UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF A GENDER CLASSIFICATION SYSTEM BASED ON MACHINE LEARNING TECHNIQUES

Carlos Fuentes Sánchez

Enero de 2017

TRABAJO FIN DE GRADO

Título:	Desarrollo de un Sistema Clasificador de Género para Twitter basado en Técnicas de Aprendizaje Automático
Título (inglés):	Development of a Gender Classification System for Twitter based on Machine learning techniques
Autor:	Carlos Fuentes Sánchez.
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Juan Quemada Vives
Vocal:	Joaquín Salvachúa Rodriguez
Secretario:	Gabriel Huecas Fernández-Toribio
Suplente:	Santiago Pavón Gómez

FECHA DE LECTURA:

CALIFICACIÓN:

Resumen

En los últimos años Internet se ha introducido progresivamente en la vida de las personas. Contiene unas cantidades inmensas de información, y el acceso a ella se ha convertido en algo cotidiano. Gran parte de toda esa información proviene de las redes sociales, tales como Twitter o Facebook. Con el crecimiento exponencial en el número de usuarios de tales redes, la demanda de la caracterización de los atributos de esos usuarios también aumenta. Atributos como el género, edad, posicionamiento político,etc pueden ser de utilidad tanto para los usuarios como para las grandes empresas y sus análisis sociológicos.

Esta memoria es el resultado de un proyecto cuyo objetivo ha sido desplegar y desarrollar uns sistema clasificador que predice el género del usuario de una cuenta de Twitter. Para llevarlo a cabo, se ha desarrollado un plug-in en la plataforma Senpy que permite ejecutar nuestro sistema como un servicio. El lenguaje de programación usado para implementarlo ha sido Python

Para la implementación, han sido necesarias técnicas de Lenguaje Automático Supervisado (Supervised Machine Learning), así como herramientas de Procesado de Lenguaje Natural (NLP). Esto es debido a la necesidad del sistema de obtener y almacenar datos de Internet para su posterior procesado.

Para la prueba y evaluación de los distintos módulos de este sistema, se ha utilizado un conjunto de datos compuesto por Tweets de diferentes usuarios en dos idiomas diferentes: español e inglés.

Palabras clave: Twitter, Género, Senpy, Machine Learning, Análisis, Python, Scikit-Learn, Análisis, Clasificador, NLP.

Abstract

In the last years, Internet has been introducing progressively in people's lives. It contains huge amounts of information, and accessing it, has become something that people do in the day-to day. A big part of all that information comes from social networks, such as Twitter or Facebook. With the exponential growth in the number of users in those networks, the demand of the characterization of the attributes of those users also grows. Attributes such as gender, age, political beliefs, etc can be very useful for the users and for the big companies and their social analysis.

This thesis is the result of a project whose objective has been to deploy and develop a classifying system that predicts the gender of the user of a Twitter account. To do so, a plug-in has been developed in the platform Senpy, that allows the execution of our system as a service. The programming language that was used for the implementation has been Python.

For the development, Supervised Machine Learning techniques had been necessary, as well as Natural Language Processing (NLP) tools. This is due to the necessity of the system of obtaining and storing data from the Internet for its posterior process.

In order to test and evaluate the different modules of this system, a set of data composed by Tweets from different users had been used. They are in two different languages: Spanish and English

Keywords: Twitter, Gender, Senpy, Machine Learning, Anlaysis, Python, Scikit-Learn, Classifier, NLP.

Agradecimientos

Lo primero es dar las gracias a mis padres, que ellos han sido los que más me han apoyado para poder terminar este grado. En el día a día siempre han tenido palabras de ánimo tanto en la buenas, como en las malas situaciones. Y gracias por haberme ayudado a dar el salto a mi vida que habrá supuesto acabar esta carrera.

Por supuesto quiero dar las gracias a mi hermano David. De él obtuve la motivación de comenzar con la carrera de Teleco. Quiero agradecerle el entusiasmo que ha puesto siempre que le pedía cualquier tipo de ayuda, y el apoyo que me ha aportado en estos años.

Además quiero dar las gracias a todos mis amigos y compañeros de esta carrera, que me han acompañado todos estos años. Gracias a ellos las eternas tardes de estudio y las interminables entregas, resultaron momentos que merecerá la pena recordar.

Quiero dar las gracias a Carlos Ángel, mi tutor, por la ayuda prestada en el desarrollo de este proyecto.

Y por último a todos mis compañeros del GSI, que sus consejos y ayudas han supuesto un apoyo fundamental para que este proyecto haya podido terminarse.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

In the last few years, Internet has been a great revolution. It is reachable for everyone. Its benefits are enormous and their advances for the communications have been vital for a better access to different sources of information. Great part if this information received, comes from social networks such as Twitter.

This information provided by millions of interactions between the users, and thousands of text messages in their posts can give hints of personal information. That information may help build a characterization of those users, identifying their gender, age, personality or political and religious beliefs between other characteristics.

With the exponential increase in the number of users, demand for analytics on this data is also growing. Twitter does not store gender, age, or other characteristics of the users. These attributes of a user could be useful both for user experience as well as for consumption by brands in their social analytics.

In this project we are focusing on studying more in depth one single feature, the gender of the users. In order to do so, we take advantage of the social conventions that dictate

that on average, in the social networks, there are differences of behaviour between men and women. Some of these differences may be the style of writing, or the kind of people that they follow.

Characteristics such as the gender of the writer can not be directly discovered by an objective interpretation of the text. Detecting the gender of a user, could not be done in the majority of the cases just reading the tweets. It is based on detecting hidden information, that might not be obvious for the human direct perception.

1.2 Project goals

The amount of information proportioned by a Twitter account is huge, but not all the information is useful for our purpose. This project will show how important is organising and treating that data when we are trying to get rid of the useless information, and keep, in an usable way, all the information that can help our purposes.

More particularly, the principal goal is to study the possibility of performing an accurate enough system based on Machine Learning Techniques for data analysis. This project will use classification techniques in order to predict the gender of the user of a Twitter account. We will not only try to achieve the best possible accuracy results. We will also have in consideration if these results are worth it with the amount of resources needed in the implementation.

The other main goal would be wrapping up the system in a plug-in in the platform Senpy [1] so it can be executed independently and in a service form.

1.3 Tasks

The different tasks that this document performs are mainly three:

- Implementing a classification system for gender recognition. In order to do so we must perform the following steps:
  - Obtaining the corpus and division in training and test sets.
  - Organization and prepocessing of the information.
  - Definition of the features that are going to be used and extraction on them.
  - Election of classification algorithms and first experiments with the training set.

- Using different datasets, languages and feature extraction techniques and comparing them in order to draw conclusions about the performance and efficiency of them.
- Wrapping up all the work by building a plug-in of the classifier in the platform Senpy [1], this platform will be explained with more detail in the following chapters. The main goal of this task is to be able to use the system for real applications, not only for testing. This way any time that the plug-in is launched you can make a prediction of the gender of a user.

1.4 Structure of this document

In this section, we are explaining briefly all the chapters included in this project. After this introduction chapter (chapter 2), we will explain the enabling technologies that made this project possible. We will explain with more emphasis Scikit-Learn library and the platform Senpy, and we will also mention others such as the Twitter API, or the Textblob library.

The next chapter corresponds to the architecture of the project(Chapter 3), in it we will explain with detail the different modules that will form the system built. Right next to the architecture, we will pass to the next chapter. On that chapter we will explain that some tests were performed to check the characteristics and accuracy of the system built. The results of those tests were evaluated in this particular chapter (Chapter 4).

Finally we will conclude the document with a some brief conclusion (Chapter 5). On that conclusion, we will also include the problems faced during the realization of the project, and also some future work that can be performed later related with this theme.

CHAPTER 2

Enabling Technologies

2.1 Introduction

The most important technology used is Machine Learning [13]. It is a sub-field of computer science, and it comes from the most exhaustive techniques of pattern recognition and computational learning theory in artificial intelligence. Depending on the nature of the learning "signal". Machine Learning techniques can be classified into supervised learning, unsupervised learning and reinforcement learning. This project applies supervised learning techniques focusing on classification tasks.

Supervised learning algorithms are based on labelled training data. Using those labels, they can produce a function that is used for mapping new examples. Labelled training data consist of a usually large number of examples, each one with an input object that is usually a vector and its output value.

Those input objects are divided into classes, so each of the inputs must have a class assigned by the learning model. Classification and supervised learning techniques, usually come together, because classification problems contain input objects and their output values are their classes.

Resuming, in order to archive that the classifier works properly with the new examples, it needs a number of labelled examples. This way it can process all the labels, look for patterns and create classes. The following sections in this chapter describe technologies used to make this possible.

