Universitat de Lleida

TREBALL FINAL DE MÀSTER

Estudiant: **Kevin Garcia Millan**

Titulació: Màster en Enginyeria Informàtica

Títol de Treball Final de Màster: Detecting political bots on Twitter social network

Director/a: Francesc Giné de Sola, Magda Valls

Presentació

Mes: Octubre

Any: 2020

Abstract

With the emergence of new technology and media such as social networks, new methods of advertising or expanding ideas appear. These new methods use these networks to reach many people and thanks to the automation provided by bots, they can target specific users who may be prone to these ideas or advertising. In our case we will study how bots are used in election campaigns and use automatic learning techniques to detect them among other users.

Keywords: Twitter, Social networks, Electoral campaigns, Bots

Contents

List of Figures

1 | Introduction

In this work we are going to study the intrusion of bots in Twitter social network. In the first chapter, we will set out the context of our problem and propose the objectives and steps for our study

1.1 Background

In 2004 a website known as TheFacebook is launched, with features like a photo gallery, personal biography and online messages. This was a site exclusive for the students at Harvard University but in 2006 it was opened to the whole world receiving thousands of new users and caching the interest of big technology companies such as Microsoft in 2007. Nowadays it is the biggest social network and one of the most visited websites with 2,5 Billion monthly users. With such immense user traffic, it is evident that social networks are the ideal place to meet with other people, share ideas, announce meetings, etc. With these features it is easy to create internal communities of fans of music, movies, sports, etc. In the same way celebrities can use these tools to contact their followers or gain more audience.

In 2008, Barack Obama started his election campaign using up to 15 social networks, among which Twitter was the most successful, quickly reaching the top 10 users. Today, the former president of U.S. has more than 120 million followers on Twitter, being the user with the most followers, and with another 54 million on Facebook. But this campaign had controversy, he was accused of using micro-targeting techniques, which consist of modifying

political discourse to address issues of concern to the voter and respond based on information extracted from their profile. All this using the messages and profiles of the users without their consent.

Another political case is the 2016 United States presidential election. In which social networks were also used, but this time under the more aggressive strategy of candidate Donald Trump, they were used to distribute messages that would discredit his opponent Hillary Clinton, These messages were mostly distributed by Bots. It is believed that 50,000 bots of Russian origin spread up to 2 million tweets [Twia], in which the number of hashtags and mentions with the word Trump increased enormously. Due to the great commotion produced by these events, the providers of Internet services such as Twitter, Facebook or Google have decided to impose measures to detect bots and combat false news.

1.2 Case of Study

Bots by definition are programs that perform automatic tasks on the Internet, and the use of bots in Twitter social networks is no only exclusive at the political field. There are another fields where it is widely used, such as marketing pushing tweets tweets with advertising or reading and collecting information from consumers. In fact, any field where the opinion of users can be accessed and influenced, is a target for bots.

Society has ended up accepting this manipulation as the norm for future campaigns, which is why many researchers have launched news studies on social networks with the purpose of seeing if it is really possible to use this information to predict the winner or to detect and unmask dishonest practices of the political parties.

The Spanish elections on April 2019 are taken as a case study. Spain has long been a country with 2 major political parties, PP (Partido Popular) and the PSOE (Partido Socialista Obrero Español), center-right and center-left parties respectively. Traditionally, these parties have been governing alternatively in the last decades, either with absolute majority or even with the support of regional parties. However new parties have emerged since 2016

willing to contest their supremacy, Ciudadanos a center-right party, Podemos now called Unidas Podemos (UP) positioned on the left. In the November 2019 elections, another party gained importance, Vox, a Spanish nationalist party that followed Trump's campaign strategy, due to its great use of social networks, patriotic discourse and confrontation with traditional media.

The arrival of these new political parties also creates a break between traditional and modern methods of electoral campaigning. With this case study we will be able to see if the use of bots is repeated as it has happened in other countries such as the United States.

1.3 State of the art

The importance of social media has not only attracted the interest of individual researchers or educational institutions, but has also reached large technology companies or even government agencies [Sub+16]. As expected, a large number of studies have been carried out, all of them touching on different topics, from a person's voting intention or opinion on situations or events in society.

By the nature of the problem, machine learning algorithms are used for classification between human and bots. Although no preferred or superior method has been established Random forest is widely used and carries very accurate results [Rod+20].

Another aspect that we can see is that various works use different features in the user profile and behaviour when looking for clues to the classification [Dav+] mostly the account usage, the frequency of posting messages [Chu+12], detecting spam in the user messages [SS18].

1.4 Previous work

This project is born from a previous work, called "Real Time Classification of Political Tendency of Twitter Spanish Users based on Sentiment Analysis" by

Marc Solé [Sol+18] in which the Tweets Analysis Strategy (TAS) is exposed. TAS calculates user sentiment of a tweet by using Positive and Negative words Matching (PNM) and Neural Networks Strategy (NNS), the latter being the most accurate, thus being able to predict political trends.

This work concluded with two proposals to expand, the first to detect and model relationships between users and the second to use TAS to detect the level of intrusion by bots in political conversations on social networks. Our purpose is to pick up where they left off, and continue with one of these proposals by detecting bots on Twitter in a political environment.

1.5 Objectives

In this work we will implement and search for the most effective alternative methods for the detection of bots. So our steps will be:

- Create a data set for training and a second one for test against real data.
- Identify which features to use for the detection of bots, from the user's profile and message history.
- Quantify or transform these features in order to use them on the selection algorithms.
- Apply relevant techniques to detect the bots and analyze classification results.

1.6 Document Structure

This work is divided into 6 chapters.

- The **First Chapter** reviews the current events that have led to this work, mentions some of the research or techniques that have already

been used in similar works and proposes the final objectives, as we have already seen.

