Detection of Suspicious Users Posting Claims about Cancer on Twitter

BY

MASSIMO PIRAS

B.S, Politecnico di Torino, Turin, Italy, 2016

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2018

Chicago, Illinois

Defense Committee:

Bing Liu, Chair and Advisor
Barbara Di Eugenio
Elena Baralis, Politecnico di Torino

To Dalia, always in our thoughts, forever in our hearts.

May you watch over us all.

ACKNOWLEDGMENTS

I want to thank all the people that made this work possible. First, I would like to express my gratitude to my advisor, professor Bing Liu, for giving me the opportunity of working with him and for having followed and advised me in every step of this journey. His guidance was essential to the success of this work. I want to thank professor Elena Baralis from my home university, Politecnico di Torino, for her support. A thank you to professor Barbara Di Eugenio for being interested in my research and accepting to become part of my committee. I want also to thank all my friends and colleagues I shared this experience with, who helped make it wonderful and unforgettable. Finally, I am very grateful to my family for their unconditioned love and support.

MP

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
CDF	Cumulative Density Function
CM	Confusion Matrix
HDFS	Hadoop Distributed File System
HTTP	HyperText Transfer Protocol
LDA	Latent Dirichlet Allocation
NLTK	Natural Language Toolkit
POS	Part Of Speech
RBF	Radial Basis Function
RDD	Resilient Distributed Dataset
SVM	Support Vector Machines
TF	Term Frequency
TF-IDF	Term Frequency Inverse Document Frequency
UIC	University of Illinois at Chicago
URL	Uniform Resource Locator

SUMMARY

Due to the massive success of social media, online user-generated content has increased exponentially in the last years. Twitter, as a microblogging platform, allows users to share information about their opinions or activities by means of short posts called tweets. However, opinion spammers see social networks like Twitter as an opportunity to propagate their ideas, promoting or discrediting some target product or service, without showing their true intentions.

In this study, we focused on detecting suspicious users who posted dubious claims about cancer treatment and prevention on Twitter. We addressed the task with a supervised learning approach, a binary classification problem in which we had to predict whether users were suspicious or genuine. We collected a set of 60 thousand tweets related to cancer posted in October 2017, including more than 36 thousand users. Since manual labeling could be a very complicated process, we elaborated a set of features for each user, both related to the content of her posts and her behavior on Twitter, and combined them to compute a spam score. The basic idea was that suspicious users would have different feature distributions with respect to genuine users and that would help us to separate the two classes. Then, we generated a ranking using the spam score and exploited it to assign the labels.

Finally, we ran a few classifiers on our labeled data, showing that suspicious users had different textual and behavioral patterns which could be used to distinguish them from genuine ones.

CHAPTER 1

INTRODUCTION

Social Networks are one of the most popular and widely used applications of Web 2.0 [1]. Among them, Twitter offers a microblogging service which can count more than 330 million active users per month. Through this platform, users can form social bonds and post updates about their life experience and their opinions, communicating with friends exchanging short messages. Twitter has become a system for obtaining real-time information since, as soon as users post some content, their followers are immediately notified, allowing information to be propagated incredibly fast. Unfortunately, opinion spammers can use Twitter as a vehicle to spread misinformation, posting malicious links and deceptive messages which can influence people's opinion on some subjects for their personal gain. They usually target trending topics, the most tweeted subjects in a short time window. Cancer cure and prevention is definitely a trendy topic: people could be induced into believing anything if that means giving them hope and finding a solution to such a delicate matter. In fact, when addressing cancer, it is easier for malicious users to get people to trust their opinions. Hence, it is necessary to detect suspicious users to ensure people that social media can be a trustworthy source of information.

In our research, we wanted to detect those users who posted some dubious and odd claims about cancer treatment and prevention. Since we did not have either authority or the competence to determine whether some claims were utterly wrong, we just defined the users with ambiguous conduct on Twitter as suspicious. The problem of distinguishing genuine users from

suspicious ones can be seen as a binary classification problem, where the majority class is represented by normal users, and the minority class is associated with suspicious users. We used Twitter APIs [2] to download a set of 60 thousand tweets related to cancer treatment and prevention, posted by a total of 36 thousand users in October 2017.

In order to perform supervised learning, we needed a labeled dataset, so we had to assign a label to each user. Manual labeling is a very complicated task, and we did not have either the workforce or the time to do it from scratch. Therefore, we extracted a set of features for each user, we combined them to compute a spam score, and then we generated a ranking. This score represented the likelihood of a user being suspicious. We exploited our ranking to complete the labeling task, making it easier: we performed a process composed of multiple steps, which started analyzing the top users in the ranking, that allowed us to assign the labels without having to go through every single user in our dataset.

Then, we tried different machine learning algorithms for classification to separate the two classes of users and evaluated their performance. We used ten-fold cross-validation to divide our dataset into multiple training sets and testing sets. We had three different sets of data: feature dataset, text dataset, and combined dataset. In the feature dataset, we had all the computed behavioral features for each user, and we tried to exploit them to predict the class label, convinced that suspicious users would have different and peculiar feature distributions. For the text dataset, we just used the content of the tweets posted by users to predict their labels. In the third set, we combined both behavioral and linguistic features to perform classification.

Lastly, we investigated how suspicious users could collude, forming groups to have a more

significant impact on people's opinions on cancer prevention and treatment. By analyzing similar content and relationships between suspicious users, we found out that some of them could be working together to achieve a common goal.

1.1 Outline

Our work is going to be divided into several chapters.

Chapter 2 gives an outline of the related work on fake review detection on commercial platforms and opinion spam detection in social media. Chapter 3 describes Twitter as a microblogging platform, highlighting its essential features as a vehicle for spreading information. Chapter 4 illustrates how we collected our dataset and which preprocessing steps we performed. Chapter 5 explains how we performed topic modeling and extracted different topics from our tweets. Chapter 6 deals with the computation of the behavioral features and the generation of the ranking. Chapter 7 describes in detail our labeling process, how we assigned labels to users and why. Chapter 8 illustrates the machine learning algorithms we used for classification and also explains how we tuned the parameters. Chapter 9 describes the experimental results obtained by running our classifiers. Chapter 10 deals with the detection of groups of suspicious users who could be working together. Finally, chapter 11 summarizes our work and suggests future directions for it.

CHAPTER 2

PREVIOUS WORK

Spam detection has been studied in many different fields, and the most studied types of spam are probably email and web spam. However, due to the massive success of social media, opinion spam has become more and more critical.

Since we were focusing on a specific topic, cancer prevention and treatment, and because many products, drugs, and foods were claimed to be effective against cancer, our problem was somehow related to fake review detection. Indeed, in our work we also exploited some features which were used to identify fake reviewers. Similarly to opinion spamming on social networks like Twitter, fake reviews may promote or damage some entities, influencing people's opinions and also damaging businesses.

Moreover, as for Twitter accounts, fake reviewers may be professional fake reviewers, who are easier to find since they follow some recognizable behavioral pattern, and nonprofessional fake reviewers, who mainly write to help themselves. Fake reviewers also collude, creating groups of spammers who cooperate to promote or discredit target entities [3]. Since we could find many similarities between fake reviews and opinion spamming in social media, we considered previous work on detecting fake reviews as related to our study. The only significant difference was that on Twitter we could not analyze and exploit rating behavior for users.

Furthermore, many studies have been done on spamming in social networks, including Twitter. In many cases, the researchers wanted to find campaigns and groups of spammers

colluding to spread their information for a common hidden purpose. Because of the absence of a gold-standard dataset for fake opinions and the dual behavior of some users, this is a very complicated task, and no one has found an automated spamming detection method yet. To the best of our knowledge, this is the first study of opinion spamming about cancer in social media. Here we give a survey of the previous work on detection of fake reviews and opinion spamming in social media.

In [4], A.Mukherjee et al. found a way to detect opinion spammers on review platforms with an unsupervised model, called Author Spamicity Model. Expecting spammers to have different behavioral patterns than genuine reviewers, they exploited behavioral footprints of users to divide them into two clusters: spammers and non-spammers. This method was innovative because it did not need any manually labeled data, but used an unsupervised Bayesian framework to characterize users and create a separation margin between population distributions of the two groups. We exploited some of the behavioral features they extracted, such as content similarity for reviews, maximum number of reviews, burstiness for reviews and number of near-duplicates reviews. Most importantly, we got the idea of computing a spam score for users to help us identify and label them. Indeed, we created a ranking based on the computed score, and we fully exploited it to reduce the complexity of our labeling process.

In [5], F. Benevenuto et al. performed a classification task on a large manually labeled Twitter dataset using textual and behavioral features of users. They conducted a study about characteristics of tweets content and users behavior to detect spammers. In our research, we used many of their features: number of URLs, hashtags and mentions, number of followers and

friends, reputation, i.e., number of followers per friend, the age of the account and many more. Moreover, they noticed that their classifier misclassified about 30% of non-spammers: this was due to the fact that many spammers showed a dual behavior, posting genuine tweets most of the times and just a few ones considered as spam. Also in our study, some classifiers could not accurately identify genuine accounts.

In [1], A. Wang tried to automatically detect spammers by using a graph model which could adequately describe the non-reciprocal following relationships on Twitter. He modeled the reputation as a way to measure the importance of a user on the platform: it was computed as the ratio between the number of followers and the sum of followers and friends. As a matter of fact, spam accounts try to follow as many users as they can to spread their information, while prominent accounts are likely to have a considerable number of followers and fewer friends. In addition to graph-based features, behavioral features were used to perform classification, distinguishing spammers from non-spammers. They mostly used content-based features like the number of duplicate tweets, and the number of HTTP links, hashtags, and mentions posted.

In [6], C. Grier et al. conducted a study on spammers behavior, clickthrough and on the effectiveness of blacklists to prevent spam propagation on Twitter. They analyzed the content of a large dataset of tweets, which contained over 2 billion URLs pointing to scams and phishing pages and malware. To better understand and detect spammers, they also studied their behavior, checking the number of mentions, hashtags, retweets posted by each user. They described a particular activity, which we also found in our data, called Tweet hijacking: spammers tend to retweet posts from authoritative users to be trusted, but then they modify the content to

suit their intentions better. Then, they observed the clickthrough on spam links, showing that the most-clicked malicious URLs were phishing ones. Finally, they analyzed the effectiveness of Twitter blacklists, proving that they were too slow to detect spam links correctly. Moreover, they found out that shortened and obfuscated URLs were immune to the blacklists. Indeed, shortening techniques are used by spammers to evade detection.