2.2 Scikit-learn

Scikit-learn is an open source machine learning library for Python, built on NumPy, SciPy and matplotlib [11] . It used for several tasks of classification, because of its great amount of classification algorithms. Those algorithms also have a great variety of categories such as Support Vector Machines, Nearest Neighbours or Decision Trees.

This library also allows the combination of multiple classifiers with Ensemble methods. An example of Ensemble methods is a Voting Classifier, whose inputs are the outputs of the desired classifiers and learns from it.

Classification in Scikit-learn follows the following architecture:

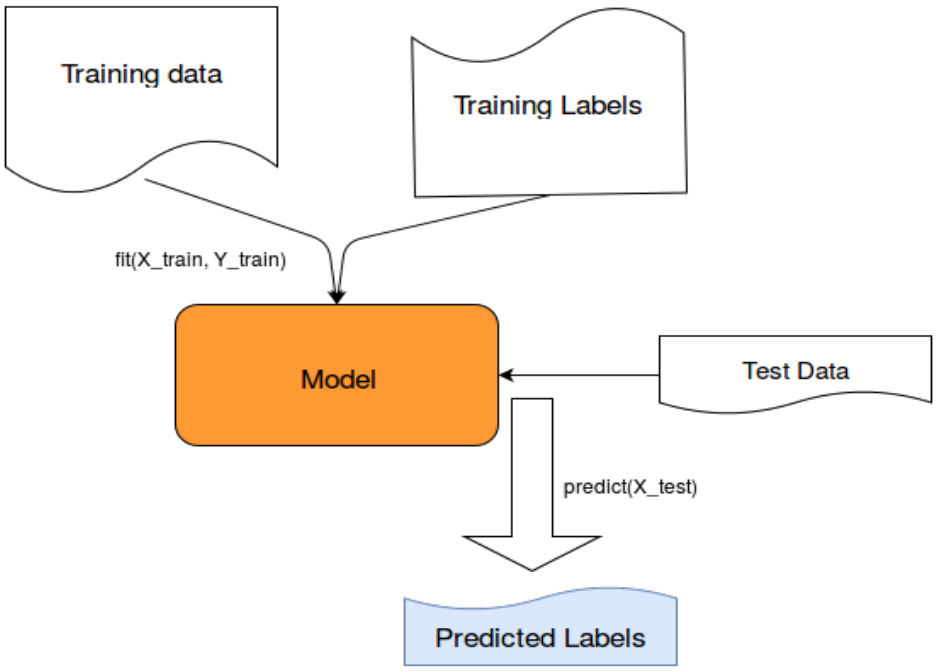

Figure 2.1: Model training and output prediction

Both training data and test data are in the form of features that are extracted from the raw data. This way it can be understood by the classifier. Scikit-Learn provides some tools for feature extraction. In this project are used the following two techniques of feature

extraction:

- ***Using dictionaries:*** they convert feature arrays represented as lists of standard Python dictionary objects to NumPy or SciPy representation.
- ***Text feature extraction:*** Scikit-learn provides utilities for the most common ways to extract numerical features from text.

When we try to learn more about feature extraction, we discover Term frequency-inverse document frequency (TF-IDF). It is a method that evaluates how important a word is in a text. It accomplishes that with the conversion of the textual representation into a Vector Space Model. This model is an algebraic model that represents textual information as vectors, being their components for example the importance of a term. The aim is modelling documents into a vector space ignoring ordering of words but focusing on information about occurrences of words.

The first step is creating a model of the texts into a vector space. It can be accomplished by creating a Dictionary of words that appear on those texts. But since not all the words contain information, such as a, the, at, on..., there is a need of pre-processing those texts. This way those words, that introduce noise can be avoided later in the classificatory. Those words are called stop words.

But we also need to pre-process those texts in a different way. It consists on the extraction of the root of the words. This is useful, because the concept of the word in general terms, gives more information than all the variations of the form. This way we also can avoid some noise.

The next thing to do is measuring the frequency that each word appears into that vector space. This is called "Term Frequency". For each word that appears in those documents, every document will have a matrix of their appearances on it. This way, each document is represented by a long vector with zeros on the missing words, and with the number of appearances on the words that did appear.

Once we obtained those long vectors, it is vital normalizing them. This is because the importance of a word appearing a number of times depends directly of how long the text is.

The next step is calculating the inverse document frequency. It can be defined as:

$\log \frac{|D|}{1+|d:t \in d|}$

Being the denominator the number of documents where t appears, and D the total

number of documents in the dataset. This function could variate, as it tries to get the impact of a word in the corpus and in this case it smooths it by computing it into a logarithmic scale.

Once we have calculated both term frequency and inverse document, the TF-IDF for each word on a document is calculated by multiplying both frequencies. A high TF-IDF is reached with a word with a high frequency in the document and with a low frequency in terms of the whole set.

2.3 Senpy and Linked data

Senpy [1] is an open source implementation of a linked data model that is usually used for sentiment and emotion analysis services based on semantic vocabularies Marl, Onyx and NIF. Senpy can use several formats such as turtle, JSON-LD and XML-RDF. This project is not about sentiment or emotion analysis bot, this system can be used as well for our purpose.

JSON-LD [3] is a method of encoding Linked Data using JSON. The data is serialized in a way that is similar to traditional JSON format. JSON-LD is designed around the concept of a context to provide additional mappings from JSON to an RDF model. The context links object properties in a JSON document to concepts in an ontology, as Marl or NIF. In our project are not going to use any of the ontologies previously described. This is due to the nature of our predictor.

RDF [7] is a family of World Wide Web Consortium (W3C) specifications originally designed as a meta data model for information interchange on the Web. RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link.

Before proceeding to explain the use of Senpy, it will be necessary to introduce briefly the ontology that is going to be used. IT is the ontology FOAF [4]. FOAF is an ontology based on linking people and information using the Web. Regardless of whether information is in people's heads, in physical or digital documents, or in the form of factual data, it can be linked. FOAF integrates three kinds of network: social networks of human collaboration, friendship and association.

Senpy proposes a modular and dynamic architecture that allows:

- Implementing different algorithms in a extensible way, yet offering a common interface

- Offering common services that facilitate development.

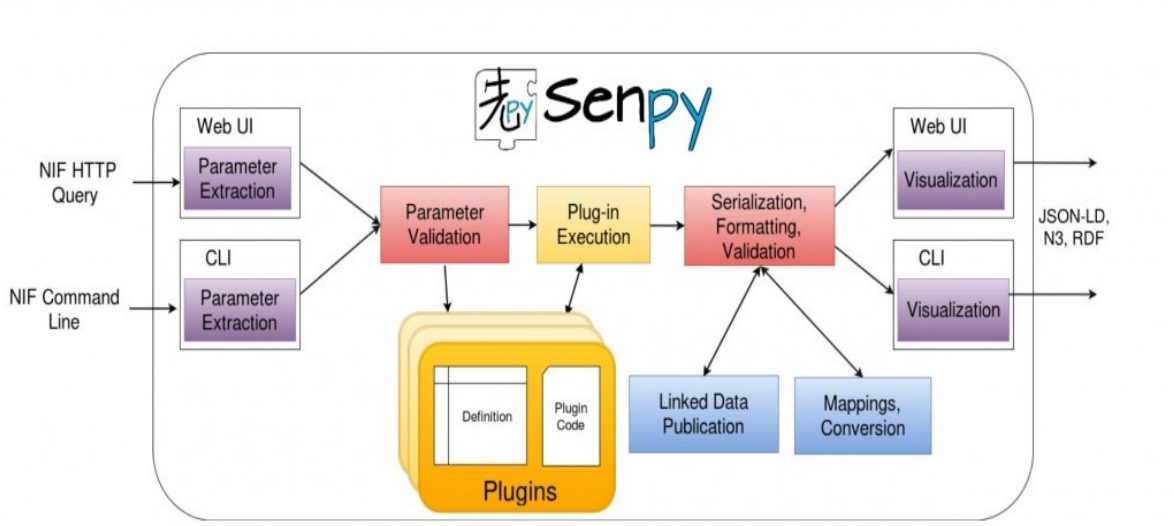

Figure 2.2: Modules involved in an analysis with the reference implementation of Senpy [14]

The framework consists of two main modules: Senpy core, which is the building block of the service, and Senpy plug-ins, which consists on a number of NLP algorithms. In the figure 2.2 we can see a simplified version of the processes involved in an analysis with the Senpy framework. The tool extracts the parameter from a NIF HTTP query and executes the code of the plug-in selected with the inputted parameters previously validated. Then, use models to output a linked data publication in the desired format.