- The **Second Chapter** talks about the different types of users in social networks and some markers used to distinguish them.
- The **Third Chapter** defines the most important mathematical algorithms, as well as some small examples in order to understand them better.
- The **Fourth Chapter** discusses the tools used, in this case the different programming libraries used and programs.
- The **Fifth Chapter** is the results of the algorithm training tests as well as the actual results against real data.
- And finally the **Sixth Chapter** will contain the Conclusions as well as point out the next steps to make or suggest next improvements.

2 | Users and their behavior

An estimated 4.57 billion people have Internet access worldwide [Kem]. Of the traffic generated by 53.3% corresponds to mobile devices, 44% to computers and the rest corresponds to tablets and consoles. The most common uses of the internet are email communication, information searches, audiovisual entertainment, online shopping and social networks. For example Facebook has 2.5 billion unique users (without making a distinction between humans, business pages, or bots) monthly, YouTube has 2 billions and Instagram has 1 billion.

The large number of users in these social networks is a problem, and therefore it is necessary to divide them into types according to their uses of social networks, some of these uses are being in contact with friends and family, following famous people, entertainment, portal news, to follow products or access consumer service, among others. Their interaction with these functions is also important, whether they are active users who communicate openly and frequently with other users or passive users who hardly create messages or content, limiting themselves to only reading or marking some messages as favorites.

2.1 Characteristics and differences

Social networks are inhabited by billions of users, human or not, but thanks to several features that they share between them we can differentiate them into various categories. These are some of the most common:

- **Bots**: A bot is an application created to automatically perform repetitive tasks faster than a human being could do. In the context of social networks, a bot can perform tasks such as publishing content from web pages, such as a newspaper that publishes a link to the full story on its main website, bots that answer questions or perform tasks requested by users when called, publish periodic information such as weather, or messages to other users within certain communities.

- **False follower**: This type of automated user, as the name indicates follows important or influential people by giving the appearance that they are more important than they really are, in addition to marking these people's tweets as favorites affecting the popularity rankings of that opinions [Gur+16], and ultimately they rarely comment on or respond to other users. These types of bots are among the most numerous. Due to their low profile and little contact with users, it has been decided to ignore them as the scope or popularity of tweets is not covered in this study.

- **Lurker**: The Lurker is a legitimate user that can be easily confused with a false follower. They use Twitter as a mean to learn about current events or their hobbies, but without getting involved in discussions or comments with other users, but still mark the messages as favourites for further reading.

- **Troll**: Trolls are human users who post comments and images of sensitive topics, or with an out of the ordinary opinion with the purpose of causing a commotion within a community. So that the rest of the users respond to the Troll's comments with equal intensity, making the discussion deviate from the topic, fill with hateful comments or lose interest until a moderator has to close the thread or delete the Troll's messages. Trolls are the best known outside the social networks, thanks to the great media coverage they can achieve in the traditional media such as television, radio and newspaper, either by the large number of people who are able to annoy or the large and complex maneuvers they are able to create when working in group.

- **Cyborg**: cyborgs can be human-assisted bots, or human-assisted bots, according to the purpose of the account. They have the same features and functions as bots by launching certain automated messages, but

under certain circumstances the account creator takes control of the account to participate in discussions which allows for easier generation of friendship connections with real users [Chu+12], something that is rarely seen in those with bots, as they usually have no followers, except legitimate and beneficial bots such as a news site.

2.2 Behavior of political Bots

Bots by themselves are not harmful, it all depends on the person who created them and for what purpose. In our case, the use of bots in social networks during political campaigns, we can understand through empirical observation some of their behavior during several real cases.

One of the most controversial cases was the 2016 presidential election in the United States, in which approximately 50,000 bots of Russian origin sent up to 2 million tweets [Twia], specially at the end of the campaign, in which the number of hashtags and mentions with the word Trump increased enormously. Due to the great commotion produced by these events, the providers of internet services such as Twitter, Facebook or Google decided to impose measures to detect bots and combat false news.

The most related we have to our field of study, are the Spanish elections of 2019 [Pas+20]. In which the emerging parties of Ciudadanos, UP and Vox use the largest number of bots, comparing them with traditional parties such as the PP or the PSOE. We were able to see similar behavior to elections in other countries, increasing the number of bots and messages in recent weeks, especially on the day of the traditional main electoral debate between the main parties, one of the main political events in Spain.

2.3 Chosen features

With the previously exposed case studies and user examples we can deduce the behavior of bots in a political environment. This allows us to know

which are the characteristics and features in the bots that will help us to detect them. Taking into consideration other previous work on the subject [Sub+16] [Rod+20], we choose the most significant:

- **Post frequency**
  - **Definition:** Entropy of post frequency between 0 and 1. If is near 0 it means the post frequency is regular, but if is near 1 it is irregular and hardly predictable.
  - **Justification:** Bots tend to repetitive behaviors.
- **Spam**
  - **Definition:** Percentage of similarity with other spam messages.
  - **Justification:** Some bots are designed to give automatic answers according to which combination of topics, so looking for these repetitions will tell us if it can be a bot or not.
- **Time of Creation**
  - **Definition**: Its account was created during the campaign
  - **Justification:** These bots are created with the purpose of massively spread a specific political discourse in the election season. These bots will be created during or just before starting the elective season [Gur+16].
- **Follower-Friend ratio**
  - **Definition:** Proportion between the number of followers for each friend.
  - **Justification:** Bots have a tendency to follow all those who follow them, while humans follow their real-life friends getting a ratio close to 1, or celebrities who don't follow them back having a low ratio in this case.
- **URL ratio**
  - **Definition:** Ratio of tweets that have a link to a web page.

  - **Justification:** Bots will post links to newspapers or news websites that share the same ideology and have an article about this false news.

- **Tweets per hour**
  - **Definition:** Maximum number of tweets posted in a single hour.
  - **Justification:** Bots are capable of outperforming humans in different tasks, for example in reading and replying tweets. So numerous tweets in the same hour can be expected from a bot account.