In [7], G. Fei et al. exploited the burstiness nature of reviews to identify spammers. Bursts in reviews are usually generated by either increased popularity of the product or spam attacks. They stated that malicious users usually work with other malicious users, while genuine users appear in the same burst with other genuine users. Hence, they represented reviewers and their relationships in a graph and then tried to link reviewers in the same burst. They proposed a burst detection method based on kernel density estimation, and then they distinguished between spam and genuine bursts. To better identify spammers, they also used a set of behavioral features such as review burstiness and content similarity. Finally, they evaluated their detection algorithm performing a supervised text classification, showing that reviews written by spammers were different from the ones written by genuine users.

In [8], H. Li et al. studied an extensive dataset with more than 6 million reviews of all restaurants in Shanghai, China, provided by Dianping. Differently from other works, they performed spam detection exploiting spatial and temporal patterns. They observed how spammers and non-spammers were distributed in China, showing that IP addresses that were most distant from Shanghai were mostly malicious. In our work, we also tried to exploit the registered location for each user, but it was generally unavailable. Professional spammers often register

multiple accounts with different userids to avoid detection and to propagate their pieces of information better. Since spammers may register many accounts in a short time period, they also supposed that they could often change IP address while publishing reviews to evade spam filters. Hence, they defined the Average Travel Speed, considering the location of the IP address and the timestamp of the review. Finally, they also exploited behavioral features to perform classification.

In [9], N. Jindal et B. Liu proposed a solution to the labeling problem in supervised learning for opinion spam in reviews. Since manual labeling is very hard and requires much time, they found a way to assign labels to reviews automatically. They supposed that spammers were most likely to write duplicate or near-duplicate reviews because writing new, different ones would require too much effort and would not allow them to propagate their ideas quickly enough. Hence, they found all the duplicate and near-duplicate reviews and labeled them as spam reviews. Then, they built a model to perform classification, based on a set of review, user, and product features. Even if posts on Twitter and reviews on commercial platforms are not the same, we also considered the similarity between tweets as a significant feature: it is undoubtedly effective to re-post the same content over and over again to reach out to many users and propagate ideas. However, we found that in our dataset many accounts posting a lot of duplicate or near-duplicate tweets were genuine and they were just trying to spread useful information. In most of the cases, they were authoritative and prominent users, such as journals and universities.

In [10], A. Mukherjee et al. studied Yelp spam detection algorithm trying to understand

how it works. Differently from other studies, they had a real-life labeled dataset: Yelp's filtered and unfiltered reviews. They tried a supervised learning approach using both linguistic and behavioral features. Indeed, they observed that linguistic features, such as n-grams and POS, were not useful for real-life reviews leading to a maximum accuracy of 68%. However, this result proved that there was a difference between filtered and unfiltered reviews on Yelp. On the other hand, behavioral features, as content similarity, reviews length and number of posted reviews, rendered a higher accuracy of 86%. Therefore, they concluded that Yelp filter, which is known to be very good, might be using a behavioral-based approach. It is also likely that it combines both behavioral and linguistic features, as we did in our study. Since they reported that linguistic features did not perform well, we just exploited unigrams and bigrams as textual features, without using POS.

In [11], A. Mukherjee et al. tried a supervised approach for fake reviews detection using pseudo-fake reviews generated using Amazon Mechanical Turk crowdsourcing tool. Due to the lack of gold-standard fake reviews data, they used those pseudo-reviews to obtain a labeled dataset. However, they were different with respect to real fake reviews from commercial sites, since the authors did not have the same state of mind of professional spammers. They built a model based on linguistic features and then ran a classifier reaching an accuracy of almost 90%. With respect to Yelp's data, which yielded 68% accuracy, the performance for this model was much better, showing that fake review detection on real-life data was more complicated than detection using AMT pseudo fake reviews. This was mostly due to the fact that AMT authors used a different words distribution with respect to genuine users, making it easier to

separate fake and genuine reviews. On the other hand, Yelp fake reviewers had a more similar words distribution to nonfake reviewers. This means that either Turkers did not do a good job in creating pseudo fake reviews or that Yelp users were very good at faking. Finally, they proposed a set of behavioral features that could help distinguish between genuine and malicious users, as activity window, review count, review length and content similarity.

Group spamming has much more impact than single accounts, because a group of users working together may post a lot more reviews and messages, propagating their ideas more effectively. As a matter of fact, a single user might not be sufficient to influence people's opinions on a subject. Moreover, while a single user is easy to detect for abnormal behavior, in a group of colluding users members do not appear as behaving suspiciously. Groups of malicious users are very dangerous and they represent an issue both for reviews on commercial websites and posts on social media.

In [12], A. Mukherjee et al. proposed a technique to identify spammer groups, based on frequent itemset mining. Each itemset was represented by a set of users who reviewed the same set of products. Once they extracted the candidate set of groups of users, they computed some spam indicator values to identify malicious groups. These behavioral features included time windows, content similarity for members, group sizes and early time frames. Then, they ranked all the groups according to the likelihood of them being spam groups and used a set of labeled groups to evaluate their model, showing promising results.

In [13], X. Zhang et al. focused on detecting promoting and spamming campaigns on Twitter. To identify users in a campaign, they used an URL-driven estimation method to measure

the similarity between users and then they used a graph-based approach to find users that were linked to a candidate campaign. Finally, they exploited a set of behavioral features to distinguish between promoting and spam campaigns. Among these features, the most relevant ones were average posting intervals, content similarity for messages, number of URLs posted, and average number of URLs per tweet for the account. Then, they built a classifier that correctly predicted the class of candidate campaigns, showing the effectiveness of their technique.

In [14], Z. Chu et al. used a supervised approach based on both content and behavioral features to distinguish between spam campaigns and legitimate ones. They defined a spam campaign as a collection of multiple accounts manipulated by a spammer, used to spread information on Twitter with malicious intent. First, they grouped users into clusters which represented campaigns, by relating the ones sharing the same URLs. Then, they computed a set of features which would be used during classification to learn a model to distinguish between spam and genuine campaigns correctly. They also analyzed the text of the tweets, checking whether they contained spam words or blacklisted URLs, and the accounts characteristics, such as the number of followers and friends, reputation, number of hashtags and mentions, registration date, self-similarity score and posting time intervals. Finally, they ran different classifiers which could accurately predict class labels.

In [15], K. Lee et al. investigated the problem of campaigns detection in social media. They proposed a content-driven graph-based framework for identifying and extracting campaigns from Twitter. They defined a campaign as a set of users and posts linked together by a common goal. Hence, they linked messages with similar content, building a message graph,

where each node corresponded to a message, and each edge connected two correlated messages. They extracted different campaign graphs from it via graph mining techniques. Since they used a dataset of labeled campaigns, they could evaluate the performance of their extraction method, presenting promising results. Therefore, they showed that campaigns could be detected using automated techniques.

For group, we mean a set of multiple userids. However, a group does not have to be composed of different users, but of different accounts. There could be a single person that registers multiple userids and posts from many accounts to increase her impact and to avoid detection. Since an author with multiple accounts can be regarded as a group, identifying users with multiple userids is a very similar task to finding spamming groups.

In [16], Quian and Liu investigated the problem of detecting users who use multiple userids to post on social media. They used a particular supervised learning approach that learned in similarity space, where each document was represented by a similarity vector. Indeed, to assign a document to a user, the classifier exploited documents belonging to other users instead of documents from the same author: the method could determine whether a document was written by an author without using any of her documents in training.

CHAPTER 3

WHY TWITTER?

In the last years, Social Networks have made their way into our lives, allowing anyone to follow the lives of her friends and relatives and to communicate with them, despite the distance. The usage of these platforms has grown more and more, such that nowadays, they represent a critical and powerful tool. As a matter of fact, due to the incredibly large number of users, they can carry a considerable amount of information, especially from a commercial and political point of view: users share their opinions and feelings about events, products or their personal experience. Companies and political parties may collect and exploit this data for their own business.

Among these social networks, Twitter is an "online news and social networking service on which users post and interact with messages known as tweets", which can count more than 330 million monthly active users. It offers a microblogging service, which may be accessed through its website interface, through Short Message Service (SMS) or mobile-device application software ("app") [17]. Every message, or tweet, is 140 characters long[1]. This constraint sets up an environment of concise statements that leads to relatively easy posting and reading, as people do not need to put much effort in either thinking about what they write thoroughly nor reading tweets posted by others. Since less time is needed to post content, the frequency of the

[1]Twitter extended the maximum tweet length to 280 [18]

updates is increased as well. Updates may be either direct or indirect: directs updates aim to specific users, mentioned by the '@' mechanism, while indirect updates are meant to be read by anyone.

Unlike most social networks, the relationship between users is not reciprocal. Following a user implies receiving notification of each tweet posted by the followed user, but the user being followed does not need to follow back [19]. This peculiar relationship makes the Twitter social network a directed graph, in which users represent the vertices and the following relationship is a directed link from the follower to the followed. However, as shown in [20], this graph has a very high degree of correlation and reciprocity, which highlights close mutual acquaintances among users. Furthermore, people interact with a subset of the users in their network. If we consider only this subset, which we can call friends, we can notice that users have a number of friends which is way smaller with respect to the number of followers and followees they declare [21]. Therefore, the link between two users does not automatically imply an interaction between them.

As a micro-blogging platform, Twitter allows people to form social bonds and to propagate their ideas, writing brief text updates and sharing information about their opinions or activities. Since it is a powerful tool for spreading ideas, it is essential to understand why people use it and which are their intentions. [19] presents a taxonomy of user intentions on Twitter:

- **Daily chatter**: most people use twitter just to share their activities and update their friends on what is going on with their lives. This class represents the majority of Twitter users.

- **Conversations**: since there is no direct way for users to chat, the only way they can directly interact is by exploiting the mentioning mechanism, with '@' followed by the username. 14% of the posts in the gathered collection contained mentions.
- **Sharing information**: Many posts in the collection contained URLs, almost 67%. Due to the 140 characters limitation, it is difficult for users to express their ideas fully. Therefore, URLs are often used to redirect the reader to a web page where the whole concept is fully described. Moreover, URLs are shortened in order to make the posts fit in the 140 characters. This was the most interesting category of users for this study because we found many people spreading misinformation among them.
- **Reporting news**: Many users post tweets to report the latest news or to share their opinion about current events or trending topics on Twitter. These are interesting as well: they could report fake news or fake events and discoveries for many different reasons.

Since the usage of Twitter and the generated content is increasing, alongside with the fact that a huge amount of information is shared, all this produced data may be used for opinion mining and sentiment analysis [22]. Indeed, due to the heterogeneity of the users, it is possible to collect tweets from users of different social and interests groups, coming from many different countries. Companies may analyze all this data and extract information for a better business decision making. They can collect opinions on products, suggestions on how they should modify them and even which new features they should develop, according to what the audience demands. Also, politics may benefit from such a massive amount of data: we just have to think about the U.S. Presidential Elections in 2012, which were surrounded by huge online traffic of breaking

news, statements, debates and voters' opinions. Back in 2012, Twitter reported over 100 million active users worldwide with a posting rate of 250 million tweets per day; more than two-thirds of U.S. Congress members had created a Twitter account, trying to reach out to their constituents [23].