2.4 TextBlob

TextBlob [2] is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. Particularly in this project, its use will be necessary for part-of-speech tagging.

Is it based on NLTK and pattern libraries. Initially, the library chosen for the task of part-of speech-tagging was pattern. But unfortunately, the Pattern is not implemented in Python 3, the programming language in our project. This is the reason why we picked Textblob as our used library.

2.5 Twitter API

This is a very important part of our project. The Twitter API is a tool developed by Twitter [6] that provides programmatic access to read and write Twitter data. Create a new Tweet, read user profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are in JSON format.

In this project, we are going to use this tool for the following requests:

- A list of the last tweets posted by a user
- A list of all the Twitter accounts followed by a determined user
- The number of followers of a user
- The response of if an account is verified
- The real name of the user (or at list the name that the user gave to the Twitter account)

2.6 MediaWiki API

This is the official API of Wikipedia [5]. It is a web service that provides access to characteristics, data, and meta data of the wiki using HTTP. Clients apply for particular requests specifying the desired parameters to obtain information from the huge amount of it accumulated in the Wikipedia.

In our project we are using this service to obtain information about the age of some famous people. The reason of this request will be described with more detail in the following section.

CHAPTER 3

Architecture

3.1 Introduction

In this chapter, we are going to explain the architecture of a learning system that predicts the gender of the users in Twitter. In order to do so, we are going to give a general view of the ideas, and also we are including some of the implementation details.

First, we are going to present a general overview of the project. We are going to identify the different subsystems and the steps that the algorithm follows for the correct functioning. Once this is done, we are going to focus on explaining more in depth each module separately, this way we can discuss the main tasks and goals of each one, and the reasons why they may or may not be necessary.

3.1.1 Overview

In this section we will present the global architecture of the project. Just a general sight of the main scheme of the modules that form the system. We are also giving a brief definition of the different subsystems that participates in the algorithm

We are going to explain the main core of the system. The goal of it is generating

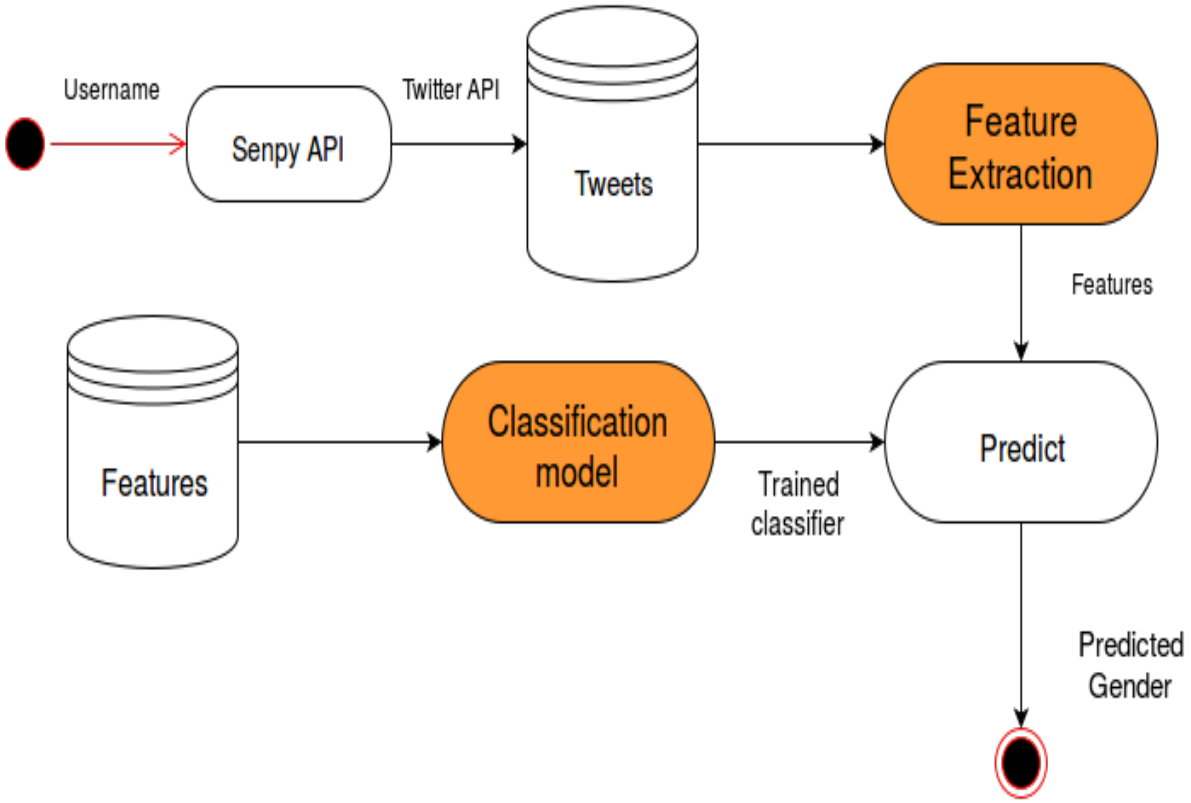

Figure 3.1: Architecture of the gender classification system

a classification system that can predict the gender of the writer of a Twitter account. In order to achieve that, we will implement a system based on the Machine Learning techniques explained in the previous section.

As we can see in the left side of the figure 3.1, the system receives the username of the account whose gender we are going to predict. Later, we use the API proportioned by Twitter to get a list of the last Tweets that the user posted. This way, we can say that we have both username and a list of tweets, now we have all the information that we need to begin the analysis. Later, we must convert that information into something readable by the model. In order to do so we must extract the so called features from the text. Those features is a way of representing the usable information that the raw data from the tweets into some vectors that later can be readable by the program. In this project we are going to use two types of features, those referred to the celebrities followed by the user, and those that are the results of analysing the tweets written by that user. The module in charge of that task is the Feature Extraction module, explained in the section 3.3 of this chapter.

Once he have all the features extracted, it is time to make the prediction. For that, we need a trained classifier to compare the features obtained from the account that we want

to Analise. In order to get that we will need a big amount of vectors with the features extracted so we can train the model. For that, we will need firstly a dataset of users. That dataset, is going to be the learning source of the model. From the information saved in the dataset, the system must be able to learn, and accumulate usable data, because in the future, that data must be used to predict the gender of the following users. In one of our datasets, we will have a list of usernames in Twitter, and also a great amount of Tweets from each user in the database. But on the other hand, the other dataset that we used in this project, only contains the list of usernames. Those datasets are explained with more detain in the chapter 4. The extraction of the information of the dataset is explained in the section 3.2. Once we have extracted the dataset information, we use the Extract Features module the extract those features and save them. It is important to save the features, because the system not only works with one set of features. IT works with various, so depending on the parameters of the request, we must use one set of features or another.

Once we explained how we got the set of features, in order to make the prediction, we must train the classifier. The classification model used in this system and the structure if it will be explained with detail in the section 3.4

Lastly, since we also want to approach the implementation of the classifier in one service entirely functional, we wrapped all this work into a Plug-in implemented in Senpy. Senpy is a platform that allows the creation of plug-ins that can be deployed in a Senpy server. This way we can execute our system as a service. The implementation of this Plug-in will be explained in the section 3.5

In the following sections, we will describe with more details all the subsystems involved in the project.

3.2 Extraction of the dataset Information

In this section we are going to explain the process of extraction of the information from the dataset. Firstly we must obtain the three datasets that are going to be used in the classifier from two different places. One from the “Kaggle” platform [10] and the other two were gotten from the “PAN” platform [8], that is a competition. More details from these datasets is given in the section 4.2.

From the three datasets used we can find two different formats of it:

- The first one is the dataset from the “Kaggle” platform. It is a large JSON file with a very large list of usernames in English. More than 20000 names in total. For each

username it has a gender associated with it. This dataset has a small problem, the obtaining of the tweets. In order to do so, we must use the Twitter API to request a list of the last 200 tweets (maximum number of tweets allowed by the API).

- The other two datasets from "PAN". Have a different format. It is a set of JSON files, each of them corresponds to a user. Each one of the files contains a list of 1000 tweets written by tat user, without any retweets and the username of the writer of those tweets. In both datasets, there is a file called truth.txt were all the gender of every user are included. This two datasets are in different languages. One is written in English and the other is written in Spanish.

Once we have the usernames and the list of tweets in both of the datasets available to be used, the next step would be extracting the features of those users. In order to do that, we will use the Extract Feature module. This module will be explained in the following section.