- **Mention ratio**
  - **Definition:** Ratio of tweets that have a mention to another person.
  - **Justification:** As political bots, they are very likely to mention their party's accounts or important members from it.

- **Hashtags ratio**
  - **Definition:** Ratio of hashtags per tweet.
  - **Justification:** They will resend hashtags about political events or occurrences.

3 | Analysis and classification algorithms

To determine whether a user is a human or not it is necessary to study their behavior using their tweets and other characteristics of their profile. Due to the impossibility of checking every user it has been decided to use an automated process for checking, and the accuracy or veracity of this process will be measured using manually checked sets and other similar detection tools.

3.1 Analysis methodology

The process of detecting bots has 3 key steps which can be seen in the Flow Diagram in Figure 3.1. The first is the collection of users and their tweets through the Twitter API and stored in an internal database for easy access, secondly we transform the users and tweets from the database into quantifiable variables for example the Message Entropy or the Spam/Ham, finally we use the Random Forest algorithm to classify the users in bots or humans.

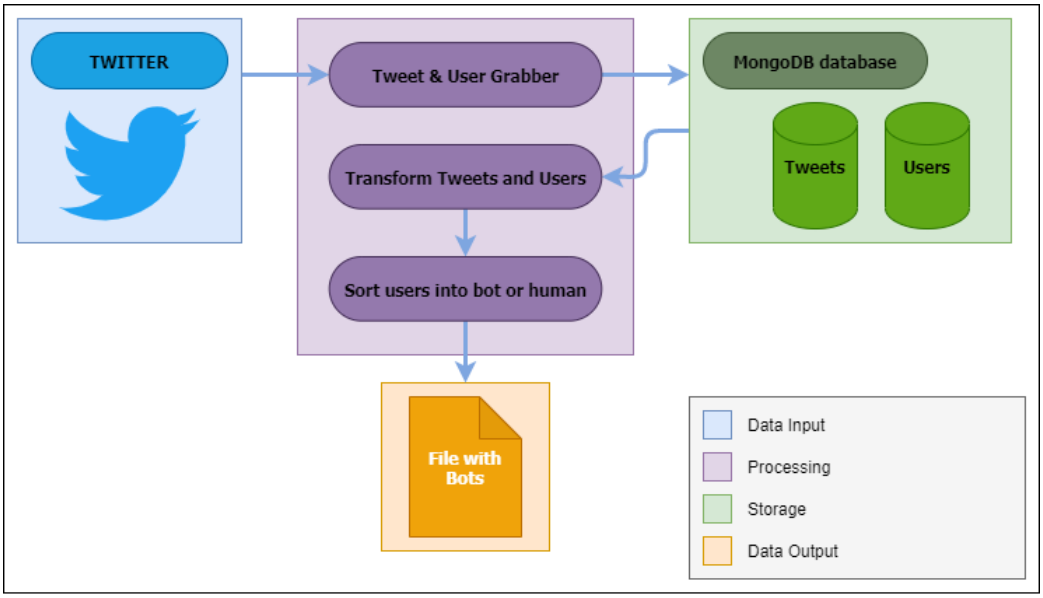

Figure 3.1: Flow Diagram

To obtain some of the user characteristics needed for the classification of bots, for example entropy or spam percentages, it is necessary to calculate them because they are not available from the raw data. The rest of features described in Section 2.3 can be obtained directly from the raw data or with little work. The following section will explain the algorithms used to calculate these and the algorithm used for the final step the classification.

3.2 Message entropy

Entropy is the measure of the uncertainty of a variable and its outcomes or the regularity of a data series, near 0 meaning the data is regular and predictable. Different methods exists to calculate the Entropy for different types of data, in our case, for time series we can use the approximate entropy or the sample entropy a modification of the later [DM19]. To calculate the approximated entropy the steps are:

- Step 1: Collect the data $U = (u_i...u_n)$.
- Step 2: Select m the length of the expected pattern and r, the tolerance of the values.
- Step 3: Form a sequence of vectors $x_1...x_{N-m+1}$ each one defined by $x_i = [u_i...u_{i+m-1}]$.
- Step 4: Calculate.

$$C_i^m(r) = \frac{(the\ number\ of\ vectors\ x_j\ with\ distance\ to\ x_i < r)}{(N - m + 1)}$$

  The distance is calculate as:

$$d[x, x^*] = max_a[u(a) - u^*(a)]$$

- Step 5: Calculate.

$$\Phi^m(r) = (N - m + 1)^{-1} \sum_{i=1}^{N-m+1} log(C_i^m(r))$$

- Step 6: Finally we get the Approximated entropy.

$$ApEn = |\Phi^m(r) - \Phi^{(m+1)}(r)|$$

An example of this algorithm in our case, the entropy of the tweets will be as follow:

U is a series of data of N elements that represents the number of tweets by the user made in the same spaces of time, for example how many tweets are sent every day, in a month.

$$U = [4, 1, 1, 1, 4, 5, 5, 4, 1, 1, 1, 4, 5, 5, 4, 1, 1, 1, 4, 5, 5, 4, 1, 1, 1, 4, 5, 5]$$

if we think in the normal everyday of the user,we can theorize that it will have a similar behavior between weeks, so be will take $m = 7$ as m is the length of data compared and $r = 3$.