Because Twitter is such a powerful platform for spreading information, the content of the posts is relevant for many tasks, such as news detection, sentiment analysis, and recommendation systems. Moreover, it has become a widely used system for obtaining real-time information. Therefore, it is essential to distinguish between good and malicious sources, since users have to be ensured that social media like Twitter can be a trustworthy source of information.

CHAPTER 4

DATA COLLECTION AND PROCESSING

4.1 Collecting Twitter Data

To perform and evaluate spamming detection techniques, we needed a real dataset of tweets. By means of the Twitter streaming APIs [2], using the function `filter`, we collected more than 60 thousand tweets posted in October 2017, containing at least one of the following set of keywords:

- **cancer cause**
- **cancer drug**
- **cancer food**
- **cancer prevention**
- **cancer risk**
- **cancer treatment**

As a matter of fact, we were interested in all the statuses which somehow referred to means and techniques to either prevent or cure cancer, and in those tweets reporting causes for cancer. We were looking for claims about foods or drugs helping to prevent or treat cancer, and for statements on lifestyles and diets which were said either to increase cancer risk or to eradicate it. Therefore, exploiting those keywords helped us to collect a proper dataset for this study.

We chose to collect tweets from October because it is the Breast Cancer Awareness Month [24]. Hence, cancer was going be one of the trending topics and more people were likely to talk about it and propagate their ideas. For each tweet, we gathered a set of useful pieces of information. The ones we used included timestamps, username, number of retweets and likes, the id and, of course, the text of messages.

We collected more than 36 thousand users who posted at least one status update containing the above keywords, in the defined time period. To obtain additional useful information for every user, we once again used the Twitter APIs, in particular, the function `get_user`. This procedure allowed us to obtain the creation date, the total statuses number, the number of followers and followees, and the number of lists for each account. It is worth mentioning that by the time this operation was completed, almost 200 user accounts could not be found since they were suspended by Twitter [25] as considered spam accounts. Nevertheless, we were going to consider them in our study and analyze whether they posted fake claims about cancer or not.

4.2 Data Preprocessing

Twitter posts are basically pieces of informal text, which means that the quality of the collected data was not high. Data is usually incomplete, noisy and inconsistent. There could be misspelling errors, informal intensifiers as all-caps and character repetitions to increase the intensity of a word, special characters which do not add any significant meaning to the sentence [26]. Of course, to low-quality data corresponds low-quality data mining results. Therefore, we needed to execute some data preprocessing operations to increase the quality of our data.

We performed similar steps to what had been done in [27], but with a few differences because our final goal was not sentiment analysis, but spam detection. The following operations were performed:

- **Data transformation**: every capital letter was turned into lowercase, accents were removed and HTML tags parsed. Differently from what was done for sentiment analysis purposes, we kept URLs because they may not be good indicators for sentiments, but they surely have an important role in opinion spamming. We kept retweets as well because they could help find groups of users cooperating to spread information.
- **Tokenization**: tweets were split into tokens by a pre-trained Twitter model, which kept hashtags, emoticons, and other special symbols.
- **Normalization**: to reduce each word to its stem, we used the Snowball Stemmer algorithm [28]. Stemming is a common technique in text mining, mostly used to improve effectiveness and decrease indexing size, by combining words with the same root.
- **Filtering**: we had to remove all words and characters which were not useful for our task. Stopwords removal was performed to improve effectiveness and reduce indexing size. Also, punctuation digits and special characters were filtered, as well as tokens shorter than three digits and tokens containing non-ASCII characters.

Every post was processed using Orange [29], an open-source data visualization, machine learning and data mining toolkit based on Python that can also be used as a Python library. The

TABLE I: EXAMPLES OF DATA PREPROCESSING.

Raw Text	'INHALING just *ONE* radioactive hot particle can cause cancer http://'
Processed Text	'inhal one radioact hot particl caus cancer http://'
Raw Text	'Strawberries can improve vision and also help to reduce cancer risk'
Processed Text	'Strawberri improv vision also help reduc cancer risk'
Raw Text	'Get #Prostate #Cancer Prevention > http:// #IBOtoolbox'
Processed Text	'get #prostat #cancer prevent http:// #ibotoolbox'

Text Mining plugin [30], which is based on the NLTK [31] library, allowed us to obtain the preprocessed version of the text for each tweet.

In Table I we show some examples of anonymized preprocessed tweets.

4.2.1 Pooling Tweets by User

Since our goal was to determine whether a user had to be considered suspicious or not, we aggregated all the updates posted by the same user to obtain one single text document for each one of them. Due to the fact that this operation was very memory consuming and computationally expensive, we could not run the program on a single machine. Therefore, we used the Polito cluster [32] that allowed us to have access to multiple computers working in parallel, using the PySpark programming framework [33]. Apache Spark [34] is an open-source cluster-computing framework which provides the programmers with an interface that takes care of the distributed part of the applications, such as parallelism, fault-tolerance, task scheduling

and synchronization. It is a fast and general engine for large-scale data processing which grants low-latency and generality, based on the Hadoop Distributed File System. "The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware". It is highly fault-tolerant and provides high availability and high throughput access to application data [35]. Differently from other frameworks for big data like MapReduce, Spark allows the programmer to create and execute complex, iterative jobs, including multiple input-output operations on the same data. This is made possible because data is read only once from the HDFS and then split across many servers and stored in their main memory, granting an increase in performance as well. The data is stored and represented as Resilient Distributed Dataset, or RDD: "a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel" [36]. As their name suggests, they are particularly resistant to failures, since Spark keeps track of the chain of operations which led to the creation of each RDD, being able to recompute the lost data when needed. They are also immutable, meaning that once an RDD is created, it cannot be modified. RDDs may be created and manipulated by means of parallel operations: transformations, operations on RDDs which return new RDDs, and actions, that elaborate the content of RDDs and store the results in variables in memory.

To group all tweets by user, we first had to feed the cluster with the input file containing all the posts, which was then distributed across multiple servers. We stored the content into an RDD; then we applied some transformations to obtain new RDDs, concatenating all the tweets to form a single document for each user. Finally, using an action, we stored the resulting data structure and printed it.

CHAPTER 5

TOPIC MODELING

Once we obtained the preprocessed data, we wanted to analyze the most trending topics discussed by users in their posts and the most frequent and relevant words associated with them. The goal of topic models is to extract short descriptions of each document in a collection to enable efficient processing of such collection, while still preserving essential relationships useful for data mining tasks, such as classification and summarization [37].

5.1 Topic Modeling Techniques

A topic is a subject discussed in one or more documents, tweets in this case, and each topic is represented by distribution over words [38]. One of the most important weaknesses of topic models is that they need a large amount of data to generate coherent topics and provide reliable statistics [39]. Considering every post as a document, we had more than 60 thousand tweets, which was more than enough for the topic model to work and produce a good result. However, as proved in [40], tweets are usually too short and may not contain sufficient information to learn the topic patterns. Therefore, all the tweets posted by the same author were aggregated into pseudo-documents to increase the co-occurrences of important terms within a document. It was likely that a user posted updates about the same few topics, so aggregating all the posts into a single document could increase the co-occurrences of relevant words [41].

We ran topic models for two different datasets, distinguished by the way documents were defined, and then we compared the results:

- **"tweet" dataset**: composed of documents represented by single tweets. This collection contained every tweet we gathered, therefore more than 60 thousand documents.
- **"user" dataset**: composed of documents which contained all the posts of a single user. Basically, a document was generated by grouping all the tweets posted by the same user and concatenating them. Performing topic modeling on such data could be seen as an application of the author-topic model to tweets [42]. This dataset was obtained using the pooling preprocessing described in section 4.2.1. We had more than 36 thousand documents in this collection.

[43] showed that aggregating tweets by user improved the quality of LDA-based topic modeling. It also pointed out that pooling tweets by hashtag led to even better results. However, for the purpose of this study, we just performed the aggregation by user because they were the main target and, by grouping tweets by hashtag, we would not have got any additional information about the way they posted. We chose to use the Latent Dirichlet Allocation [37] algorithm to extract topics, since it is widely used and it is the one used in the related works on Twitter. LDA is a generative probabilistic model of a corpus that specifies a probabilistic procedure by which documents are generated. Like many other topic model algorithms, LDA is based on Bayesian networks and can be graphically represented.

A corpus is a collection of D documents, where each document d is represented by a vector of N_d words, where w_n represents n^{th} word of the sequence. Each word, instead, is represented

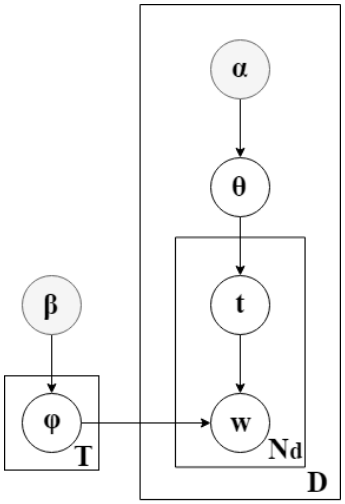

Figure 1: Graphical representation of LDA

by a unit-basis vector of size V, being V the number of different words in the vocabulary. The number of topics K is known and predefined.

It is a two-level model, where documents are characterized by a multinomial distribution θ over K topics and topics are characterized by a multinomial distribution Φ over words. Both distributions θ and Φ have a Dirichlet prior distribution [44], with hyper-parameters α and β. Dirichlet priors simplify the problem of statistical inference because the Dirichlet distribution is the conjugate prior of the multinomial distribution.

For each document, the following generative process is repeated N_d times, being N_d the number of words in document d: given a word w_n in a document d, a topic t is sampled from the multinomial distribution θ for document d and a word w is sampled from the distribution Φ for topic t. Learning the various distributions is a problem of Bayesian inference [45]. To

obtain the distributions θ and Φ, two main algorithms, variational inference [37] and Gibbs sampling [3] were proposed.

for *each topic* $t \in T_K$ **do**
  draw a word distribution for topic t, $\Phi_t \sim$ Dirichlet(β)
end
for *each document* $d \in D_M$ **do**
  draw a topic distribution for document d, $\theta_d \sim$ Dirichlet(α)
  for *term* w_i, $i \in W_{N_d}$ **do**
    draw a topic for word w, $t_i \sim$ Multinomial(θ_d)
    draw a word, $w_i \sim$ Multinomial(Φ_{t_i})
  end
end

Algorithm 1: LDA algorithm

where T_K is the set of topics of size K, D_M is the collection of documents of size M and W_{N_d} represents the collection of words for document d of size N_d.