Lastly, with all the features extracted, we can proceed to save all the features for the main system. To save all those features, we use a pickle, that is an algorithm that allows us to serializing and de-serializing a Python object structure. This way we can save those features in a way that can be reached in the future. For each of the datasets we are going to generate one of those.

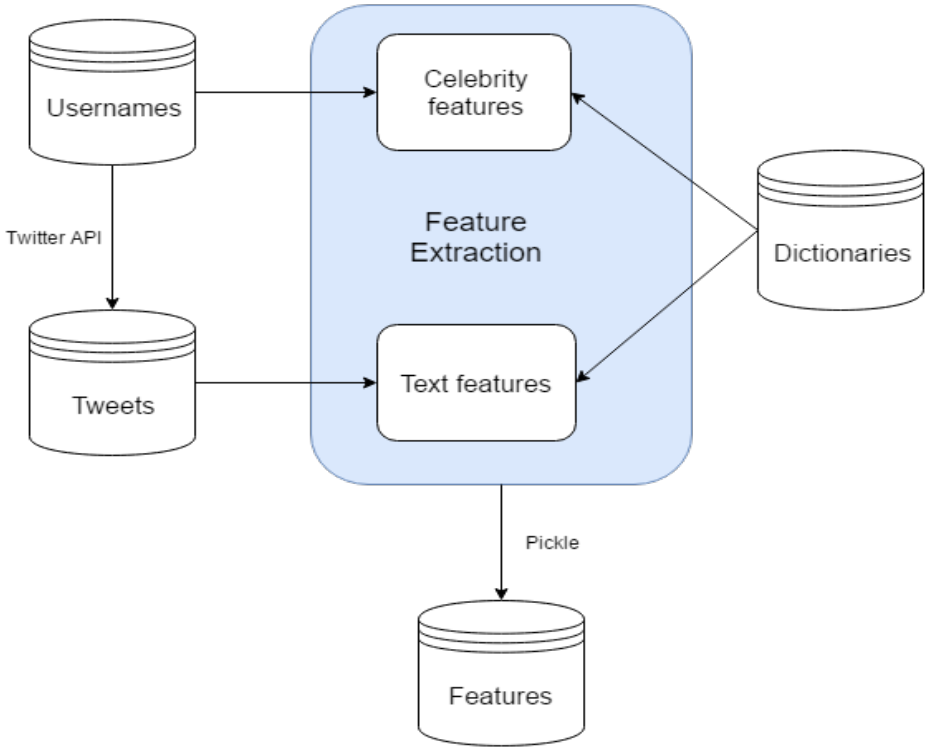

Figure 3.2: Architecture for saving the features

3.3 Feature Extraction

Once we have extracted all the information from the user, the next step is transforming that tweet and username into usable information. This is because from a computer's point of view, the raw text, and the usernames, gives no information at all. In order to get that information, we must perform some language processing techniques. Those consist in separating the text into words and perform some kind of selection of the words, so only the ones that provide information, or some kind of meaning to our purpose. To divide the raw text into words, we use the library TextBlob, explained in the section 2.3. And in order perform the selection of words that give us some kind of meaning, we must convert it into vectors that will contain all the information that will influence in the decision of the gender prediction. Those vectors are called the features.

This section explains the extraction of all those features. That will allow in the future that the classifier can understand the information extracted in the previous section. This task is one of the most important of all of them. The structure of this module can be seen in the figure 3.2

One of the tasks of this module is opening two dictionaries that are going to be needed

in the project. They are necessary for the extraction of the features explained later in this section. This is important because the dictionaries are in a TSV file, and it is the only way of translating the TSV files into something readable by python language. It opens up two of them:

- **Dictionary of names:** It is big database with a large amount of names in different languages, more than 60.000 names are included. In it there is a direct correspondence between the names and the gender that correspond to that name. This is going to be used for recognizing the gender of the celebrities that the user follows on Twitter. This dictionary was built by the German magazine "Heise" [16]
- **Dictionary of gender:** It is a huge list of words. To each word corresponds a weight. That weight symbolizes the probability that that word is used by a man or by a woman. This dictionary is going to be useful later for the extraction of the features, that is explained in the next section.

Once those dictionaries are opened we have the following information: the usernames with the large amount of tweets for each one and the two dictionaries ready to use. That is the information that we are going to need for the Extract Features module.

In this model, we have two types of features: features referred to the celebrities that the user follows and features regarding the characteristics of the tweets that the writer posts. Those two types are explained with more detail in the following sections:

3.3.1 Celebrity features

The work by Puneet Singh Ludu [15] explains that the analysis of the users that a person follows can give us information about that person's gender. More specifically the celebrities that this person might follow. If you analyze the gender and age of those celebrities the results can be even more accurate.

In this section we are going to explain the features that refer to that analysis of the features, and more specifically how to get the age and the gender of the celebrities that the user follows. In order to get that we use a module that can is represented in the figure 3.2. The steps followed by the module are the following:

- In the first step, the Extract Celebrities module takes the username of the account that we want to analyze. Then it uses the Twitter API to get a list of all the users that it follows.

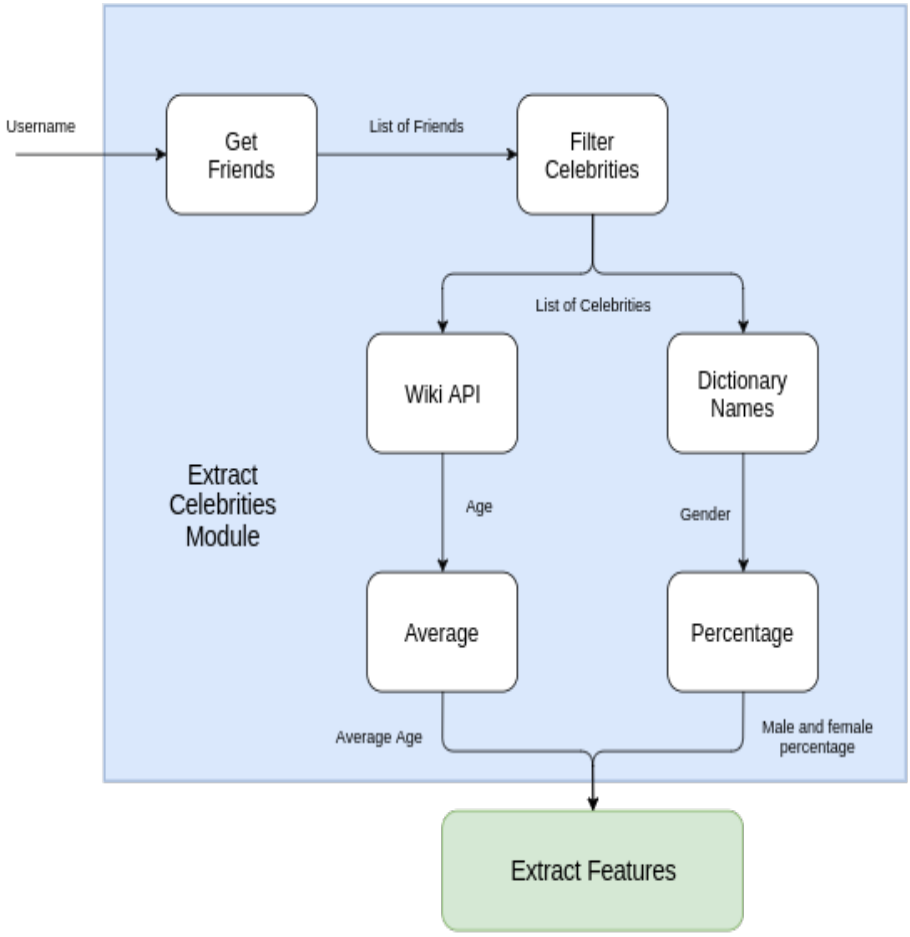

Figure 3.3: Architecture of the Extract Celebrities Module

- Secondly, knowing all the usernames of the accounts that he or she follows, it uses also the twitter API to filter from the list of accounts those that can be considered celebrities. It picks those who have more than 50.000 followers and are also verified by Twitter.

- Later, when we have the list filtered, he have to Know their real names, so we can check their age and gender. To do so, we use the twitter API again to get the name that they use as their real name. This is not 100% accurate, but it is reasonably acceptable.

- In order to get the age of the celebrity we use the Wikipedia API. With the real name, we can ask for it. Sometimes, when the name of the celebrity is not the same in Twitter than in Wikipedia, it does not work. But it also works a reasonable amount of times.

- Once you have the name and the age, the next step is knowing the gender of the

celebrity. In order to discover it, we use a very big dictionary, that we mentioned in the previous section, which includes more than 60.000 names with their respective gender. This way, knowing the name you can easily know the gender of that person just checking that big database.