Next in the Step 3, we create a the sequence of vectors $x(1)...x(N - m + 1)$

$$x_1 = [4, 1, 1, 1, 4, 5, 5]$$
$$x_2 = [1, 1, 1, 4, 5, 5, 4]$$
$$...$$
$$x_{22} = [4, 1, 1, 1, 4, 5, 5]$$

In the Step 4 its calculate from $C_1^m(r)$ to $C_{22}^m(r)$, for this example we calculate only $C_1^7(r)$. First we need to calculate all the distance from $x(1)$ to the other vectors:

$$d[x(1), x(1)] = max(|4-4|, |1-1|, |1-1|, |1-1|, |4-4|, |5-5|, |5-5|) = 0$$
$$d[x(1), x(2)] = max(|4-1|, |1-1|, |1-1|, |1-4|, |4-5|, |5-5|, |5-4|) = 3$$
$$...$$
$$d[x(1), x(22)] = max(|4-4|, |1-1|, |1-1|, |1-1|, |4-4|, |5-5|, |5-5|) = 0$$

d[x(1), x(1)]	0	d[x(1), x(12)]	4
d[x(1), x(2)]	3	d[x(1), x(13)]	4
d[x(1), x(3)]	4	d[x(1), x(14)]	3
d[x(1), x(4)]	4	d[x(1), x(15)]	0
d[x(1), x(5)]	4	d[x(1), x(16)]	3
d[x(1), x(6)]	4	d[x(1), x(17)]	4
d[x(1), x(7)]	3	d[x(1), x(18)]	4
d[x(1), x(8)]	0	d[x(1), x(19)]	4
d[x(1), x(9)]	3	d[x(1), x(20)]	4
d[x(1), x(10)]	4	d[x(1), x(21)]	3
d[x(1), x(11)]	4	d[x(1), x(22)]	0

Table 3.1: Table of distances for $C_1^7(3)$

We count all the distances which are lesser than r, in our case be obtain 10, from all the previous distances: $C_1^7(3) = \frac{10}{22}$ Calculate the remainder $C_i^7(3)$

$C_1^7(3)$	$10/22$	$C_{12}^7(3)$	$9/22$
$C_2^7(3)$	$10/22$	$C_{13}^7(3)$	$9/22$
$C_3^7(3)$	$9/22$	$C_{14}^7(3)$	$10/22$
$C_4^7(3)$	$9/22$	$C_{15}^7(3)$	$10/22$
$C_5^7(3)$	$9/22$	$C_{16}^7(3)$	$10/22$
$C_6^7(3)$	$9/22$	$C_{17}^7(3)$	$9/22$
$C_7^7(3)$	$10/22$	$C_{18}^7(3)$	$9/22$
$C_8^7(3)$	$10/22$	$C_{19}^7(3)$	$9/22$
$C_9^7(3)$	$10/22$	$C_{20}^7(3)$	$9/22$
$C_{10}^7(3)$	$9/22$	$C_{21}^7(3)$	$10/22$
$C_{11}^7(3)$	$9/22$	$C_{22}^7(3)$	$10/22$

Table 3.2: Table of $C_i^m(r)$

In step 5 we can calculate $\Phi^7(3)$

$$\Phi^7(3) = (22)^{-1} \sum_{i=1}^{22} log(C_i^7(3)) = 0.8459$$

Finally in step 6 we have to repeat all the previous step to calculate for m+1

$$ApEn = |\Phi^7(3) - \Phi^8(3)| = |0.8459 - 0.8473| = 0.0014$$

3.3 Spam/Ham classifier

A Spam/Ham classifier refers to any technique or algorithm used to detect unwanted emails, a technique that has been extended to tweets. There are many techniques [Bhu+18] such as Neural Networks, Random Forest, Support-Vector Machines, k-nearest neighbors algorithm, Naive Bayes, etc. Naive Bayesian are learning algorithm which returns the probability of an instance of belonging to one group or another. As we have seen in other works, Naive Bayes is a widely used algorithm that also grants quite high precision rates, if not among the best [Bhu+18] [AS11].

Naive Bayes solves the problem of classifying elements represented by a vector $x = \{f_1, f_2, f_3 ... f_n\}$ made of n features. Returning the probabilities of the element x to belong to each of the k classes in C_k, in our case Spam or Ham.

$$p(C_k|x) = \frac{p(C_k)\ p(x|C_k)}{p(x)}$$

It can be expressed as:

$$posterior = \frac{prior * likelihood}{evidence}$$

- **posterior**: Probability of the element x to be of the class k
- **prior**: Probability of the class k.
- **likelihood**: Probability of the features of the element x belongs to the class k.

$$p(x|C_k) = p(f_1|C_k) * ... * p(f_n|C_k)$$

- **evidence**: Variable that helps us normalize the final value, so that the set of probabilities for each class approaches 100%, which is the probability of appearing all the features with a determinate value at the same time.

$$p(x) = \sum_k p(C_k)p(x|C_k)$$

Next we will see how it is applied to our case. For example, we have taken a training data-set made by 10 tweets which have been classified manually and a message M = "Vote PartyA" that we want to classify:

Human tweets

- T1: Vote PartyA
- T1: Vote None
- T3: Vote PartyA
- T4: All Thief
- T5: Hate-speech Thief
- T6: Vote PartyB

Bot tweets

- T7: PartyA Thief
- T8: Vote PartyB
- T9: PartyA Hate-speech
- T10: Vote PartyB

With this small sample we can get the following tables, which will help us solve the problem. The first one is the probability that a message belongs to Class and correspond $p(C_k)$, being k the human or bot classes and corresponds with the prior from the Naive Bayes function.

Class	Probability
C_{human}	6/10
C_{bot}	4/10

Table 3.3: Table of Class probabilities

Our second table shows the probability of each word belonging to each group or $p(x \mid C_k)$ being x the word and k the class and corresponds to the likelihood from the Naive Bayes function.