We had to specify the parameter K, which corresponded to the number of topics we extracted with our algorithm. Usually, users of topic models prefer to extract a large number of topics in order to get high resolution and domain-specific fine-grained topics. Unfortunately, the number of topics is somehow related to the probability of having bad topics that make no sense to human judgment: increasing the total number of topics, the number of poor and small topics increases as well. They usually represent about 10% of the extracted topics, and they are often hidden from users because, otherwise, their confidence in topic models would be reduced [46].

In order to find the best number K of topics, we had to run the algorithm with different values for K. We also needed some evaluation measure which could tell us the quality of the extracted topics at each run. Traditionally, topic models quality was defined using predictive measures, such as perplexity [47]. However, models which achieve higher perplexity are often less interpretable and even without logical meaning when evaluated by human judgment [48]. Therefore, we needed another metric that corresponded well with human judgment.

As proposed in [46], we used Topic Coherence measure. Each topic t was represented by a list of the ten words with highest probabilities for t: this was a more adequate way to describe topics with respect to labels, which would have been meaningless in this case, leading to information loss. The basic idea is that, for good topics, pairs of words belonging to the same subject co-occur within a single document related to that subject. On the other hand, bad topics are likely to have a few words that co-occur, making them easily detectable.

The Topic Coherence for a given topic t is measured as follows:

$$C(t; W_M^{(t)}) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(w_m^{(t)}, w_l^{(t)}) + 1}{D(w_l^{(t)})} \tag{5.1}$$

where $W_M^{(t)}$ is the set of the M most probable words w for topic t. Given a topic t, we define $D(w_i^{(t)})$ as the document frequency for word w_i, i.e., the number of documents which contain w_i at least once, and $D(w_m^{(t)}, w_l^{(t)})$ is defined as document co-frequency, i.e., the number of documents containing w_m and w_l at least once. A smoothing count of 1 is added to avoid the

possibility of computing log(0). A higher value of Topic Coherence indicates a higher quality topic.

5.2 Topic Extraction and Evaluation

In order to perform topic extraction, we used the Orange data mining tool, in particular, the Text Mining extension [30]. Both the "tweet" and "user" datasets were fed to the process multiple times with different values of K.

The output of the program consisted of:

- **corpus file** which contained the probability distributions over topics for each single document
- **topic file** which contained the probability distributions over words for each topic

From those files, we obtained the most probable topic for each document and the set of most probable words for each topic.

Then, we computed the value of Topic Coherence for every topic in all the obtained models. We measured the document frequencies and the document co-frequencies for each topic and then obtained the values of coherence. Table II shows the average values of coherence for all the topics extracted from both datasets with different values for K.

We could confirm that the pooling operation helped to extract more coherent topics, since, for every selected number of topics K, the coherence was higher for the "user" dataset. Therefore, that dataset was used to determine which topic had to be associated with each user.

As expected, the larger K, the higher the number of poor topics with low coherence we found. However, this was balanced by the higher values obtained for good topics. As a matter

TABLE II: TOPIC COHERENCE AVERAGE VALUES.

10 Words	K = 10	K = 20	K = 30	K = 40	K = 50
Tweet Dataset	-176.83	-172.79	-169.68	-162.72	-159.96
User Dataset	-152.23	-151.81	147.35	-139.77	-137.25

of fact, the average values of coherence increased while increasing K, leading to higher quality and finer-grained topics. Since we had to analyze and understand the obtained topics, we needed a feasible number for K. By looking at Table II, we chose the value for which the immediate information gain was higher, an approach very similar to the elbow method used to determine the optimal number of clusters for K-means clustering. In our case, the elbow was represented by K equal to 40. Hence, the chosen topic model was the one of the "user" dataset with 40 extracted topics.

Table III presents the most relevant topics in the selected model. We described each topic by showing the top 10 most frequent words which appeared in the tweets related to that topic. Topics 6 and 23 were by far the hottest topics since they were associated with respectively 34% and 27% of all our tweets. Topics 4, 32 and 14 appeared in less than 20% of the documents, instead. Of course, we did not consider cancer in the list of the most relevant words since it would have been frequent in every topic.

We exploited topic models to extract information from our dataset, to better understand

TABLE III: REPRESENTATION OF THE MOST FREQUENT TOPICS.

Topic 6	Topic 23	Topic 4	Topic 32	Topic 14
risk	caus	treatment	drug	treatment
breast	food	risk	approve	new
caus	treatment	use	lung	#health
reduc	drug	percent	treatment	gene
increas	via	get	share	fund
new	death	high	patient	medic
woman	eat	help	fda	dure
prevent	good	give	die	tip
awar	skin	healthi	think	narrow
help	like	prevent	talk	effect

what users were posting about cancer. Moreover, we wanted to observe the topic distributions for genuine and spammer users, to determine whether they were different.

CHAPTER 6

COMPUTATION OF FEATURES AND RANKING

Performing supervised learning to detect spam is very difficult because there are no large-scale ground truth datasets which can be used to train a model to separate the two classes: spammers and non-spammers. Some previous works used pseudo fake reviews or exploited some mechanism to assign labels to documents automatically [11; 8]. As a matter of fact, one of the bottlenecks for classification is the labeling process. Traditionally, many documents have to be labeled to obtain a significant amount of information, but this operation is often done manually and therefore, it requires a lot of time and effort [49]. Moreover, labeling documents by reading them is a very difficult task, because spammers may craft those documents and make them look just like genuine ones. Since we did not have neither the time nor workforce to go through every single tweet for each user and label them manually, we had to come up with an alternative solution. Similarly to what was done in [4], we defined and computed some behavioral features which could help us determine the likelihood of a user being suspicious. Since spammers and non-spammers have different goals, we expected them to have different behaviors: genuine users are more likely to interact with each other, while spammers try to spread information and communicate with as many people as they can. We combined the features to obtain a spam score for each user and then we ordered them in descending order generating a ranking.

6.1 Behavioral Features

We extracted two types of features:

- **Content features**, which were derived from the text and the content of tweets posted by every user.
- **Account features**, which were computed according to the user information retrieved with the `get_user` operation. Since almost 200 accounts were banned before we could perform the search for users' information on Twitter, we could not assign values to any of those features for those users.

Every single feature was normalized dividing by the maximum attribute value, in order to make their values fall between 0 and 1.

6.1.1 Content Features

Here follows the list of content features:

- **Number of replies received**: the higher the authority of the user posting the update, or the source associated with the content, the higher the probability of the message to be replied [5]. In fact, spammers and users spreading misinformation are more likely to be ignored by the population of Twitter. However, in the whole collection, a very low percentage of tweets had been replied, about 10%, so this feature just helped to identify authoritative accounts. We computed both the total number of replies received and the fraction of tweets replied for each user.

- **Number of retweets**: similarly to the number of replies, spammers are less likely to have their tweets retweeted with respect to genuine users. The only exception is that people belonging to the same spamming group, who cooperate to propagate misinformation, may retweet the posts of their associates. Indeed, the percentage of tweets retweeted was higher than the one of tweets being replied, slightly below 20%. Broadly, we could say that a lower number of retweets corresponded to spammers. We computed both the total number of retweets and the fraction of tweets retweeted for each user.

- **Maximum content similarity**: we calculated the content similarity of all the tweets posted by the same user. We chose cosine similarity [50] as a measure for similarity: after having transformed each document into a TF-IDF vector, following the bag of words approach, we computed the similarities among vectors with this formula, which represents the angle between two vectors A and B:

$$similarity = cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6.1}$$

  where A_i and B_i are the components of vectors A and B.

  Spammers do not usually spend as much time and effort writing tweets as genuine users do, so they often post the same or similar messages multiple times. This way they can saturate the platform with their content. However, an account may be considered as spam on Twitter if it posts duplicate updates: spammers evolved to adapt to anti-spam strategies by adding mentions or hashtags to duplicate messages to avoid being detected

[1]. As in [4], we considered tweets to be near-duplicate if their cosine similarity value was higher or equal to 0.7. We computed the maximum value of content similarity between two messages and the number of near-duplicate posts for each user.

- **Number of URLs**: opinion spammers tend to spread advertisement or misinformation, and they usually include links to suspicious websites in their messages, which can contain a lot more information with respect to the short, specific post updates. According to what we found working on our dataset, the malicious URLs pointed to fake news or advertisement websites, and bad, removed or not existing links. On the other hand, URLs posted by authoritative accounts, i.e., accounts associated with journals or universities, led to websites containing articles written by trusted sources. Even if Twitter is using a blacklist mechanism to find URLs linking to malicious sites and filter tweets considered as spam, the possibility of shortening URLs represents a great vulnerability, allowing users to evade blacklists completely. The URLs shortening provides a redirection service from a short URL to an arbitrary length URL, helping the link to fit into the post too [6]. We computed both the number of URLs per tweets and the fraction of tweets containing URLs for each user. If a post contained 'http://', 'https://', or 'www.' it was considered having an HTTP link.

- **Number of links to YouTube videos**: this is a brand-new feature that, to the best of our knowledge, had never been exploited before. By going through the messages in a temporary version of the ranking, we realized that a high percentage of users posting links to YouTube videos had to be considered suspicious. Many videos were about advertising

some product or some special treatment, while many others were removed because of the YouTube policy [51]. A lot of them, instead, promoted a specific food, drug or lifestyle as a solution for cancer prevention or treatment, but without providing logical reasons or quoting any trusted source. Moreover, the vast majority of these videos were published by poorly followed accounts. Therefore, we computed both the number of YouTube links per tweet and the fraction of tweets containing a YouTube link for each user. If a post contained '/youtu.be' or 'youtube', it was considered having a YouTube link.

- **Number of hashtags**: users may post the symbol '#' followed by a term to name a topic on Twitter. Trending topics are the most mentioned and popular subjects of the moment: if many messages contain the same topic, it may become a trending topic. Spammers tend to post a lot of tweets referring to many trending topics, even if they are not related to the content in any way, just to lure users into reading their posts [1]. It is a way to make sure that their ideas are propagated and gain other users' attention. We computed both the number of hashtags per tweet and the fraction of tweets containing hashtags for each user. If a post contained the symbol '#', it was considered having a hashtag.
- **Number of mentions**: by inserting '@' followed by a username in their messages, people may address to specific users. Users exploit this mechanism to keep track of conversations and to discover each other on Twitter. Unfortunately, spammers abuse of this service to send unsolicited messages to other users, who are not often even friends with them, just as a way to reach out to as many people as possible with their malicious content. We

computed both the number of mentions per tweet and the fraction of tweets containing mentions for each user. If a post contained '@', it was considered having a mention.