- Finally, we have a list of the celebrities, with their ages and their genders. This way we can do the final step, that is getting the output for the next module. This is done by calculating the average age and the percentages of male and female celebrities.

Once we have calculated the average age and the percentages of male and female celebrities followed, we can add those three values to the list of features that are going to be read by the classifier.

3.3.2 Text Features

Once we have calculated the features that depend on the followed people by the user, we are going to focus on those that refer to the tweets that that person writes. To do so, we were inspired on the work of several authors, such as Francisco Rangel and Paolo Rosso [12] or Puneet Singh Ludu [15]. In this module, we are going to explain which characteristics of the tweets are important in the recognition of the gender and why. We will also extend in why those features are important.

A complete list of all the features that this module extracts is:

- **Capital Letters:** Men and Woman, according to some research [12] use different amount of capital letters in their tweets. This feature counts the amount of capital letters and compares them with the total amount of letters in the tweets.
- **Elongated words:** Sometimes, when people write tweets, they make some of the words shorter that they are supposed to be written. For example, some people write ASAP, when the meant "As soon as possible". Depending on the gender, the amount of those abbreviations might vary. This feature counts the number of those abbreviations.
- **Exclamations and interrogations:** According to the paper written by Francisco Rangel and Paolo Rosso [12], also the amount of exclamation and interrogation signs that the tweets contain, have a direct influence in the probability that a man or a woman wrote the tweet. This fact, is reflected in this feature that counts the number of both exclamation points and interrogation points used in the texts.

- **Omg, hearths, lols and Lmfao:** Some expressions used in the Internet writing, are very representative of Twitter. Those expressions can be also a representative factor that tells us some information for gender recognition. Some of those expressions are "OMG" (Oh my God), "¡3" shape (this characters represent a heart, "Lmfao" and "LOL"(both are expressions that represents laughter). We will count the amount of those expressions used and that will be considered a feature.

- **Emoticons:** According to the work of Puneet Singh Ludu [15], the amount of emoticons written in the tweets, can also be a factor in the final decision. But also, each emoticon has a score that represents the likelihood that it can be written by a man or by a woman. So for the emoticons, we have two features, one for the amount of the emoticons, and the other for the resulting score.

- **Mentions, hash tags, and re tweets:** This features, just counts the amount of mentions to other people that the user makes in their tweets, also counts the hash tags that he or she uses for complementing the tweets and also the amount of re tweets that their tweets have received. This three features represent the amount of activity and the quantity of interaction with other users Twitter. This can also be a factor in the gender recognition .

- **URLs and pics:** This feature counts the number of links to URLs and the number of pics that the texts have. This can be seen as the interaction of the user with the Internet outside Twitter.

- **Average text length and average word length:** According to Puneet Singh Ludu [15], the analysis of the average word length of the tweets written by the user, as well as the average length of the words written in the tweets, suppose an improvement in the information received for gender classification. This is why this two features calculate those two numbers and adds them to the classification algorithm

- **Extroversion:** According to the work of Francisco Rangel and Paolo Rosso [12], the different grammatical categories of the words have an influence on the probability of predicting the gender of the user. Some categories are more used by women and some other are more used by man. For example, nouns are more used by man, and verbs are more used by women. In order to define this feature, we divide the categories in two groups. The "Male" group and the "Female" group. This way, the number of words in every group can be counted. The final value is the subtraction of both countings.

- **Bf and gf words:** There are a certain amount of words, that implies that the

writer might be a man or a woman. Those are the words that makes reference to the sentimental partner (boyfriend, husband,...). Usually those words are written more times by the opposite gender of the person referred. For example, the word girlfriend is more likely that is written by a man that by a woman. These features count the amount of those words referenced to the male and female partner

- **PMI and male and female counts:** This feature and the following four make reference to the information of the gender dictionary. As we learned in previous sections, it is a big database with a big amount of words. Each of those words has a weight, that symbolizes the probability of each word of being used by a man or a woman. That weight is called the PMI. This feature is the value of the sum of all the values of the PMIs of every word. In the dictionary mentioned, there is also more information about the words. If the PMI value is negative, it is more likely to be written by a woman, and vice versa. Every word has assigned a tag, if the PMI is negative, the tag is "female", and if the PMI is positive, the tag is "male". These features also count the amount of words with the "male" tag. In the extreme cases that the PMI of a word has assigned its maximum value, both positive and negative, the tag is a bit different. In the case of being the maximum positive value, the tag is "smale". And in the negative case, the tag would be "sfemale". This feature counts the number of "smale" tags in the . This is a way of giving more value to this feature in comparison to the others. We use five features for the information extracted with the dictionary, a bigger amount of features that any other text analysis techniques.

Finally, once all the features are extracted, we can begin the next step in the process, the Classification model, that is going to be explained in the following section. That classification model is in charge of the training of the classifier and testing the accuracy of the algorithm.

3.4 Classification model

The extraction of all the features gives us a new dataset containing all the information that can be read by the classifier. However, all this information is useless without a classifier model, that can interpret all those features, and based on all that information, will be able to predict the gender of the user. For this task, we are going to make a selection of the right classifiers for the algorithm. We also need to have another thing in consideration, the amount of features with which we are going to feed the classifiers. This is because feeding the classifier with too many features, may cause that the classifier gets too much adapted

the the training set, and will only be able to recognize well some parts of that training set, but it will not be able to learn from new ones. For solving this last problem, a better explanation is given in the next subsection.

Now we will proceed to describe all the classifiers used in the algorithm. This system uses a combination of ten classifiers, that are already designed by the Scikit-Learn library. All of them are going to be described in the list below. Apart of using all of them separately, we decided to combine all of them. To do so, we use a function called "cross-validation" provided by Scikit Learn [9]. We call this function twice. This is because we combine those classifiers also twice using two different methods:"Hard Voting Ensemble" and "Soft Voting Ensemble". The difference between them is that in Hard Voting, we predict the final class label as the class label that has been predicted most frequently by the different classification models used. But in Soft Voting, on the other hand, we predict the class labels by averaging the class-probabilities of the classifiers.

This is a list of all of the classifiers used in the project [9]:

- **Lineal model classifier:** More particularly using the logistic Regression function. The probabilities describing the possible outcomes of a single trial are modeled using a logistic function.
- **Decision Tree classifier:** This function measures the quality of every split and makes the decision accordingly. To measure that quality it uses a maximum depth of 3.
- **Random Forest classifier:** It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. This particular one uses 100 estimators.
- **SVC:**This classifier uses Support Vector techniques. It Uses a subset of training points in the decision function (called support vectors), so it is memory efficient.
- **Naive Gaussian classifier:** This and the following classifiers are based on applying Bayes theorem with strong (naive) independence assumptions between the features. In this case, the assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution
- **Naive Bernoulli classifier:** In this other case, features are independent Boolean (binary variables) describing inputs.
- **Gradient boosting classifier:** This one builds an additive model in a forward stage-wise fashion. It allows for the optimization of arbitrary differentiable loss functions.

- **Ada Boost classifier:** It is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset, but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases

- **One vs. Rest classifier:** This strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes.

- **Extra Trees classifier:** This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Once we know all the classifiers that are going to be used, we must test the accuracy of the algorithm. In order to do so, we must follow the steps that are explained can proceed to explain what are the steps that the module follows to achieve the training and testing:

- **Splitting the features into training set and testing set:** This is done by a function of Skicit-Learn called "KFold". We divide the set in three, the first two form the training set, and the last one is the testing set.

- ***Training the classifier:*** Once we have all the features splitted, the next step is the training of the classifier. For this, Scikit-Learn uses the first two thirds of the features (the training set), that division was made with the KFold function, as we explained before. This way, the classifier can make an association with the features so it can predict future results.

- ***Testing the remaining features:*** When the classifier is trained with the first two thirds of the set, we have to test the performance of it. Do do so, we use the testing set. The classifier, comparing with the results that it has from the previous training, predicts the gender of the writers, whose features are in the testing set.

- ***Showing the results:*** Finally when the testing is done, the function compares the predictions of the classifier with the true genders of the writers, saved in the datasets. The percentage of success, is going to be the Success Rate of the classifier.

The KFold function also has an additional function. It does not only divide the features into training set and testing set and performs. It also makes all the possible combinations of all those three thirds into training and testing sets. The results are the average results of all those possible combinations. This way, the testing is more accurate, and reduces

the probability of false results due to training the classifier with a specific combination of features.