Word	Probability in C_{human}	Probability in C_{bot}
$PartyA$	$2/6$	$2/4$
$PartyB$	$1/6$	$2/4$
$None$	$1/6$	$0/4$
$Vote$	$4/6$	$2/4$
$Thief$	$2/6$	$1/4$
$Hate-speech$	$1/6$	$1/4$

Table 3.4: Table of word probabilities in each Class

And finally $p(x)$ or the evidence, the probability of the features of x to appear in that combination, or in our case the probability of the message M expressed as $p(M)$.

$$p(M) = p(Vote\&PartyA) = \frac{6}{10}\frac{4}{6}\frac{2}{6} + \frac{4}{10}\frac{2}{4}\frac{2}{4} = \frac{7}{30}$$

Now we can calculate the probability of being a human or a bot:

Human Probability

$$p(C_{human}|M) = \frac{p(C_{human})\ p(M|C_{human})}{p(M)}$$

$$p(C_{human}|M) = \frac{p(C_{human})\ p(Vote|C_{human})\ p(PartyA|C_{human})}{p(M)}$$

$$p(C_{human}|M) = \frac{\frac{6}{10}\frac{4}{6}\frac{2}{6}}{\frac{7}{30}} = 0.5714$$

Bot Probability

$$p(C_{bot}|M) = \frac{p(C_{bot})p(M|C_{bot})}{p(M)}$$

$$p(C_{bot}|M) = \frac{p(C_{bot})p(Vote|C_{bot})p(PartyA|C_{bot})}{p(M)}$$

$$p(C_{bot}|M) = \frac{\frac{4}{10}\ \frac{2}{4}\ \frac{2}{4}}{\frac{7}{30}} = 0.4286$$

The message M will be consider to be a human message

3.4 Random forest

With the two previous algorithms we can quantify the users, and together the characteristics explained previously in chapter 2, we can use these to classify them in bots or in humans. For this last step, we use the Random Forest algorithm to classify the users, it has been chosen for its great precision[Rod+20].

The Random Forest[Bre01] is mainly based on the creation of a large number of decision trees, using different subset of data to train them. In order to create it's nodes and leafs, it calculates the Gini impurity of the features. This Gini impurity is the likelihood of classifying wrongly a new random element in the different classes C with $p(i)$ the probability of a element to fall in a class, and is calculated as follows:

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

For example having a Gini impurity of 0 means that it will always predict correctly the group of the new random element, because of that the first feature for the creation of the decision tree will be those with less impurity.

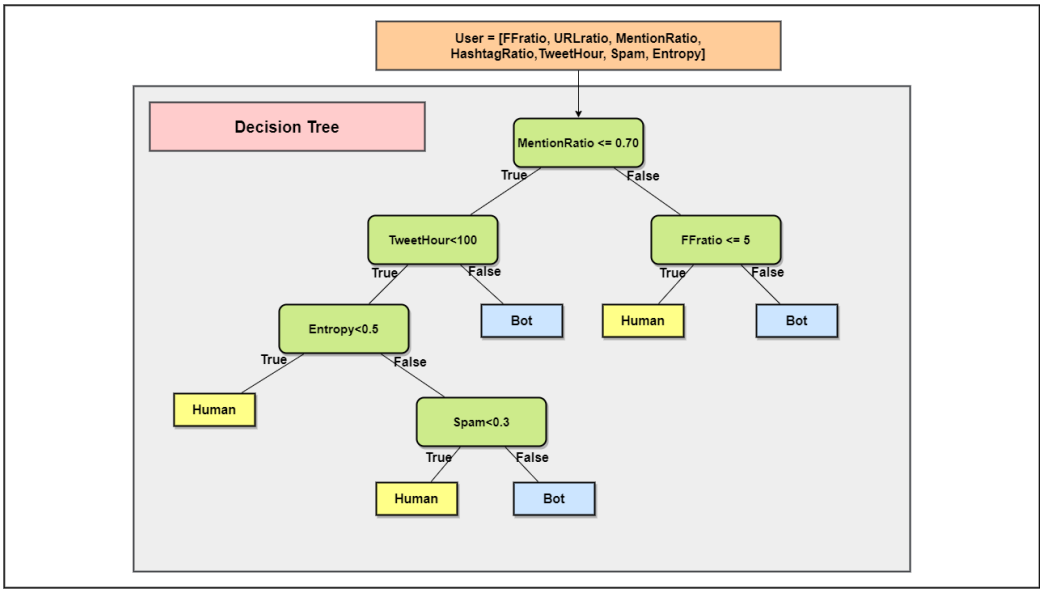

Figure 3.2: Example of a single decision Tree

To avoid Overfitting, the algorithm uses different features in each subset, this technique is call feature bagging. An example can be seen in Figure 3.3 in which the training data is separated into different subsets, which will be used to create the different trees that make up the random forest.

