, which is suitable for our model and a better result is expected. In addition, for this new dataset, a new technique was also tested as a new feature extraction: SIMON. As already mentioned, this tool was provided by the GSITK library. We will therefore add one more column in the first table showing all tested pipelines. As in the previous case, in the preprocessing, it was only done with "Tokenization". On the other hand, in order to choose the optimal N-gram, tests were performed with unigrams and bigrams together (as in the previous case) and with bigrams only, giving better result the latter, as we will see below. From pipeline 5, it is made with bigrams only. Because a lot of pipelines were made, only the top 10 will be represented.

Pipeline	Lexical Stats	TF-IDF	N-grams	POS	LDA	W2Vec	SIMON
1	X		X	X	X		
2	X		X				
3			X				
4			X	X	X		
5	X		X	X	X		
6	X	X	X	X	X		
7	X	X	X	X	X	X	
8	X	X	X	X	X		X
9	X		X		X		X
10	X	X	X	X	X	X	X

Table 4.2: Pipelines made with New Headlines Dataset

CHAPTER 5

Evaluation

5.1 Introduction

In this section we will describe the results of each pipelines in Tables 5.1 and 5.2, in order to determine the best pipeline for our model. We will select the best one in terms of F1-Score and then we will take into account its confusion matrices. We will also analyze the choice of some parameters in the pipelines and we will discuss about the conclusions reached.

5.2 Results

This first Table 5.1 shows the results obtained for each pipeline with the SARC corpus, with the Recall, Precision and F1-Score metrics, which as we have seen in the previous Section 4.1, the F1-Score is the average between Recall and Precision. All the results shown in the tables are weighted over a total of '1'.

Then, we describe the metrics obtained from this second trained dataset (Table 5.2), showing the results of 10 pipelines developed. We will discuss in the next chapter the results shown here.

These metrics, precision, recall and f1-score, are the average of the results obtained for the classes of "0" and "1" of the Logistic Regression Classifier.

In reference to the first dataset trained, there are not any difference between the results

Pipeline	Precision	Recall	F1-Score
1	0,69	0,71	0,70
2	0,69	0,71	0,70
3	0,70	0,72	0,71
4	0,69	0,71	0,70
5	0,70	0,71	0,70
6	0,70	0,72	0,71
7	0,69	0,72	0,71
8	0,69	0,72	0,71
9	0,69	0,73	0,71

Table 5.1: First model metrics

obtained in the models. Finally, 0.71% was obtained as the best F1-Score score, as shown in Table 5.1 . Therefore, due to this low score it was decided to train another dataset, as explained in the previous chapter. The baseline provided by the SARC corpus was 0.75%, but this one took into account the context of each comment. The model developed here did not take this factor into account, that is why the final result was not as expected.

We then proceeded to train with the new dataset, which had a larger number of tagged comments (26.708 comments) and better tagged than the first set. As you can see in Table 5.2, with the second dataset our model got better results, thus obtain 0.87% of F1-Score as the best scores in pipelines 7, 8, 10.

For these cases, the features analyzed in each of them were the following:

- **Pipeline 7:** Lexical Stats, TF-IDF, N-grams (bigrams), POS, LDA and W2Vec.
- **Pipeline 8:** Lexical Stats, TF-IDF, N-grams (bigrams), POS, LDA and SIMON.
- **Pipeline 10:** Lexical Stats, TF-IDF, N-grams (bigrams), POS, LDA, W2Vec and SIMON.

The same result was obtained by training the model with all features and W2vec and SIMON individually (pipelines 7, 8), as W2Vec and SIMON in the same pipeline (pipeline

Pipeline	Precision	Recall	F1-Score
1	0,83	0,82	0,82
2	0,80	0,79	0,79
3	0,79	0,78	0,79
4	0,83	0,81	0,82
5	0,85	0,86	0,86
6	0,86	0,85	0,86
7	0,87	0,86	0,87
8	0,87	0,86	0,87
9	0,85	0,86	0,85
10	0,87	0,86	0,87

Table 5.2: Second model metrics

10). Therefore, since the latter would have more processing time, the models chosen were the first two: pipeline 7 and pipeline 8.