- **Average length of messages**: the length of a post, i.e., the number of words it is composed of, might be significant in detecting suspicious users. It is likely that users with some information to propagate, possibly with suspicious intentions, will make full use of all the 140 characters to describe their ideas better. Therefore, for all users, we computed the average number of terms contained in their messages.

6.1.2 Account Features

Here is the list of extracted features associated with each account:

- **Age of the user account**: Since the infringement of Twitter rules [25] may lead to the suspension of the account, spammers are most likely to create and be associated to new accounts, because previous ones may have been suspended. Moreover, even without considering suspensions, the activity window of suspicious users is different from the one of genuine users: they may register and post content in a short burst [11] when they need to propagate their ideas, and they do not behave as longtime members. It was showed that it is useful to exploit the freshness of accounts in spamming detection. Hence, we computed the age as the timestamp difference between the last day of October 2017 and the account creation date. Intuitively, the shortest the age, the higher the probability of a user being a spammer [5].

- **Location**: It would have been useful to analyze the spatial distribution of users. Observ-

ing it could have led to finding spatial patterns and groups of cooperating accounts, but it could also have been a way to understand users' activity better [8]. Unfortunately, this information was neither available nor reliable for most users in our dataset, and therefore we could not exploit it.

- **Number of tweets posted in the considered time period**: posting many tweets may indicate an abnormal behavior. Indeed, spammers tend to post as many messages as they can to spread information and reach out to more users. Furthermore, many of them may post the same message over and over, with the very same content and minor modifications. Therefore, the higher the number of messages posted during the studied period, the higher the probability that users had suspicious intentions.
- **Statuses count**: As for the number of tweets posted in the considered time period, also the total number of messages from the account creation date may be a good indicator of the user's intentions. As a matter of fact, a user posting many updates is more likely to have many pieces of information to share, possibly even malicious ones. Therefore, the higher the total number of messages, the higher the probability that the user had suspicious intentions. We computed both the total statuses count and the number of statuses per day for each user.
- **Time between tweets**: the size of the time windows between posts from the same users is critical in detecting their intentions. As a matter of fact, some time periods were bursty [7], meaning that the concentration of posts was very high. While genuine users use their accounts to post tweets from time to time with a more uniform time distribution,

suspicious users may post many messages about the same topic in a short period of time to overfill the platform with their content. This posting policy helps them to reach out to more users and propagate their messages. Moreover, users with different accounts may cooperate in the burst, meaning that many users could work together and post a massive amount of messages on targeted topics. Group spamming is very dangerous, since due to the larger number of users involved, it can take total control of the sentiment on a product or service, misleading people reached by that malicious content [3]. In order to observe and analyze time distribution of posts, we computed the average, maximum and minimum size of the time windows for each user. Each time window was computed by difference of timestamps between two posts: the maximum was computed between the last and first message in the studied month, while the minimum was computed between the two closest messages.

- **Followers count**: Commonly, the number of followers of a user is associated with her popularity, but also to her trustworthiness. Usually, spam and malicious accounts do not have a large number of followers [1]. However, in some cases, they may have many followers. This may be achieved by collusion of many different spam accounts, which follow each other to increase their credibility, but also by luring genuine users to follow them. In the majority of cases, suspicious users had fewer followers with respect to genuine ones, also because genuine accounts posting news, associated with journals or authoritative sources, had a very large number of followers.

- **Friends count**: as a matter of fact, malicious users try to reach with their content as

many people as they can. In order to do that, in addition to exploiting mentions and retweets, they usually follow a large number of users. A friend is an account followed by the user. However, due to Twitter spam-policy, users may follow a limited amount of accounts in a given time period. They cannot perform an aggressive following [25], which happens when users start following a large number of accounts in a rather short period. Therefore, we determined that, for the computation of the spam score for each user, the higher the number of followees or friends, the higher the probability of the user being suspicious.

- **Reputation**: one important indicator of users' intentions is the correlation between the followers and friends count. Indeed, trustworthy users, like journals or popular characters, are likely to have many followers and a smaller amount of friends. On the other hand, spam accounts have a high ratio of followers per followees in comparison to non-spammers [5]. We computed the reputation according to the formula:

$$Reputation = \frac{\#Followers}{\#Followers + \#Friends} \tag{6.2}$$

Our reputation may be compared to the Degree Prestige value [49] for Social Networks analysis. Given a directed graph defining the structure of the social network, with a set of actors representing its vertices and the links among them as edges, the importance of an actor is related to the number of links connecting her to other actors. In particular, the prestige is a measure of the prominence of an actor in a social network that focuses

only on in-going links to define the importance of a node: an important actor is one with many in-links. The degree prestige is computed as:

$$P_d(a) = \frac{d_I(a)}{n-1} \tag{6.3}$$

where d_I(a) is the in-degree for actor a, and n is the total number of actors in the network. Similarly, in our case, the network was composed of the set of followers and friends for a given user. We computed the in-degree as the number of followers and the total degree as the sum of followers and friends. Intuitively, a suspicious user is supposed to have a lower reputation with respect to genuine and authoritative ones. Indeed, we expected them to have a number of followers much lower if compared to the number of friends.

- **Number of lists**: other available piece of information for an account was the number of lists it was enrolled in. A list is an alternative method by which a user may follow other users on Twitter. It is an organized group of users: by accessing the list, the members may see what has been posted by all the enlisted accounts [52]. Usually, lists are related to particular topics and group together users with the same interests. Being part of a list is another way to communicate with many users. Hence, suspicious users are supposed to try and become members of lists to propagate their ideas better. The higher the number of lists they were part of, the higher the probability of users being malicious.

- **Flag found**: because almost 200 Twitter accounts had been banned, we could not collect all the needed information for those users. To represent that condition, we added this flag

which was not used for the computation of the spam score but was useful for classification. It is worth mentioning that we did not consider all of those accounts as suspicious: more than two-thirds were labeled as genuine users since their behavior was not considered malicious, at least for their posts regarding cancer.

- **Spam sources**: instead of using a list of spam words as in [5], we generated a list of untrustworthy sources which posted dubious claims without evidence. We included in the list mostly websites which were known to publish unverified information. Of course, all tweets pointing to those websites were considered suspicious, because they were contributing to spreading misinformation, even if unwittingly. Therefore, we computed the number of references to those untrusted sources per tweet. However, we did not use this feature to compute the spam score, but it helped our classifiers a lot to distinguish between genuine users and suspicious ones.

- **Username**: as a matter of fact, many users in social media such as Twitter register multiple accounts to post messages multiple times [16] avoiding suspension. They often collude, posting many tweets on the same topic at the same time to have a greater impact on the subject. When users make use of multiple accounts, they usually do not waste much time thinking about different usernames, and quite often they use similar ones. As for the spam sources feature, we did not use the username to compute the spam score, but it was exploited for classification: indeed, a user with a name which was very similar to a suspicious username was likely to be suspicious as well.

6.2 Ranking

Once all the features had been extracted, we finally had to combine them in order to compute a spam score for each user obtaining a final ranking. We also had to weigh them, according to their relevance, to improve the quality of the ranking. However, we did not require very accurate scores since we only had to get approximate values which could help us identify the top users in our list. Indeed, we needed to obtain a rough ordering for the users in order to start analyzing and labeling the ones in the highest positions of the ranking.

We calculated a total of 26 features, and we normalized them to have values falling between 0 and 1. We simply added them up to compute the final spam score. The normalization process was done by dividing each attribute by the maximum value associated with that attribute. In some cases, due to the peculiar value distribution of some features, we first computed the log of both values and then we performed the division. The reason behind this process lied in the fact that for some features the maximum value was incredibly larger than average values. Therefore, we thought that the logarithmic scale could better represent the distribution of values. We performed this logarithmic scaling on the following features: number of tweets during the considered month, the total number of statuses and the number of statuses per day, followers count, friends count and reputation.

Finally, by summing up all the features multiplied by their weights, we obtained the spam scores for all users. Of course, for those users who had been suspended by Twitter, we could not compute all the features related to the account, and we did not calculate the spam score

either. Hence, we treated suspended users differently during the labeling process, analyzing their activity independently from their ranking position.

CHAPTER 7

LABELING PROCESS AND ANALYSIS OF EXTRACTED FEATURES

Due to the lack of large-scale gold-standard data which could be used to design and evaluate spam detection algorithms, performing supervised learning for spam detection on an extensive collection of documents is a complex task. As a matter of fact, classification needs a labeled training set to create the model which can be used to predict the class for each record in the dataset correctly. Moreover, as opinions in social media have been exploited more and more, opinion spamming is becoming sophisticated, making the detection problem even more complicated [3]. Indeed, it is very hard to recognize fake opinions by manually reading them. In this chapter, we describe in detail our labeling process and then illustrate the value distributions for our most relevant computed features to verify that users labeled as suspicious followed specific behavioral patterns.

7.1 Labeling Process

In our dataset, we had more than 36 thousand users to label, and we could not afford to read all of their posts because it would have taken too much time. Hence, we had to come up with an alternative solution. As we already said, it is very difficult to distinguish between genuine and malicious posts. To detect whether a user was suspicious or not, and to assign labels, we performed multiple checks:

- **Message content**: of course, we first analyzed the content of the messages. In some

cases, they contained dubious claims about foods, drugs or natural products which could prevent or even treat cancer. One remarkable example concerned camel milk and its curative properties for cancer. However, in the majority of cases, we could not tell the difference between suspicious and genuine users, just by reading the tweets.

- **URLs analysis**: since many tweets contained URLs, we had to understand the intentions of the users sharing those links. Usually, they pointed to some web page supporting and better illustrating the content of the message: it could be a research study published by some journal or university, but also magazine articles and blog posts. Our policy here was to draw a neat line and distinguish between trusted and unreliable sources. All the content coming from authoritative sources were considered genuine since they intended to spread useful information. On the other hand, posts coming from personal blogs or articles on websites which are not regarded as trustworthy were considered suspicious, because their statements were not supported by evidence and they often tried to induce the user to purchase some product. Moreover, many tweets contained also URLs pointing to advertisement websites or even to phishing, malicious and not existent links. One specific activity we found in our dataset was Twitter Hijacking [6]: suspicious users retweeted messages posted by other users, usually prominent and authoritative ones, modifying them and appending spam URLs. This way they exploited the trust gained by those prominent users to spread malicious links and misinformation. Finally, one particular type of URL was the YouTube link: many messages linked to videos. Similarly for HTTP links, we found genuine videos with trusted sources and supported by concrete evidence,

and suspicious ones, with odd claims, advertisements or removed videos [51]. All the users posting this kind of content were considered suspicious.

- **User behavior**: the last step in our analysis concerned how the user behaved on the platform. We were looking for standard spammer features: number of friends, number of followers, ratio followers per friends, time activity windows, posting frequency, topics and trending topics in their messages. This analysis helped us to find many users that we could consider suspicious. It is worth mentioning that a few users we defined as suspicious were pretty popular, with some thousands of followers too. We were able to detect their malicious activity because they are known fake news and untrustworthy sources who suggest dubious methods for cancer treatment and prevention.