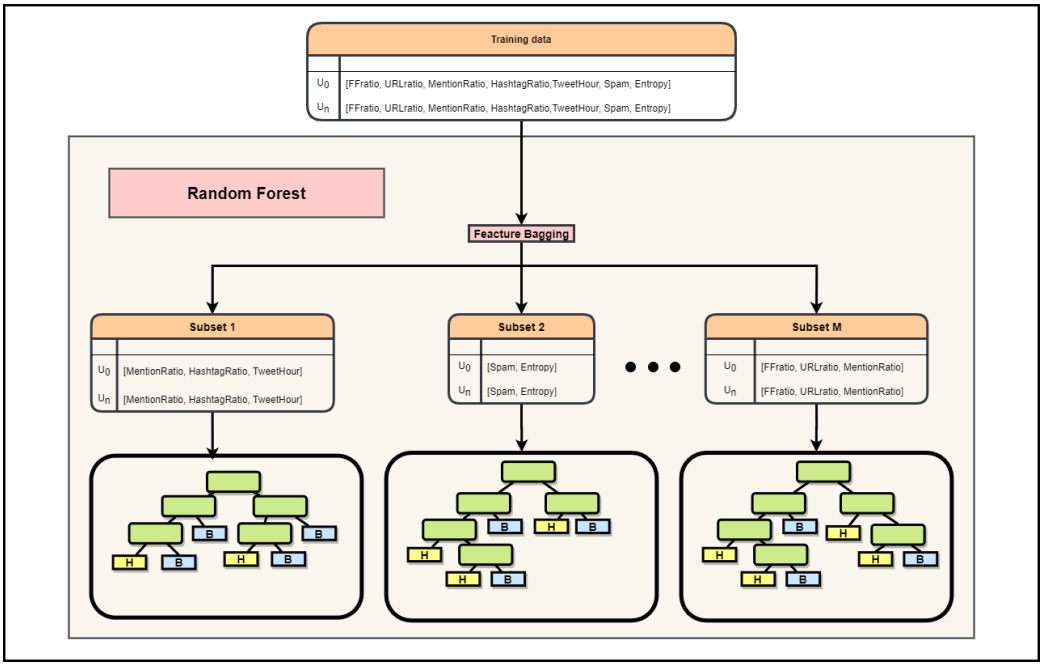

Figure 3.3: Random Forest and feature bagging

Once trained, when an unknown user arrives at the Random Forest, each tree will be given a different evaluation and through a simple majority the final classification will be selected.

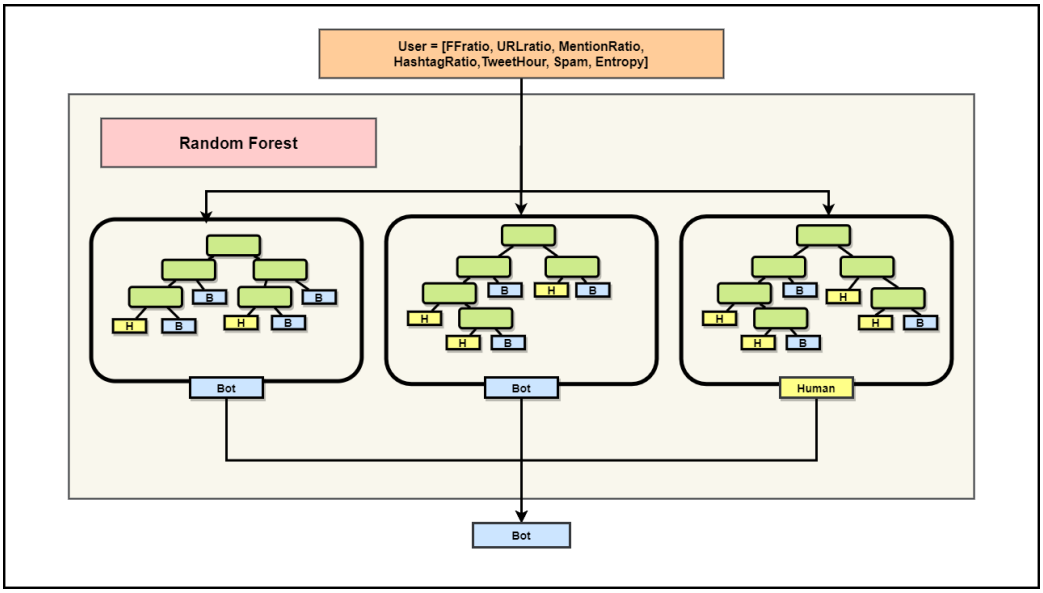

Figure 3.4: Random Forest selecting a final decision

4 | Tools and Implementation

In this chapter the different libraries used for the development of the project will be explained, as well as a short explanation of the most notable functions and parameters that have been used. We understand that the validity of the algorithms and mathematical formulas of these functions have gone through the verification methods of their own community

4.1 Twitter API

Twitter API [Twib] is a tool provided by Twitter itself, so that various companies and individual users can develop their own programs capable of searching for tweets with certain words and sending tweets to specific users using HTTP request. It needs an account to get an authentication key to connect to the service, and the free version limits the number of request and only offers the most basic methods.

Tweepy library: Tweepy [Roe] is one of libraries that offers access to Twitter API for the Python programming language. The library has authentication and all RESTful methods of the Twitter

API. Thanks to this library we can make a crawler that can collect the profiles and messages of the users.

4.2 MongoDB

MongoDB [Mon] is a NoQSL database, which will allow us to save the information of the user's profile and their messages in JSON format and to access them easily from external tools thanks to the libraries in multiple programming languages.

Pymongo library: PyMongo [DH] offers the tools to work with MongoDB from Python such as creating, reading, writing, updating an collection in MongoDB.

4.3 Python

Python [VD09] is a programming language well known for its easy to understand syntax reducing the entry barrier for new programmers, making it the favorite code of beginners and educational centers, also with its efficiency it continues to attract veteran programmers.

Thanks to its great popularity, many developers and companies have created a large number of open source libraries compatible with their own programs or service, as well as scientific and mathematical libraries with ready-to-use algorithms.

Pandas library: Open source library that offers tools for data analysis and manipulation.

Numpy library: Is another open source tool, specialized in working with numerical multidimensional array and matrix data structures.

Json_util.Bson library: Because MongoDB stores the data in its own format, Bson that are similar to Json, we use this library to transform from one to the other.

Datetime library: Library to work with date formats.

Sklearn library: Free machine learning library, which offers operations which work together with other libraries such as matplotlib, numpy, pandas or scipy.

4.4 Implementation

In this section, some examples of the main lines of the code are show. This lines contain function from libraries described in the previous chapter, which have facilitated the creation of the classification program.

4.4.1 User and Tweet Grabber

User Grabber

```
user_tweepy = api.get_user(user[0])
db.users.insert(user_tweepy_json)
```

Tweepy API searches an user using it's id, user_id or screen_name,

returning a User JSON with all the information of the users profile such as name, screen name, date of account creation, number of followers, number of friends, favorite account, etc and it is stored in MongoDB.