In order to discuss these two pipelines further, we show below the confusion matrices from the second dataset, in the same format as indicate in Section 4.1:

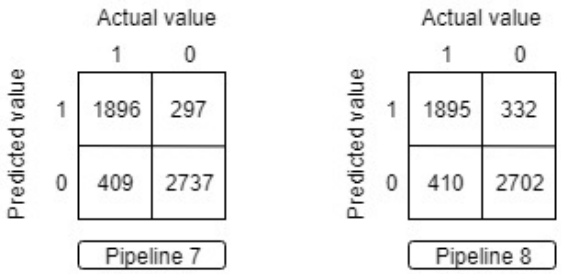

Figure 5.1: Second Dataset Confusion Matrix for Pipeline 7 (left) and Pipeline 8 (right).

Looking at Figure 5.1: Top left are the True Positives (TP), successful sarcastic comments; top right are the False Positives (FP), failed sarcastic comments; bottom left are the False Negative (FN), failed non-sarcastic comments; bottom right are the True Negative (TN), successful non-sarcastic comments.

5.3 Discussion

First of all, we must comment on the chosen preprocessing. In the first dataset our model was not prepared to study the context of the comments. Tagged comments that we were able to obtain as inputs for the model were relatively few. For this reason, removing punctuation marks and other information in the preprocessing phase would eliminate information for training. It should also be noted that users often use a lot of punctuation marks when they want to write a sarcasm comments. The question marks (for irony), or exclamation marks (to emphasize a comment) are quite frequent patterns when someone wants to write sarcasm. That is why removing these punctuation marks could eliminate information in the model. The same for the second dataset. It was necessary to keep these punctuation marks for the training phase.

In the traning model with the first dataset, as we metioned before, there were not any difference between the final results of each pipeline, despite have trained with different combinations of features extraction. One of the reason of this problem is because we have not analyzed the context of the comments, and this dataset was prepared to analyzed that feature. Our model was not take into account this fact. Another possible problem may be that Overfitting was produced.

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. What will happen next is that our model will only adjust to learn the particular cases we teach it and will be unable to recognize new input data. Another case of overfitting can be a excessive quantity of features, with many different variants, without sufficient samples.

As mentioned above, the second dataset contains comments collected from "The Onion", an organization that publishes sarcastic news. Therefore, the noise that this dataset can apply to the training phase of our model, will be less than the first one, because we know that sarcastic comments are well labeled. This is reflected in a good way in the metrics for this dataset.

On the other hand, it is note that the use of W2Vec and SIMON together in the same pipeline does not make any difference in terms of result. However, if we put them separate in pipelines individually, the result is better. A possible explanation for this is that both provide the same information. As already explained in Section 2.7, both vectorize the words, placing these words close together in a vector space, closer the same words if they are syntactically and semantically common.

With regard to the confusion matrices described above (Figure 5.1), first of all, high-

light the “TP” and the “TN”, which are the correct sarcastic and non sarcastic comments respectively. In both cases, the “TPs” have essentially the same value. However, pipeline 7 is more accurate than pipeline 8 in terms of non-sarcastic comments (as shown in the “TN” value). Therefore, in terms of “PF” values, which are the sarcastic comments that the system has failed, Pipeline 8 has a higher value than Pipeline 7.

For these two reasons, Pipeline 7, by having more accurate non-sarcastic comments and by having fewer failed sarcastic comments, becomes the chosen pipeline.

In conclusion, having tested two datasets in the model developed with the different pipelines and according to the values shown by Confusion Matrices, for the reasons we have recently discussed, the model chosen is pipeline 7: Lexical Stats, TF-IDF, N-grams (bigrams), POS, LDA and W2Vec.