Due to the complexity of the analysis described above, it is clear that we could not perform such an investigation on every single user. As a matter of fact, it was a very time-consuming study. In order to be able to individuate suspicious users without having to go through the whole dataset, we performed a labeling process composed of multiple steps. As shown in [1], the percentage of spam on Twitter is close to 3%, therefore our goal was to reach about one thousand suspicious users. The basic idea was to extract a list containing a subset of users and to study only the accounts in the list.

- First, we used the top 700 users of the ranking as the initial seeds for our list, the ones who were most likely to be malicious. Hence, we analyzed this set of users. We chose 700 arbitrarily, as a compromise between the effectiveness of the analysis and its feasibility. As a matter of fact, it was incredibly time consuming and we could not afford to go

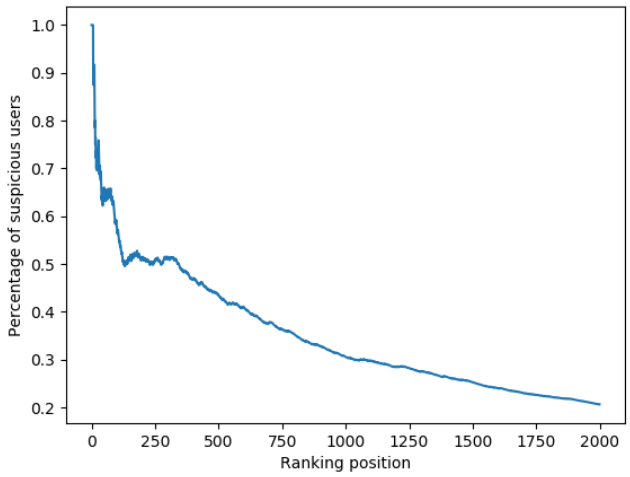

Figure 2: Distribution of suspicious users: top 2000 positions

through thousands of users. Moreover, our ranking was proved to be quite effective since a high percentage of this set of users was labeled as suspicious, the 37%. However, this percentage would decrease when considering lower positions in our ranking. Figure 2 and Figure 3 show the distribution of suspicious users over the ranking positions, proving that suspicious users were likely to occupy the top positions in our ranking.

- As second step, we tried to look for all the users with content similar to the ones posted by the previously found malicious users. Once again, we used the cosine similarity to measure the distance between documents. We used the same threshold to define near-duplicate tweets, which was 0.7. This way, we extended the list of potential suspicious users. We went through all the users in this list and labeled them. This second step was

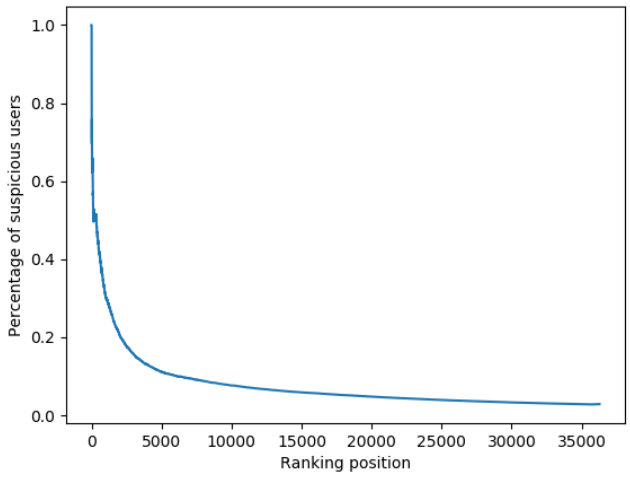

Figure 3: Distribution of suspicious users

repeated multiple times to extend the list further, as it was growing. By looking at the results of this process, we could already determine that many accounts were cooperating to spread the very same information. Indeed, this step helped us define groups of users posting near-duplicate messages who could have been working together.

- As third step, we expanded the list by adding all the users whose messages contained some unique keywords. Since some claims, also dubious ones, were particularly popular on the platform, we wanted to individuate all the accounts posting about that specific subject and analyze their activity. First, we looked at all the messages referring to sources which were known as untrustworthy, either because of their lack of evidence or because they were maliciously advertising some product. All the accounts posting messages referring

to those sources were labeled as suspicious. Moreover, we inserted in the list all the users who tweeted with the following keywords: alternative, astrology, natural, marijuana, cannabis, nutrition, chiropractic, acupuncturist, holistic, and many others. This way we could study what users said about these trending topics, and we found out that a lot of them were spreading suspicious information. Mostly due to untrusted sources and references, almost one-third of users posting messages containing those specific keywords were labeled as suspicious.

- As last step, we added some random users to the list in order to reduce the bias in our analysis. However, the percentage of malicious accounts we found, in this case, was a lot lower. We picked users randomly from the ranking, and, as expected, the likelihood of the ones at the bottom being suspicious was lower with respect to the ones with higher spam scores.

- Finally, we performed a special analysis for those accounts that were suspended by Twitter. For that set of users, we could not find information about the account features, and we did not compute the spam score. Hence, we added them all to our list and analyzed their messages. Unexpectedly, less than one-third of them satisfied the requirements for being defined as suspicious. It is likely that they were suspended for their wrong behavior, but at the same time, they did not behave maliciously in posting tweets related to cancer. As a matter of fact, many users had a dual behavior, sometimes acting as spammers, sometimes as genuine. Because of this double nature, the accuracy of class prediction

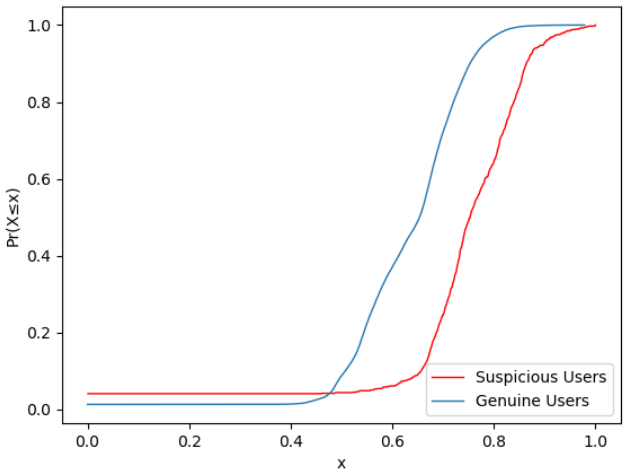

Figure 4: CDF for spam score

decreased, since the trained models had a hard time assigning the proper label to bivalent users.

At the end of this process, we reached more than one thousand suspicious users, an amount which was consistent with the expected 3%. However, we cannot exclude that more users, who were not included in our list, could potentially be suspicious.

7.2 Cumulative Density Functions for Behavioral Features

After we completed our labeling process, we wanted to verify whether our features could be useful in distinguishing suspicious users from genuine ones. As a matter of fact, we exploited those features to compute the spam score and generate the ranking, which was the critical element in building our list to assign labels. Observing Figure 4, which shows the Cumulative

Density Function for the spam score, we could state that the spam score was positively correlated to the way we assigned labels to users. Indeed, the CDF showed how suspicious users had higher scores with respect to normal users.

Furthermore, we computed the correlation values between our normalized features and the assigned labels. The correlation measures the degree of association between two attributes, represented as a value between -1 and -1, where +1 expresses the strongest possible agreement and -1 the strongest possible disagreement. It was computed using Person Correlation Coefficient [53]:

$$\rho(a, b) = \frac{E(a, b)}{\sigma_a \sigma_b} \tag{7.1}$$

where $E(a, b)$ is the covariance or cross-correlation between a and b, and σ_a and σ_b are respectively the standard deviation of a and b.

For the most relevant features, we obtained values between 0.09 and 0.5. Here we show the CDF for the most discriminative features, i.e., the ones that were the most useful to identify behavioral patterns which could help distinguish between suspicious users and genuine users. As expected, suspicious users had different and specific values for those features with respect to normal ones, for they were likely to have larger values.

Finally, we analyzed the topic distributions for suspicious users and genuine users, but we could not find any significant difference. This means that suspicious users were very good at disguising, talking about the same subjects and using word distributions similar to the ones of normal users.