Tweet Grabber

```
    for tweet in tweepy.Cursor(api.user_timeline, \
        wait_on_rate_limit = True, id=user['id']):
```

This line from the Tweepy library gets all the tweets, call timeline, from the user with a determinate id. The wait_on_rate_limit parameter makes the code wait when the Twitter's limit of tweet per minute has been reach, until Twitter allows tweets to be picked up again.

```
    db.tweets.update_one({'id_user': id_user,'id_tweet': \
        id_tweet},{"$set": tweet._json}, upsert=True)
```

This line from PyMongo stores the tweet tweet._json,using the keys id_user and id_tweet. *Upsert* makes this whenever it does not exist any matching element in the collection.

4.4.2 Spam/Ham Classifier

```

    test_df = pd.DataFrame(tweets_final, columns =['label', 'message'])
```

We use a Pandas data frame to store the information and work with it.

```
test_bow = bow_transformer.transform(test_df['message'])
test_tfidf = tfidf_transformer.transform(test_bow)
```

We transform the test data from strings to a numerical vector list to operate with them easily.

```
# We apply Naive Bayes to classify each tweet
spam_detect_model = MultinomialNB().fit(train_tfidf,train_df['label'])
result = spam_detect_model.predict(test_tfidf).tolist()
```

We train a Multinomial Naive Bayes from the Sklearn library, and we predict the labels from the test data.

4.4.3 Random forest

```
# Instantiate model with 1000 decision trees
rf = RandomForestClassifier(n_estimators = 1000, random_state = 20)
```

Creates a forest with a 1000 trees, random_state is a seed used for random generation so me could get the same results in different executions.

```
# Train the model on Train data
rf.fit(train_features, train_labels)
```

With the *fit* function we train the 1000 trees.

```
# Get the predictions from the Test data
predictions = rf.predict(test_features)
```

We predict the labels from the test data with the rf random forest

5 | Results

In this chapter, we will present the results obtained from the training data set and the results on the real data base.

5.1 Training database

To carry out the tests to check if our classification algorithm works well, it was necessary to test it against a group of users that contained bots and humans. Taking our purpose into account, we took two different datasets from the repository of the page https://botometer.iuni.iu.edu. The firts one, named "midterm-2018" is a collection of bots and humans collected during the 2018 US midterm elections. It fits perfectly given that our final objective is to apply this on a database with users from a political environment. In addition, a second general dataset was taken, named "gilani-2017" that combines bots and humans in a more general context.

These databases contain only the id of users, so a small script was created that would search if these users accounts were not deleted or protected, which would prevent us from accessing their tweets.

Once users are verified to be available, we proceed to acquire their tweets.

We make the relevant calculations to be able to extract the features such as entropy, the spam rate, and the rest of the features of the profile and they are stored again in a collection of results in the database.

The first tests with the two previous data set, achive a precision of 73,22%, this result was too low compared to previous work. Changing the sizes of the train and test sets increased the accuracy up to 77.25% and cleaning the "gilani-2017" and "midterm-2018" sets from any account that used any different language than English gave us an accuracy of 89.65%, which is closer to the expected results, around 80-90% [Rod+20].

During these tests, we also looked at the importance of the most decisive properties during the creation of Random Forest, getting the results thanks to functions of the Sklearn library itself. Surprisingly as we can see in the Figure 5.1. the most useful quality was the percentage of spam in the users' tweets, which were the most weight features in the classification process in the Random Forest.

```
Variable: spam                  Importance: 0.46
Variable: urlRatio              Importance: 0.17
Variable: mentionsRatio         Importance: 0.11
Variable: entropy               Importance: 0.07
Variable: hashtagRatio          Importance: 0.07
Variable: MaxTweetsHour         Importance: 0.06
Variable: friendFollowers       Importance: 0.06
Variable: verified              Importance: 0.01
Variable: created               Importance: 0.0
Variable: devices               Importance: 0.0
```

Figure 5.1: Importance of feactures

5.2 Use on real Users

The users for the real test have been chosen from the data set used by Marc Solé in his project "Real Time Classification of Political Tendency of Twitter Spanish Users based on Sentiment Analysis" [Sol+18]. Due to the lack of enough time, only a small fragment of his database was used for the experiment. We proceeded to get the profile information and tweets from approximately 7000 users. Around 7% of them corresponded to accounts which were blocked by Twitter, so they were discarded from the investigation since it was not possible to get all the necessary features.

The obtained results applying our algorithms show that 1.8% of the users were bots. This result seemed too low considering that some statements tell us that approximately 8.5% of Twitter users are considered bots and in political environments it rises to 19% [Cho]. These low results may be due to the fact that the training

dataset was very different compared to the real data or the appearance of specific cases that escape the classification algorithm.

5.3 Analysis of final results

Our sorter has given strange results. So to find out what could have gone wrong, we decided to manually analyze a small sample of bots and humans in the results obtained. Most of these were pages from the offices of various parties in different cities in Spain and party members while a few other bots shared human-like behavior, which makes us think they were misclassified.

Then some of the users classified as human were also manually analyzed. The first alert was the names of users, finding proper names followed by long chains of numbers, and who shared the same behavior of repeating messages from party or political accounts. Another alert was to find a user who was clearly spamming the same message to several users just by changing the name of the recipient. This behavior was not detected by the classifier, because it checked whether the message was spam compared to the training dataset and not to the user himself.

Because retweets were the problem of misclassification, a new feature was implemented, the rate of retweets to see if users were using widespread tweets that could be fake news or retweets from their candidates. The change increased the percentage of bots to 9.16%, but such a change did not fix the problems of miss classification of dubious names or other techniques used by bots. This increase in bots, were mostly those users who only engaged in retweeting, a behavior expected from Fake Users. But retweets

apart from politics were also found, which could lead us thing that the user was a Lurker, a human who shares a similar behaviour with the False User.

We can correctly classify quite a few bots, but we cannot dismiss that there are also users who give false positives. We can also perceive a weakness in our algorithm as it mainly target those accounts that are dedicated to retweeting, while other types of bots escape the classification.