The following of those steps can be seen more visually in the figure 2.1 in the second chapter

3.4.1 Improving the algorithm

As we mentioned before, the amount of features with which we are going to feed the classifiers can't be very large. This is because feeding the classifier with too many features, may cause that the classifier adapts too much to the the training set, and will only be able to predict some very specific results, that are very similar to the training set, and it will not be able to learn from new ones. For solving this last problem, we are going to perform a solution based on feature selection.

Scikit-Learn provides another useful function for our system. This is the SelectKBest function. We use it for improving the classification algorithm. It selects the desired number of features that are more significant, and that have a bigger impact on the final decision. And eliminates the rest of them from the dataset of features extracted. This is very useful, because the classifier, can not know which of the features are more decisive. To know how many of the features must be eliminated from the model, we test one by one all the possibilities of features eliminated. And with all the results given for each test, we pick the best performance.

Using this function can suppose a big improvement in the performance of the algorithm. Sometimes we can reach and improvement of over a 5% or more. We can see an example of the difference of the results in the figure 3.3. In that example, we can appreciate that the best performance is achieved when only the most significant 17 or 19 features are considered.

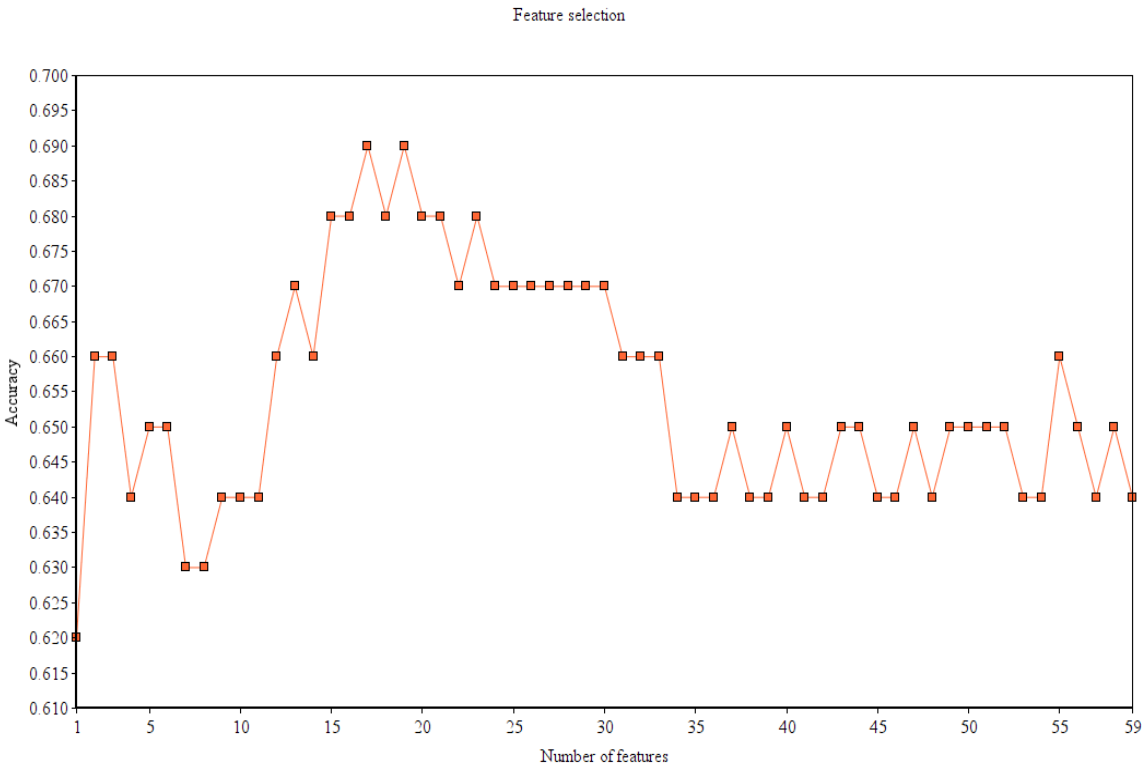

Figure 3.4: Feature selection results

3.5 Plug-in in Senpy

The purpose of this section is implementing the previous modules in one service entirely functional. Senpy allows the creation of plug-ins that can be deployed in a Senpy server. In this project, we have created one plug-in to execute our system as a service.

In order to implement the plug-in, a Python module was created gathering all the necessary modules and tools necessary to execute those modules. All the modules explained in the previous three sections are included in this Senpy plug-in.

The request in the Senpy plug in must include two parameters for the correct working of it. This is because depending on the type of service required, the classifier must be trained from a set of features or another. The first of the parameters is the language of the Twitter account introduced in the request. It can be in Spanish or English. The second parameter is the inclusion or not of the celebrities module. This is due to the limitation of

the request imposed by the Twitter API. The program gives us a better performance with those features but, it is not viable if we want to request too many times. This way, you can choose if you want to include those features or not.

The plug-in that we developed uses a semantic vocabulary called Foaf [4] (it is explained with more detail in the Chapter 2) in order to shape the information that the plug-in gives as a response. That response, that is the output of the plug-in, is going to be given in JSON-LD format. In the format of the response we can find that the information outputted is provided with a correspondence with the correct ontology for each case.

The object that the plug-in will give us as a response must have the following fields:

- **foaf:accountname:** This field contains the username of the account that we want to analyze. It is the input of the system.
- **foaf:gender:** This field contains the predicted gender of the user analyzed. It corresponds to the output of the system

We must conclude saying that the Foaf ontology is not registered in the Senpy environment. This is because originally, Senpy was designed for sentiment and emotion analysis purpose, not for gender recognition. But this is not not a problem, because we can just add manually to the response the necessary fields for the Foaf ontology.

CHAPTER 4

Evaluation of the Results

4.1 Introduction

In the project, in order to check the performance of the system, a series of different tests must be performed over the algorithm. The main goal of this section is to describe those tests, that were implemented with different datasets.

Once those test were performed, there is and important task of reading and interpreting the results. Comparing the different outcomes, we can arrive to conclusions about the performance of the classifier.

In those test we want to check the following issues:

- The importance of the amount of tweets read for each user
- The importance of a bigger dataset in the final outcome
- The improvement that supposes the inclusion of the features about the Celebrities followed by the writer, and if there is any, if all the resources that are necessary are worth it.
- The differences of the performance of the classifier when we change the language of

the tweets.

In this test we have used datasets in two languages: English and Spanish. We used two different datasets in English, and one dataset in Spanish. The characteristics of the different datasets are explained in the following sections.

4.2 Datasets

In this section the focus will be on those datasets that are in English. The results that are going to be exposed are divided in two different groups. Those without the improvement of the celebrities analysis and the others with that improvement in their features.

We are going to use two different datasets (Dataset A and Dataset B), both of them are explained with more detail in the following subsections

4.2.1 Dataset A

This first dataset contains data the following way. Consists of a group of several files. Each one of them contains the information of one user. From each file we can read a big set of tweets written by the user, without any re-tweets. Besides that information, there is a file called truth.txt, in which we can find the information of the genders of each one of the users mentioned previously. The amount of data contained on the dataset is the following:

- Contains a set of 436 files.
- Inside each file, the username of every one of the users is included.
- For each one of them, the dataset contains 1000 tweets written by that user.

This dataset dates from the 02/29/2016.

4.2.2 Dataset B

This dataset, is a public dataset from the platform Kaggle [10]. It works different from the first one. Instead of consisting of a group of files, it is all included in a single CSV file. In this file you can find a large list of usernames. This list is mapped with the gender of every one of them.

Using this dataset is a bit more tedious than the dataset A. This is because, in order to extract the tweets, for every user, the Twitter API must be used to get a list of the last 200 tweets that that user has posted. On the other hand, the list of users is much larger. There are more than 20000 of them listed in the dataset. This is the reason why this dataset has been used. We can test the classification algorithm with a much larger amount of examples.

We are using this dataset in three different ways:

- First, we are just considering the first 400 users to be able to compare it with the dataset A in amount of data.
- Second, we are taking a larger amount of users to test the importance of the quantity of them. Specifically we are going to take 4000 of them, ten times more users.
- Third, we are taking a much smaller amount of users with the same goal. This time only 40 users, more or less ten times smaller than the dataset A.

4.2.3 Dataset in Spanish

This last dataset contains data the same way as the dataset A. But with the difference that the tweets are written in Spanish instead of in English. There is also other difference, that is the amount of users. This time the there are more or less half of the amount of users in English, more precisely 253 of them.

This dataset dates from the 02/29/2016.