CHAPTER 6

Conclusions and future work

6.1 Introduction

In this chapter we are going to summarize the process that has been carried out for the development of the project. We will describe the conclusions extracted and we will also propose some suggestions for future work on this model.

6.2 Conclusions

The main goal of the project was the development of a model capable of detecting sarcastic comments in texts, applying Natural Language Processing and Machine Learning techniques.

In order to do this, two datasets have been used: the first obtained comments from the Reddit website, also providing data for the analysis of the context of a comment; the second obtained comments from the "The Onion" (for sarcastic comments) and "The Huffpost" (for non-sarcastic comments) pages. Due to how our model was made, better results were obtained from the second dataset. In order to have the input data in a suitable format to analyze the features, preprocessing the text was necessary (cleaning the text). Subsequently, different futures were obtained thanks to different tools provided by Sickit-Learn, NLTK or GSITK among others. Finally, once these features were vectorized, the Logistic

Regression algorithm was applied to train the model. Several pipelines were made with different combinations of the developed features extractor, thus being able to describe which was the best implementation for our final model. As for the results obtained, no good results were obtained when training the model with the SARC corpus (71% F1-Score). Not reaching the minimum that the corpus required (75% as baseline), the training was carried out with a second dataset, "New Headlines dataset", which had less noise and contained more comments than the first one. Good results were obtained for this second set (87%), concluding with two pipelines for the model.

Consequently, the second dataset was more suitable to be trained than the first one. The first dataset had fewer comments than the second one, it has information that our model was not prepare to analyze and more noise in labeling. It gave us no improvement in metrics, as many things were tried with no solution. However, with this first dataset we tried many things and we could use it in the training phase with the second dataset.

For this reason, SIMON feature extractor was tested in the second dataset, but not tested in the first one. Even adding more feature extractor to look for improvements in metrics with this first dataset, there were no changes in this results. Therefore, SIMON feature extractor was not tested for this first set, as adding more features to the classifier did not improve anything.

In conclusion to the purpose of this project, we can say that the objectives have been achieved, obtaining the most relevant features when analyzing a system to detect sarcasms in texts. The final result has been a model with a F1-Score metric of 87%.

6.3 Future work

In this section we will explain the possible new features or other improvements that could be done to the project.

- **Taking into account the context of a comment:** as we have mentioned many times before, taking into account the track of a conversation, we can see the context of a comment and improve our metrics, by analyzing if the discussion topics are similar.

- **Other features extractor such us:**

  - Analyze the amount of sarcastic comments that the author has previously written. To make it possible it is necessary to collect this information from any forum page that gives this information. For example, in Twitter, if we analize the comments of a specific user we can get how many comments he has in his profile with the hastag #sarcastic.

  - We can also get the age of the user and analyze if this is a relevant feature.

- **New Languages:** in this project we have only used English datasets. It would be interesting to do this type of work with a Spanish dataset.

APPENDIX A

Appendix A: Impact of the project

A.1 Introduction

The use of sarcasms to express opinions on the internet is very common in social networks. It is a way of expressing a very negative opinion without the use of bad words.

The application of this sarcasm detection model can be very useful to those companies that require this knowledge to analyze the criticisms of their customers.

In this appendix we will try to analyse the impacts related to the realization of this project from a social, economic, environmental, ethical and professional point of view.

A.2 Social Impact

As has already been mentioned, when it comes to analyse the opinion of users regarding a specific product or service, the use of sarcasm is very common. For this reason, the use of a system such as this, capable of detecting when these sarcastic situations happen, is very useful for any field in which knowledge of a user's opinion with accuracy is require. A group interested in a system like this are all companies that have a direct relationship with the customer, either to know the criticism of a final product or to know the satisfaction of a service offered.