A more thorough examination with appropriate tools and algorithms would be needed, but during manual examination of a small sample of 200 users of these 6000 accounts, 2 main political orientations were detected. Among the accounts classified as bots and human, a large majority were repeating ideas in favor of the right-wing Vox party. To a lesser extent the PSOE also received support, mainly from its own accounts such as members or office, and briefly from the rest of departments.

6 | Conclusions

The project is far from satisfactory as it cannot reliably distinguish between bots and humans. Manual verification has shown that a large number of users are misclassified.

6.1 Setbacks

This work suffered several setbacks on the technical side. The number of users collected by Marc's database was almost 2 million and the tweets were 37 million, the size of the data exceeded the computational capabilities of our equipment and time for the project. This led us to look for methods that would improve the speed of the classifier like improving the searching methods to go directly to the tweets need or the reduction of the sample. The final solution chosen was the later, to reduce considerably the size of the sample of users.

6.2 Improvements

Random Forest responded well to the feature of Spam and this was implemented in its most basic version, we could expand these with techniques that standardize the words or expressions of users. Also extend the range of spam techniques that the algorithm is able to detect, especially see if the user repeats its own tweets repeatedly.

We detect great disparity on the values of features in the training data set and the test data set. Therefore, a new training data set would be need to avoid that our scheme could be over-trained with the old training data sets.

Finally, it would be interesting to implement an specific technique to differentiate Lurkers from False Users, given that these can have a very similar behaviour, since these are the ones that give more false positive results.

6.3 Next steps

As a goal for a future project, once all the improvements have been implemented, it is to apply these results in order to know if these bots have any kind of influence on the voting intention or can interfere in algorithms to predict voting intention.

Bibliography

[AS11] Wael Awad and Sherif-Elseuofi. "Machine Learning Methods for Spam E-Mail Classification". In: *International Journal of Computer Science & Information Technology* 3.1 (2011). DOI: https://doi.org/10.5121/ijcsit.2011.3112.

[Bhu+18] Hanif Bhuiyan et al. "A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques". In: *Global Journal of Computer Science and Technology* 18.2 (2018). URL: https://computerresearch.org/index.php/computer/article/view/1705.

[Bre01] Leo Breiman. "Random Forest". In: *Machine Learning* 4.45 (2001), pp. 5–32. DOI: https://doi.org/10.1023/A:1010933404324.

[Cho] Zoey Chong. *Twitter Developer Platform*. URL: https://www.cnet.com/news/new-study-says-almost-15-percent-of-twitter-accounts-are-bots/.

[Chu+12] Zi Chu et al. "Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?" In: *IEEE Transactions on Dependable and Secure Computing* 9.6 (2012), pp. 811–824. DOI: https://doi.org/10.1109/TDSC.2012.75.

[Dav+] Clayton Davis et al. *Online Human-Bot Interactions: Detection, Estimation, and Characterization*. URL: https://arxiv.org/abs/1703.03107. March 27, 2017.

[DH] Mike Dirolf and Bernie Hackett. *Python driver for MongoDB*. URL: https://github.com/mongodb/mongo-python-driver.

[DM19] Alfonso Delgado-Bonal and Alexander Marshak. "Approximate Entropy and Sample Entropy: A Comprehensive Tutorial". In: *Entropy* 21.6 (2019), p. 541. DOI: https://doi.org/10.3390/e21060541.

[Gur+16] Supraja Gurajala et al. "Profile characteristics of fake Twitter accounts". In: *Big Data & Society* 3.2 (2016). DOI: https://doi.org/10.1177/2053951716674236.

[Kem] Simon Kemp. *Digital 2020: Global Digital Overview*. URL: https://datareportal.com/reports/digital-2020-global-digital-overview. January 30, 2020.

[Mon] Inc. MongoDB. *MongoDB, The database for modern application*. URL: https://github.com/mongodb/mongo.

[Pas+20] Javier Pastor-Galindo et al. "Spotting political social bots in Twitter: A use case of the 2019 Spanish general election". In: *Data in Brief* 32.106047 (2020). DOI: https://doi.org/10.1016/j.dib.2020.106047.

[Rod+20] Jorge Rodríguez-Ruiz et al. "A one-class classification approach for bot detection on Twitter". In: *Computers & Security* 91 (2020), p. 101715. DOI: https://doi.org/10.1016/j.cose.2020.101715.

[Roe] Joshua Roesslein. *Tweepy: Twitter for Python!* URL: https://github.com/tweepy/tweepy.

[Sol+18] Marc Solé et al. "Real Time Classification of Political Tendency of Twitter Spanish Users based on Sentiment Analysis". In: *International Journal of Computer and Information Engineering* 12.9 (2018), pp. 697–706. DOI: https://doi.org/10.5281/zenodo.1474549.

[SS18] Surendra Sedhai and Aixin Sun. "Semi-Supervised Spam Detection in Twitter Stream". In: *IEEE Transactions on Computational Social Systems* 5.1 (2018), pp. 169–175. DOI: https://doi.org/10.1109/TCSS.2017.2773581.

[Sub+16] V.S. Subrahmanian et al. "The DARPA Twitter Bot Challenge". In: *Computer* 49.6 (2016), pp. 38–46. DOI: http://doi.org/10.1109/MC.2016.183.

[Twia] Inc. Twitter. *Edgett Appendix to Responses: Update on Results of Retrospective Review of Russian-Related Election Activity*. URL: `https://www.judiciary.senate.gov/download/edgett-appendix-to-responses`. January 19, 2019.

[Twib] Inc. Twitter. *Twitter Developer Platform*. URL: `https://developer.twitter.com/en/docs/twitter-api`.

[VD09] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, 2009. ISBN: 1441412697.

A | Appendix

Code found in:

`https://github.com/KevinGarMill/Basic-twitter-classifier`