4.3 Tests performed

In order to check the behaviour of the system implemented and the improvements of the modules installed, we perform the following test with the following goals:

- Both datasets in English, A and B (400 users) without considering the features about the celebrities. This test is done with the main purposes of testing the results of the basic algorithm and testing the importance of the amount of tweets read for each user.
- Dataset B in its three modes, with 40, with 400 and with 4000 users. This one tests if the amount on users used to train the model has a real influence on the results

- Both datasets, A and B (400 users) considering the features of the celebrities followed by the user. This way, we can test the performance of the module Extract Features explained in the previous chapter .
- The Spanish dataset with and without the celebrity features. With this test, we check the accuracy of the algorithm performed on different languages.

The implementation of those test, as well as the conclusions extracted fro them are explained with more detail in the following sections:

4.4 Evaluation of the classifier performance

We are going to start with the results of the first test on the datasets A and B. But in the B, for this test, we are taking only 400 users, a quantity of them that can be compared with those in the dataset A.

This test has the main goal of just observing the results of the algorithm for the first time. Looking at the accuracy rates depending on the classifier and the possible variances and errors of the results. We are also going to compare both results and we are going to pay special attention in the variations regarding one main issue: the variation depending on the number of tweets extracted from each user. In the first dataset we are taking 1000 tweets per user, but in the second dataset, we are only taking 200. This is because at the time of extracting the tweets from the username using the Twitter API, it has a limit for the number of Tweets that can be extracted. It only let us take the last 200 tweets written by the user.

The results are given the following way. For each one of the 10 classifiers used, there is a result of the accuracy of it and next to it, we can see the Standard Derivation of the result. This symbolizes the grade of variance that the final result can suffer depending on the the test. There is also two values that are the combination of all 10 of the classifiers. The first one is called "soft voting ensemble" and the second one is called "hard voting ensemble". The difference between both of the ensembles is the following. In hard voting, we predict the final class label as the class label that has been predicted most frequently by the classification models. In soft voting, we predict the class labels by averaging the class-probabilities.

The results after running the whole system for both datasets are detailed in the table 4.1

From this first test, we can extract the following conclusions. We can see that the results

Table 4.1: Accuracy and Standard derivation without celebrity features

Classifier model	**Dataset A**	**Dataset B**
Lineal model classifier	0.65 +/- 0.01	0.61 +/- 0.04
Decision Tree classifier	0.60 +/- 0.05	0.59 +/- 0.04
Random Forest Classifier	0.70 +/- 0.01	0.65 +/- 0.03
SVC classifier	0.64 +/- 0.01	0.59 +/- 0.03
Naive Gaussian classifier	0.56 +/- 0.04	0.59 +/- 0.03
Naive Bernouilli classifier	0.57 +/- 0.02	0.60 +/- 0.02
Gradient Booster classifier	0.66 +/- 0.02	0.67 +/- 0.03
Ada Boost classifier	0.68 +/- 0.02	0.63 +/- 0.02
One vs. Rest classifier	0.64 +/- 0.01	0.59 +/- 0.03
Extra Trees classifier	0.71 +/- 0.02	0.66 +/- 0.02
Hard Voting Ensemble	0.69 +/- 0.02	0.65 +/- 0.02
Soft Voting Ensemble	0.67 +/- 0.02	0.64 +/- 0.02

of the algorithm implemented on the Dataset A are slightly better than those implemented on the dataset B. We can see that with some of the classifiers we get better results, but if we consider the combination of those classifiers (Hard Voting and Soft Voting), we can clearly see a small difference. This is caused by the difference of the tweets read by the program for each user, in the Dataset A, the number of tweets per user is 1000, but in the Dataset B, only 200.

We can also observe that the Standard Derivation in the second test is slightly bigger too. This is also caused by the number of tweets on each user. Since there is less amount of text to analyse, the results are less accurate and suffer more variations.

4.5 Evaluation regarding the amount of users

The second test is going to have in consideration just the dataset B. But we are going to test it different times, each one of them with a different amount of users. 40, 400 and 4000 respectively. The results are going to be given the same way as before, giving the results of every one of the classifiers and later the Hard voting and Soft Voting. Those tests will include both accuracy percentage and Standard derivation

In this test, we are going to compare the results of three different trials including different amount of users. This way we can reach our goal, that is testing the influence on the amount of users with which the classifier is trained on the accuracy of the algorithm.

The results of this test are given by the table 4.2

Regarding the results of this test, we can arrive to the following conclusions: First, we can see that in the test using only 40 users, there is a huge diminution of the accuracy of the results. And in the following two, with 400 and 4000 users, there is a tendency of improvement. Regarding this fact, we can say that the rhythm of improvement is pretty high. There is a more or less a 10% of difference between the first trial and the last one. Although we can not make that comparison, because the 40 users trial is not a realistic test. A classifier, would never be trained with a number of users so small. But we can check the other two tests. We can see almost a 5% of improvement. That number is a high improvement for just increasing the number of users utilised in the training of the classifier.

We also observe that the lack of a small Standard derivation in the previous test in the Dataset B with 400 users is clearly compensated with a large amount of users. The new Standard Derivation is in every classifier 0.01%. With all this data mentioned, we can conclude that if you have enough data in your datasets, increasing the number of users

Table 4.2: Accuracy and Standard derivation with different amount of users

Classifier model	**40 Users**	**400 Users**	**4000 Users**
Lineal model classifier	0.54 +/- 0.04	0.61 +/- 0.04	0.67 +/- 0.00
Decision Tree classifier	0.49 +/- 0.04	0.59 +/- 0.04	0.66 +/- 0.01
Random Forest Classifier	0.56 +/- 0.07	0.65 +/- 0.03	0.71 +/- 0.01
SVC classifier	0.51 +/- 0.10	0.59 +/- 0.03	0.62 +/- 0.01
Naive Gaussian classifier	0.64 +/- 0.04	0.59 +/- 0.02	0.62 +/- 0.01
Naive Bernoulli classifier	0.59 +/- 0.04	0.60 +/- 0.02	0.63 +/- 0.01
Gradient Booster classifier	0.67 +/- 0.04	0.67 +/- 0.03	0.71 +/- 0.01
Ada Boost classifier	0.69 +/- 0.06	0.63 +/- 0.02	0.70 +/- 0.01
One vs. Rest classifier	0.51 +/- 0.10	0.59 +/- 0.03	0.62 +/- 0.01
Extra Trees classifier	0.69 +/- 0.06	0.66 +/- 0.02	0.71 +/- 0.01
Hard Voting Ensemble	0.62 +/- 0.04	0.65 +/- 0.02	0.70 +/- 0.01
Soft Voting Ensemble	0.59 +/- 0.07	0.64 +/- 0.02	0.68 +/- 0.01

utilised in the training does not suppose a great increase of resources, but it does generate a big improvement on the results. This is the reason why increasing the number of users is very efficient if you do not use the celebrity features.

4.6 Evaluation regarding the celebrity features

The main goal of this test is checking the improvement of the algorithm with the inclusion of the Extract Celebrities Module. In order to do so, we evaluate the improvement of the classifier if the celebrities feature is considered, by comparing with the results of Sect. 4.4.

The test results are shown in Table 4.3.

From this third test, we can extract the following conclusions: we can see a small improvement in the behaviour of the algorithm in both datasets. But unlike in the previous test, the improvement is less significant, less than a 5%. In the Standard Derivations, on the other hand, we can not see any improvement or loss. This makes sense, because the celebrity features add some information to the algorithm, but the amount of data is the same in the test with and without this module.

For this test, we must remark that for a large amount of predictions, the algorithm takes a very long time. About 3 hours per 100 users more or less. This is due to the limitations that the Twitter API imposes to its users. You can only request a limited amount of request in a single period. This supposes a complication at the time of deciding if this algorithm is better with or without the celebrities.

Regarding that limitation and the data in the table 4.3, we can conclude that this celebrity features are useful if the amount of predictions that you have to make in a certain period is small. But in the other case, for a larger amount of predictions, the celebrity features improvement is not worth it for the amount of time require to make the predictions.