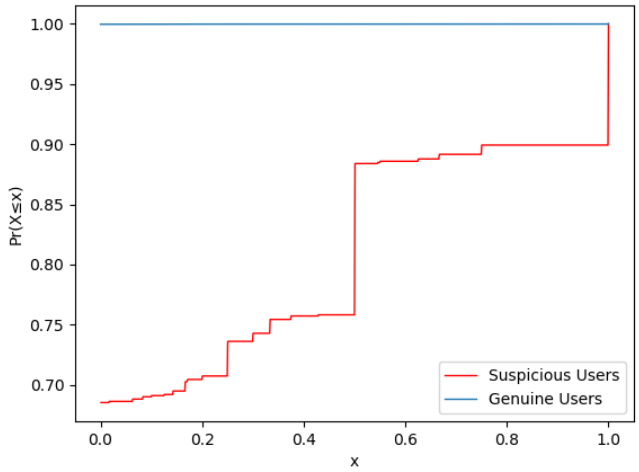

Figure 5: CDF for feature fraction of tweets referring to spam sources

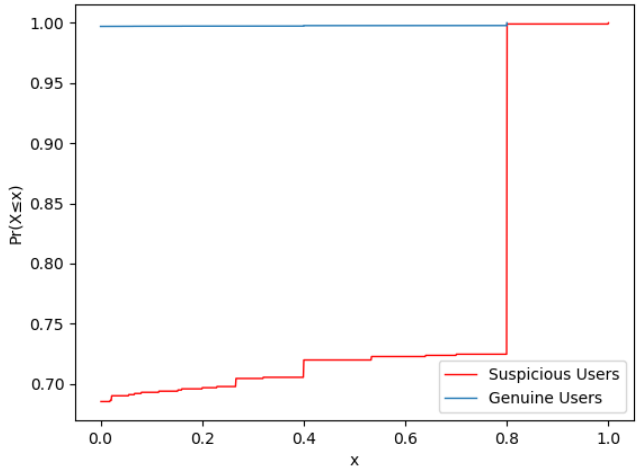

Figure 6: CDF for feature number of YouTube links per tweet

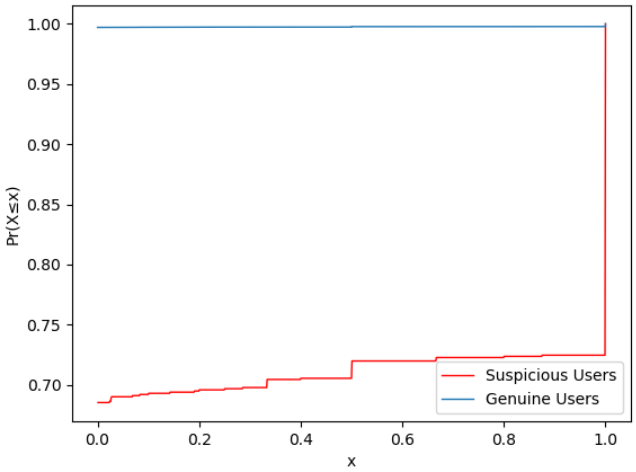

Figure 7: CDF for feature fraction of tweets with YouTube links

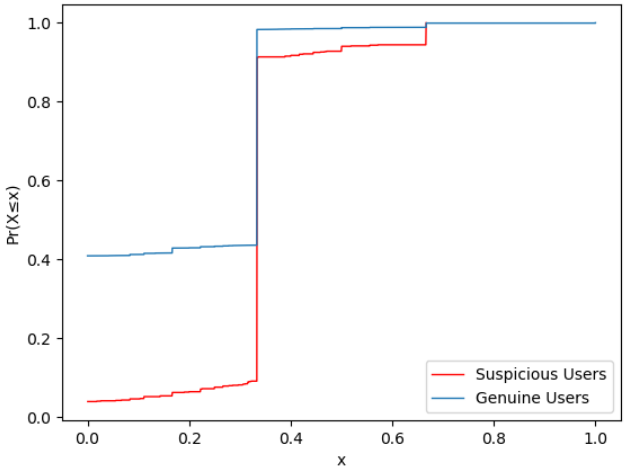

Figure 8: CDF for feature number of URLs per tweet

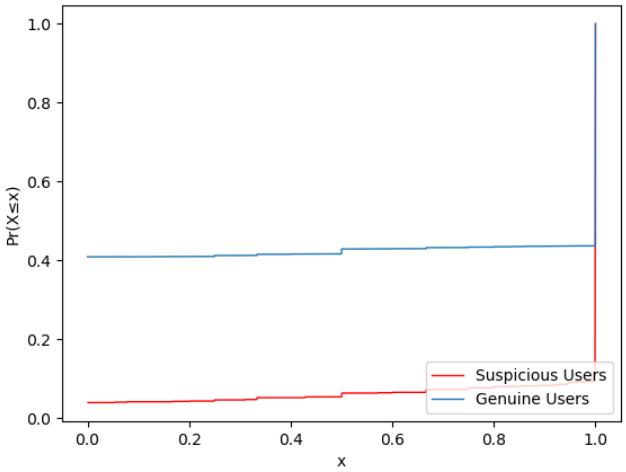

Figure 9: CDF for feature fraction of tweets with URLs

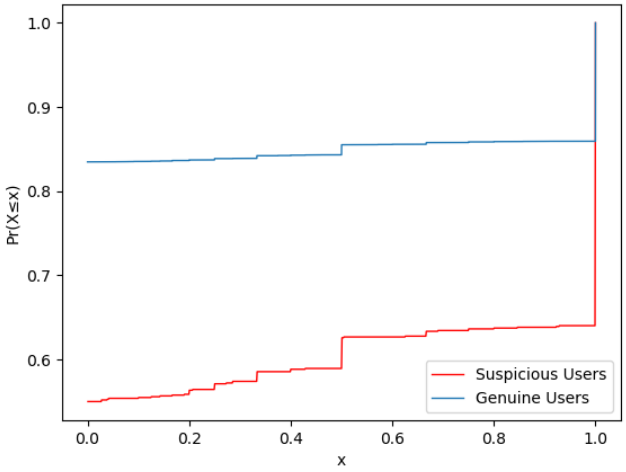

Figure 10: CDF for feature fraction of tweets with mentions

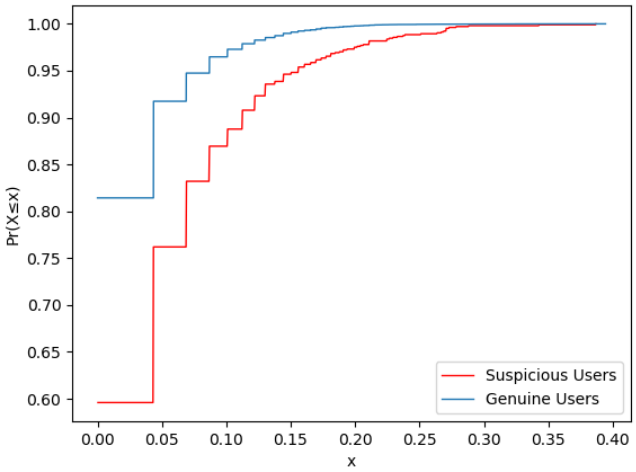

Figure 11: CDF for feature number of tweets posted in October

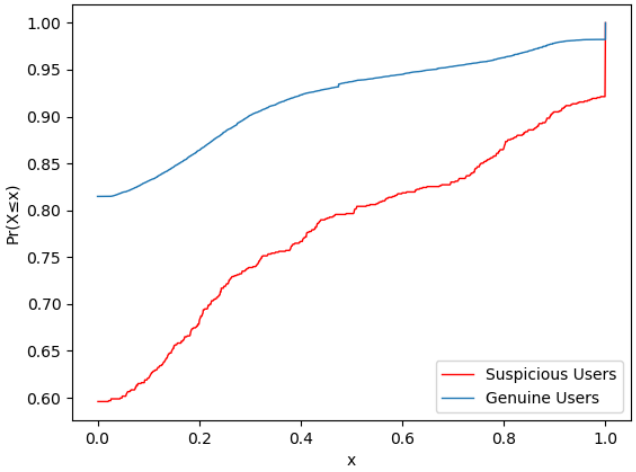

Figure 12: CDF for feature maximum content similarity between documents

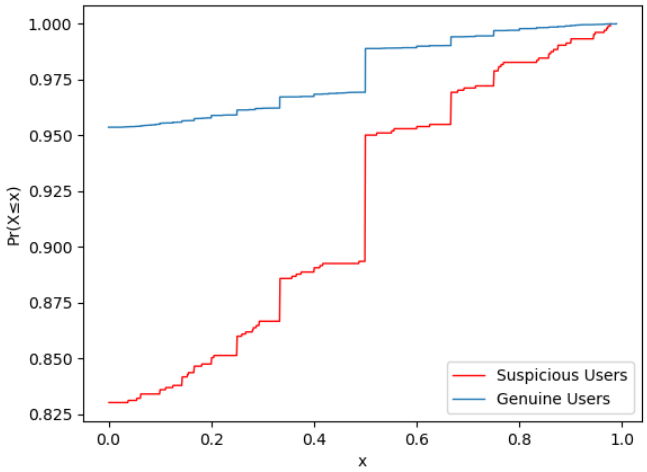

Figure 13: CDF for feature number of near-duplicate tweets

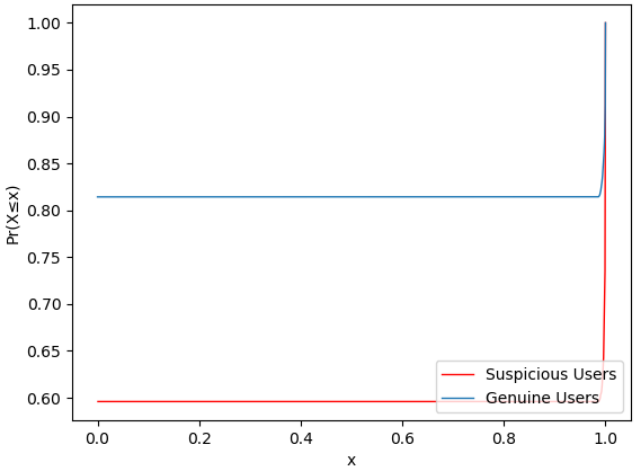

Figure 14: CDF for feature minimum time interval between tweets

CHAPTER 8

CLASSIFICATION

Once we assigned a label to each user, we wanted to try a supervised learning approach to determine whether the textual content of the posts and the features we extracted could be used to separate the users into suspicious and genuine classes correctly. As a matter of fact, as shown in [7; 11], linguistic features could be sufficient to divide the two classes of users properly. This means that posts written by malicious users might follow a particular textual pattern, showing some linguistic differences if compared to genuine ones. Furthermore, we were expecting our computed features to be useful for class prediction too.

For each classifier, we built and evaluated many models for each of the three different training sets we used:

- **Behavioral features**: for this set of data, we trained and evaluated the model using only the features we previously computed. Each user was represented by a vector of features, which were normalized with values from 0 to 1.
- **Linguistic features**: in this case, we performed the classification task considering the text of the messages only. We used the user dataset described in section 4.2.1, in which for each user we had all her posts concatenated into a single document. We used the bag-of-words model [49]: the text was represented as a bag of its words, and for each word we stored the number of occurrences. It is important to underline that many other

features could be used for this task, which are proposed for future work, but we only used word occurrences as linguistic features.

- **Combined features**: finally, we combined both the linguistic and behavioral features, expecting to improve the performance. In this case, we represented each user as a vector, created by appending the features vector to the bag-of-words vector.

To perform supervised learning, we used the most popular machine learning algorithms, as SVM and Bayes classifier, and then we built a Neural Network. Since we did not have any labeled testing set, we used 10-fold cross-validation to evaluate the overall performance. We picked ten as the number of folds since it is the most commonly used value. We divided our dataset into ten folds, with stratified sampling for keeping the same label distribution in the different folds. Then, iteratively, each fold was used as testing set, while the others were used as training set to create a classification model which classified the records in the testing set. Of course, this procedure was run ten times, as ten was the number of folds.

Now, we illustrate the machine learning algorithms we used to perform class prediction, underlining their most relevant characteristics.

8.1 Bayesian Classifier

"Naive Bayes is a high-bias, low-variance classifier, and it can build a good model even with a small dataset" [55]. Moreover, it is simple to use and computationally efficient.

Classification problems basically consists in computing the following posterior probability:

$$Pr(C = C_j | A_1 = a_1, ..., A_n = a_n) \tag{8.1}$$

where C_j is a specific class and a_i is a possible value for attribute A_i.