A.3 Economic Impact

In this section we will evaluate the possible economic impacts that companies may experience when using our system. Since it is mainly aimed at big companies, capable of analyzing large amounts of data (in this case comments) the use of this system can increase or maintain sales of a product in the long term. They will be able to know with a greater certainty the opinion of their clients, being able to act in this way to change something in future sales if it is necessary. In addition, you will be able to reduce the analysis time of these opinions since it verifies the feeling obtained from a comment, without the need of a physical person who does it.

A.4 Environmental Impact

This section aims to define the main environmental impacts of the development of a system such as ours. One of the biggest environmental problems that society must deal with today is the generation of waste from the point of view of technologies. For the implementation of this system in computer does not require any special hardware equipment.

However, developing software requires more power in the computers in which it is to be implemented, which could place a huge charge on our electricity networks and contributing to the gases emissions. To this consumption must be added the energy needed for the cooling system associated with this equipment. If the processing power of a computer increases, the energy to be applied to the cooling system is higher, which means a greater electricity expense.

A.5 Ethical and Professional Implications

Any system that contains artificial intelligence involves the automation of something, or easy access to something, which usually means jobs reductions. In our case, the fact that it automatically detects if a text is sarcastic means that it facilitates a task. Therefore, if this task of verifying a feeling in a comment was previously carried out by a physical person, the implementation of this system would eliminate this job.

APPENDIX B

Appendix B: Cost of the system

B.1 Introduction

The most relevant budgets for the project will be explained in this section. We will divide these budgets into 3 parts: Physical Resources, Human Resources and Licenses.

B.2 Physical Resources

The physical resource necessary for the development of this project has been mainly a computer. To quantify the cost of this device, we will take as a reference the one that has been used in the development of this work. Its main characteristics are:

- *Operating System:* Windows 10 Pro, 64 bits.
- *CPU:* Intel Core i7, 2,5 Ghz x 4.
- *RAM:* 16 GB
- *Disk:* 500 GB

It is estimated that the approximate cost of a computer with these characteristics is 1.000€.

B.3 Human Resources

Firstly, a estimation is made for a person who has no previous experience in the sector, or who does not have any qualifications yet. In this case, a job is estimated at 8€ per hour. Approximately, the time dedicated to this work has been 300 hours, counting the times of previous research, development and writing the final report. Therefore, for this first situation, the cost would be 2.400€.

On the other hand, we can consider a second case where you already have a degree, in this case Telecommunications Engineering. For a person with knowledge of NLP and ML, the average salary is 24.000€ per year.

B.4 Licenses

For the development of our project, all the software used is open-source. "Jupiter Notebook" amount other, is free for any user. So the cost in this case is 0€

Bibliography

[1] Natural language toolkit. https://www.nltk.org/.

[2] Numpy. http://www.numpy.org.

[3] Pandas. python data analysis library. https://pandas.pydata.org.

[4] Scientific computing tools for python. https://www.scipy.org/about.html.

[5] University of cambridge. https://dictionary.cambridge.org/es/diccionario/ingles/irony.

[6] University of cambridge. https://dictionary.cambridge.org/es/diccionario/ingles/sarcasm.

[7] A large self-annotated corpus for sarcasm. 2017.

[8] Ethem Alpaydin. Introduction to machine learning. MIT press, 2009.

[9] Oscar Araque. Design and Implementation of an Event Rules Web Editor. Trabajo fin de grado, Universidad Politécnica de Madrid, ETSI Telecomunicación, July 2014.

[10] Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. Expert Systems with Applications, 77:236 – 246, 2017.

[11] Oscar Araque, Ganggao Zhu, and Carlos A. Iglesias. A semantic similarity-based perspective of affect lexicons for sentiment analysis. Knowledge-Based Systems, 2018.

[12] David Bamman and Noah A. Smith. Contextualized sarcasm detection on twitter. In ICWSM, 2015.

[13] Adi Bronshtein. Train/test split and cross validation in python. https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca 2017.

[14] E. Cambria and B. White. Jumping nlp curves: A review of natural language processing research [review article]. IEEE Computational Intelligence Magazine, 9(2):48–57, May 2014.