Table 4.3: Accuracy and Standard derivation with and without the celebrities

Model	**A without**	**B without**	**A with**	**B with**
Lineal model	0.65 +/- 0.01	0.65 +/- 0.04	0.67 +/-0.01	0.65 +/- 0.03
Decision Tree	0.60 +/- 0.05	0.61 +/- 0.04	0.62 +/-0.02	0.59 +/- 0.03
Random Forest	0.70 +/- 0.01	0.71 +/- 0.03	0.70 +/-0.04	0.66 +/- 0.03
SVC	0.64 +/- 0.01	0.63 +/- 0.03	0.67 +/-0.01	0.63 +/- 0.02
Naive Gaus.	0.56 +/- 0.04	0.55 +/- 0.03	0.58 +/-0.04	0.59 +/- 0.02
Naive Ber.	0.57 +/- 0.02	0.58 +/- 0.02	0.58 +/-0.02	0.61 +/- 0.02
Grad. Booster	0.66 +/- 0.02	0.67 +/- 0.03	0.69 +/-0.02	0.67 +/- 0.02
Ada Boost	0.68 +/- 0.02	0.63 +/- 0.02	0.70 +/-0.01	0.63 +/- 0.02
One vs. Rest	0.64 +/- 0.01	0.63 +/- 0.03	0.65 +/-0.02	0.63 +/- 0.03
Extra Trees	0.71 +/- 0.02	0.71 +/- 0.02	0.73 +/-0.02	0.66 +/- 0.02
Hard Voting	0.69 +/- 0.02	0.67 +/- 0.02	0.71 +/-0.02	0.67 +/- 0.02
Soft Voting	0.67 +/- 0.02	0.67 +/- 0.02	0.69 +/-0.02	0.68 +/- 0.02

4.7 Evaluation regarding the language used

And finally for the last test, we are going to test the Spanish dataset twice. The first time not using the celebrity features improvement, and the second time, that module will be included in the test.

This last test's goal is testing the repercussion in the accuracy of the algorithm of the language used by the writer. To do so, we compare the results of the tests with and without the celebrities of the datasets A and B (with 400 users), that include tweets written in English, with the results of the testing of the dataset in Spanish.

The results of this last test are all included in the table 4.4.

Lastly, from this test we can extract several conclusions. Looking at the results, we can see that the accuracy of the algorithm run in Spanish is a bit smaller than the tests in English, but not in a great amount, just around 3%. This might be due to various things: the first is that the algorithm works worse in a different language, and the second is that since in the Spanish dataset, there are less users, this can affect the performance.

Comparing this results with the results in the test mentioned in the section 4.4, we can conclude that 3% is more than it should decrease from 400 users to 200. So there is a worsening in the accuracy of Spanish tests, but that accuracy is almost no perceptible. This is due to some features that are missing using Spanish language, for example those that refer to the dictionary of English words mentioned previously in the last chapter.

Knowing that we can conclude saying that the system works well in both languages, English and Spanish. There is a small loss of accuracy in Spanish, but it is so small that it is barely perceptible.

Table 4.4: Accuracy In English and Spanish with and without the celebrity features

Model	**A without**	**B without**	**Spa without**	**A with**	**B with**	**Spa with**
Lineal model	0.65	0.65	0.65	0.67	0.65	0.68
Decision Tree	0.60	0.61	0.56	0.62	0.59	0.67
Random Forest	0.70	0.71	0.66	0.70	0.66	0.63
SVC	0.64	0.63	0.63	0.67	0.63	0.64
Naive Gaus.	0.56	0.55	0.59	0.58	0.59	0.61
Naive Ber.	0.57	0.58	0.61	0.58	0.61	0.57
Grad. Booster	0.66	0.67	0.64	0.69	0.67	0.66
Ada Boost	0.68	0.63	0.62	0.70	0.63	0.64
One vs. Rest	0.64	0.63	0.63	0.65	0.63	0.67
Extra Trees	0.71	0.71	0.66	0.73	0.66	0.65
Hard Voting	0.69	0.67	0.66	0.69	0.67	0.66
Soft Voting	0.65	0.67	0.64	0.71	0.68	0.67

CHAPTER 5

Conclusions and future work

5.1 Conclusions

We are going to compare the results obtained in our tests with the results obtained by other previous papers. In the work of Puneet Singh Ludu [15], the accuracy obtained using the celebrity features module was around 78%. This result is a bit larger than our results, that are around the 70%. That difference might be to to the differences in the datasets used. Now, looking at the work of Francisco Rangel and Paolo Rosso [12], we can see that the accuracy obtained by them is the 57%. Our results are better, but this is due to the difference of the features used by them. They only used a fraction of the features used in this project. With that information, we can conclude saying that the results obtained are not the best results achieved yet, but they have a decent accuracy if we compare them with other previous works.

Now analysing the results from our tests more in depth we can compare the two types of features used (text related features and the celebrity features). Those results show a slight improvement if the celebrity features are included but it is not a very big improvement. Less a 3% on average in the accuracy, which is really small. Apart form that, we must indicate that if we want to predict a large amount of user's gender, we will face with some limitations

handling those requests, because it would take a long time to process the information due to the limitations of the Twitter API, that we will describe in the previous section. With all these results, we can conclude that the adding of the module that extracts the celebrity features is provides a small improvement in the accuracy, but only in the requirements whose number of users is not very big.

We can compare too the accuracy of the algorithm regarding the language of the Tweets analyzed. This has a small impact on the results of the results, between a 3% and a 4% of difference between the Tweets in English and those written in Spanish. So in this aspect we can conclude that the algorithm works, with more or less the same accuracy in English than in Spanish, there is a small decrease on the performance, but not big enough.

5.2 Problems faced

During the development of this project we had to face some problems. These problems are listed below:

- **Limitations of the Twitter API:** During the implementations of the project we faced some problems with the performance of the Twitter API. Twitter limits the number of request to this tool to a restricted number in a period of time. But, since sometimes, we were dealing with great amount of data to be analyzed. That was a problem. For example, we could not implement the tests with Celebrity features included for a large number of users. Because the amount of time required in the simulation was simply too big.

- **Ontologies in Senpy:** Senpy is a platform designed for Sentiment and Emotion analysis, not for gender prediction. So the ontology that this project requires for their responses (foaf) is not implemented. That is not a major problem, but the results given by the plug-in, will not be as accurate as if they were the results of a Sentiment or Opinion analysis system.

- **Gender recognizing of the celebrities:** In order to recognize the gender of the celebrities, we first used the API Genderize [17]. But it also gave us lot of limitations, such as the number of requests in a period of time. But those limitations, unlike the Twitter API, were not solvable. This is the reason that we used a dictionary to perform that task.

5.3 Future work

In the following section I will explain the possible new features or improvements that could be done to the project.

Analyze the appearance in Twitter: One possible next step in the line of work could be, adding more aspects to the analysis. Between those we could include the appearance of the Twitter page of the users, analysing the color of the background or some other elements such as the color of the sidebar to predict more accurately the gender

Analyze the images of the profile: In order to recognize the gender of the user more accurately, we could try to analyze the images appearing in than Twitter account. Both the profile images and the images posted in the Tweets. With a correct analysis we could identify if the person that appears the most in the pictures could be a man or a woman.

Analyze the information given in the profile: Such as the description that every user have to give a short introduction to their account. The name that every user uses for being recognized in Twitter, and also analysing the usermane itself looking for information about the possible real name of the user.

Bibliography

[1] Official github of senpy. https://github.com/gsi-upm/senpy.

[2] Official on-line manual of textblob. https://textblob.readthedocs.io/en/dev/.

[3] Official page of json for liked data. http://json-ld.org/.

[4] Oficcial manual of foaf. http://xmlns.com/foaf/spec/.

[5] Oficcial on-line manual of the mediawiki api. https://www.mediawiki.org/wiki/API:Main_page/es.

[6] Oficcial on-line manual of the twitter api. https://dev.twitter.com/rest/public.

[7] Page of rdf in w3. https://www.w3.org/RDF/.

[8] Pan official website. http://pan.webis.de/.

[9] *Scikit-Learn on-line manual*. http://scikit-learn.org/stable/.

[10] User CrowdFlowder. Dataset from the kaggle plattform. https://www.kaggle.com/crowdflower/twitter-user-gender-classification.

[11] Alexandre Gramfort Vincent Michel Bertrand Thirion Olivier Grisel Mathieu Blondel Peter Prettenhofer Ron Weiss Vincent Dubourg Fabian Pedregosa, Gaël Varoquaux. Scikit-learn: Machine learning in python. the journal of machine learning research. 2011.

[12] Paolo Rosso Francisco Rangel. Use of language and author profiling: Identification of gender and age. 2013.

[13] William L. Hosch. Artificial intelligence. http://global.britannica.com/technology/machine-learning.

[14] Ignacio Corcuera J. Fernando Sanchez-Rada, Carlos A. Iglesias and ´ Oscar Araque. Senpy: A pragmatic linked sentiment analysis framework. 2016.

[15] Puneet Singh Ludu. Inferring gender of a twitter user using celebrities it follows. *airXiv*, 1, 2014.

[16] Heise magazine. Anredebestimmung anhand des vornamens. https://www.heise.de/ct/ftp/07/17/182/.

[17] Official website of Genderize. https://genderize.io/.