[15] J. Fernando Sánchez Carlos A. Iglesias. Course notes for learning intelligent systems. https://github.com/gsi-upm/sitc/blob/master/ml1/2_5_0_Machine_Learning.ipynb, 2017.

[16] Florian Kunneman Christine Liebrecht and Antal van den Bosch. The perfect solution for detecting sarcasm in tweets not. https://www.aclweb.org/anthology/W/W13/W13-1605.pdf, 2013.

[17] S. Sánchez D. Pascual, F. Pla. Algoritmos de agrupamiento. http://marmota.dlsi.uji.es/WebBIB/papers/2007/1_Pascual-MIA-2007.pdf, 2017.

[18] Ricardo Barandela Eduardo Gasca. Influencia del preprocesamiento de la muestra de entrenamiento en el poder de generalización del perceptron multicapa. Laboratorio de Reconocimiento de Patrones Instituto Tecnológico de Toluca, México, 1960.

[19] Javier Luna Gonzalez. Tipos de aprendizaje automático. https://www.zemsania.com/recursos-zemsania/whitepapers/DTS/Machine_learning.pdf, 2018.

[20] Frank E. Harrell. Binary Logistic Regression, pages 219–274. Springer International Publishing, Cham, 2015.

[21] Nicolás Hellers. Machine learning e inteligencia artificial: impacto en la educación, el trabajo y la formación profesional, 2017.

[22] Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A large self-annotated corpus for sarcasm. CoRR, abs/1704.05579, 2017.

[23] Eric Postma Laurens van der Maaten. Dimensionality reduction: A comparative review. http://www.math.chalmers.se/Stat/Grundutb/GU/MSA220/S18/DimRed2.pdf, 2009.

[24] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2177–2185. Curran Associates, Inc., 2014.

[25] Afshin Rostamizadeh Mehryar Mohri and Ameet Talwalkar. Foundations of Machine Learning. MIT Press book, 2012.

[26] Microsoft. Algoritmo de regresión logística de microsoft. https://docs.microsoft.com/es-es/sql/analysis-services/data-mining/microsoft-logistic-regression-algorithm?view=sql-server-2017, 2017.

[27] Rishabh Misra. News headlines dataset for sarcasm detection. https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection/home, 2018.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[29] Raona. Los 10 algoritmos esenciales en machine learning. https://www.raona.com/los-10-algoritmos-esenciales-machine-learning/, 2017.

[30] Sebastian Raschka. Python Machine Learning. Packt Publishin Ltd, 2015.

[31] Antonio Reyes, Paolo Rosso, and Tony Veale. A multidimensional approach for detecting irony in twitter. Lang. Resour. Eval., 47(1):239–268, March 2013.

[32] Xin Rong. word2vec parameter learning explained. CoRR, abs/1411.2738, 2014.

[33] J.G.Carbonell y T.M.Mitchell R.S.Michalski. Machine Learning: An Artificial Intelligence Approach. Springer-Verlag, 1984.

[34] S. Sagiroglu and D. Sinanc. Big data: A review. In 2013 International Conference on Collaboration Technologies and Systems (CTS), pages 42–47, May 2013.

[35] J. Fernando Sánchez-Rada. Design and Implementation of an Agent Architecture Based on Web Hooks. Master's thesis, ETSIT-UPM, 2012.

[36] Ewan Klein Steven Bird and Edward Loper. Natural language toolkit (nltk). book. `https://www.nltk.org/book/ch00.html`, 2014.

[37] Byron C. Wallace, Do Kook Choe, and Eugene Charniak. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference, volume 1, pages 1035–1044. Association for Computational Linguistics (ACL), 2015.

[38] Steven Huffman y Alexis Ohanian. Reddit web page. `https://www.reddit.com/`, 2015.

[39] Álvaro García Gutiérrez. Machine learning en bases de datos de lenguaje natural, June 2016.