The predicted class is the one for which the posterior probability is maximal. In our case, the attributes were the behavioral and linguistic features, while the classes represented suspicious and genuine users. According to Bayes theorem, the posterior probability may be computed as:

$$Pr(C = C_j | A_1 = a_1, ..., A_n = a_n) = \frac{Pr(A_1 = a_1, ..., A_n = a_n | C = c_j) Pr(C = c_j)}{Pr(A_1 = a_1, ..., A_n = a_n)} \tag{8.2}$$

where Pr(C = C_j) is the class prior probability, that can be computed by analyzing the training set [49]. The Bayesian classifier is called naive, because of the conditional independence assumption: it assumes that all attributes are conditionally independent given the class Cj. In other words, the values of some feature in a class are unrelated to any other feature. Although this is a very strong assumption which is often not true, the Naive Bayes classifier performs reasonably well in many different tasks. Moreover, to handle continuous attributes without performing discretization, we modeled the conditional probability distributions for attributes in a given class as Gaussian probability densities. Using the naive assumption, we can assign a class to each instance computing:

$$C = \arg\max_{c_j} Pr(c_j) \prod_{i=1}^{|A|} Pr(A_i = a_i | C = c_j) \tag{8.3}$$

However, if in the training data some attribute value never occurs in a class, the conditional probability $Pr(A_i = a_i | C = c_j)$ is equal to 0. Of course, when this 0 value is multiplied with

all the other probabilities, the result will still be 0. To avoid this problem, Laplace correction is used: we made sure that no 0 values occurred by simply adding 1 to each count.

In addition to the Naive Bayesian classifier, we also tried a variant, the Kernel Naive Bayes, where multiple Gaussians are combined to create a kernel density [54] to properly model conditional probability distributions. A kernel-based Bayesian classifier estimates the density of continuous variables using nonparametric kernel-based estimators, instead of parametric Gaussian ones. Kernel-based estimators were proven to be more flexible since they do not have any fixed structure and depend only on the data to reach an estimate. As for all the other classifiers we tried, we used Rapidminer [55], a software platform for data mining based on Java. We used both its Naive Bayes and Kernel Naive Bayes implementations to run the classification task.

8.2 SVM

Support Vector Machines are linear binary classifiers that find a hyperplane to separate two classes of data, positive and negative, suspicious and genuine users. Each instance x in the dataset may be represented as a point in an n-dimensional space, being n the number of features. Each point x $\in R^n$ has a class label.

The goal of the SVM is to find a hyperplane, also called decision boundary, which can correctly separate the positive and negative training data [49]. The hyperplane is in the form:

$$\langle w \cdot x \rangle + b = 0 \tag{8.4}$$

where x is a point, w is the vector of weights and b is the bias. The target decision boundary is the one with the largest margin, which minimizes the error. The margin is defined as the

distance between the hyperplane and the closest points belonging to the class. We can define two margin hyperplanes H_+ and H_-. Hence, the classification task becomes an optimization problem which can be solved with the Lagrangian method. The final decision boundary is given by:

$$\langle w \cdot x \rangle + b = \sum_{i \in SV} y_i a_i \langle x_i \cdot x \rangle + b = 0 \tag{8.5}$$

where SV is the set of support vectors and y_i and a_i are respectively the desired output and the Lagrangian multiplier for support vector x_i. The support vectors are all the points on the margin hyperplanes H_+ and H_-.

Once the decision boundary has been found, to classify any point x, we need to substitute it into the formula and calculate the sign of the obtained result: if it is positive, the point is assigned to the positive class, otherwise to the negative class. SVM is considered one of the best classifiers, suitable for many different kinds of data. However, it has two weaknesses.

First of all, data may not always be separable. To overcome this limitation, we just have to relax the constraints by introducing slack variables and inserting the errors in the cost function which has to be minimized by the optimization problem. This algorithm is called soft-margin SVM.

Another problem is related to the fact that the two classes are not always linearly separable: it means that a straightforward hyperplane cannot correctly divide them. If data cannot be separated in the input space, we have to transform it into a higher-dimensional space such that a linear decision boundary can perform the classification task in the new space, called feature

space. Basically, the process consists in mapping the input data from the input space to a feature space F via a nonlinear mapping Φ that takes a point from the input space and returns a new point of the feature space.

The main problem with this transformation is that the feature space may be huge and therefore applying the SVM algorithm on such a space might be computationally infeasible. Furthermore, the computation alone of the new values in the feature space could be a very complex and demanding process as well. Fortunately, the explicit transformation is not required. The decision boundary computed by solving the optimization problem in the new feature space F becomes:

$$\langle w \cdot x \rangle + b = \sum_{i \in sv} y_i a_i \langle \Phi(x_i) \cdot \Phi(x) \rangle + b = 0 \tag{8.6}$$

There is no need to explicitly compute the transformations, since we only have to calculate the dot product between $\Phi(x_i)$ and $\Phi(x)$. Kernel functions allow us to compute this product, without even knowing the feature vectors $\Phi(x_i)$ and $\Phi(x)$. Indeed, given two points x and z:

$$K(x, z) = \langle \Phi(x) \Phi(z) \rangle \tag{8.7}$$

So, to calculate the decision boundary, we only need to compute the Kernel function for the required points. This procedure is called kernel trick. There are many types of kernel functions; we used two of them:

- **dot kernel**: it is simply defined as the dot product between two points x and y.

$$K(x, z) = \langle x \cdot y \rangle \tag{8.8}$$

- **RBF kernel**: it is defined by

$$K(x, z) = e^{(-g||x-y||^2)} \tag{8.9}$$

  where g is specified by the kernel gamma parameter. It is critical to tune this parameter since it has a significant influence on the overall performance.

In our classification task, the dot kernel performed better than the RBF. Again, we used Rapidminer to implement the SVM classifier.

8.3 Neural Network

"A neural network is a massively parallel distributed processor made of simple processing units", the neurons, which is particularly suited for storing knowledge and make it available for a later use [56]. Those neurons are interconnected and work in parallel within the network. It works similarly to the brain, since:

- Knowledge is acquired through a learning process
- Knowledge is stored by means of synaptic weights, representing the strength of connections between neurons

The learning algorithm is the procedure which defines how knowledge is acquired and stored: the learning process is accomplished through dynamic updates of the synaptic weights associated with each neuron [57]. Inputs are iteratively fed to the network, and the weights are repeatedly adjusted according to the training data and the cost function, which represent the distance of the current output from the desired output. Each iteration is called epoch. Synaptic weights are usually represented with a matrix W and updated using the generic formula:

$$W(n+1) = W(n) + \eta C \tag{8.10}$$

where n refers to the epoch number, η is the learning rate parameter, and C is the cost function. This process is a typical supervised learning task, in which the network learns the expected output associated with each input value. Moreover, neural networks have many important properties which make them suitable for machine learning algorithms:

- **non-linearity**: neurons may be linear or non-linear, according to their activation function. A network made of non-linear neurons can perform well on nonlinear data.
- **adaptivity**: the network may change its synaptic weights according to the surrounding environment. This means that neural networks could be very robust in performing machine learning tasks since they can be easily re-trained to deal with modifications.
- **fault-tolerance**: due to their highly distributed nature, neural networks will keep working and producing good results even with minor damages. It requires big damage to degrade their performance.

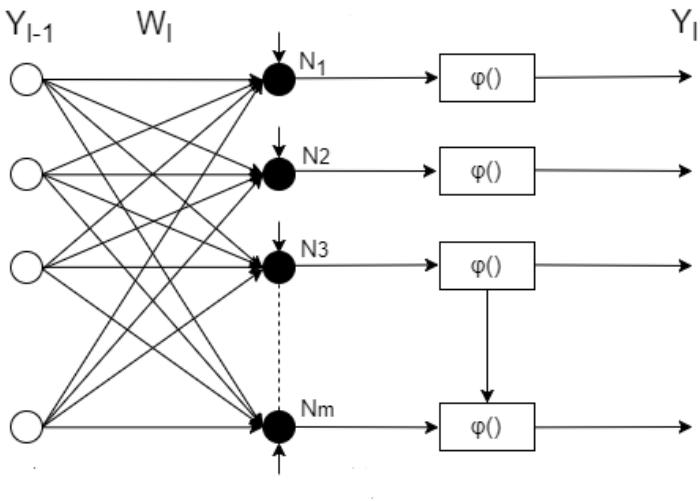

Figure 15: Graphical representation of a layer in Neural Networks

- **evidential response**: they can provide confidence values for their decisions, showing the trustworthiness of their outputs.

The basic processing units, the neurons, may be combined to form different architectures which are organized in layers. In Figure 15, we show the basic structure of a layer in a neural network. The white dots represent the inputs for layer l, Y_{l-1}, while black dots are the neurons of layer l. W_l represents the matrix of weights, ϕ is the activation function for neurons of layer l and Y_l is the output vector. Typically, the neurons of each layer take as input the output of the previous layer.

The output y for a neuron n may be computed as:

$$y_n = \phi(\sum_{i=1}^{m} w_{n_i} x_i) \tag{8.11}$$

where ϕ is the activation function for neuron n, x_i is the i^{th} input and w_{n_i} is the synaptic weight between neuron n and neuron i.

Traditionally, in classification tasks, the input layer represents the input data as a vector, and the output layer is the one responsible for class prediction, producing as output the value corresponding to the predicted class. Other layers are called hidden layers because they are not seen directly either by the input or the output of the network. A neural network may be feedforward, if the information flows only from the input to the output layer, or recurrent, if there is at least one feedback loop, meaning that the output is propagated back to the input.

To perform our task, we used the RapidMiner implementation of neural networks in which we had to tune all the required parameters and chose the optimal architecture. We used a feedforward network with two hidden layers of 50 neurons with rectifier activation function, although usually, one hidden layer is good enough to perform supervised learning tasks [58]. We tried different configurations and this architecture gave us the best results for class prediction. It was trained with the backpropagation algorithm over ten epochs. The backpropagation algorithm adjusts the synaptic weights of each neuron by applying a correction which is proportional to the partial derivative of the cost function with respect to the weight of the neuron itself, using the formula:

$$W(n+1) = W(n) - \eta g \tag{8.12}$$

where g is the gradient of the loss function and W is the matrix of all synaptic weights.

Intuitively, as we reach the minimum of the cost function, the gradient will decrease, and the updates will be smaller until convergence. The value for the learning rate is critical because a value which is too large would produce big jumps during training which could make the model skip the optimum value for the cost function, while a value which is too small would lead to slow convergence. In order to overcome these issues, we used an adaptive learning rate, which decreased in value while approaching the local minima.

As cost function, or loss function, we chose cross-entropy because it was the one which performed best, yielding the highest accuracy. When training neural network classifiers, cross-entropy has many advantages with respect to the quadratic squared-error function [59]:

- while the squared-error function assumes a normal distribution for data, cross-entropy is designed for binary classification, making it suitable for our task.
- squared-error function may be strongly influenced by records with large errors, but cross-entropy exploits a log-linear error function which reduces the impact of outliers.
- squared-error is based on absolute errors, while cross-entropy relies on relative errors, making it work better with both small and large data.

As also stated in [60], average cross-entropy is slightly better at training neural network classifiers with respect to the squared-error function. Moreover, it was successfully used for text classification by previous works as [61]. We also used L1 regularization to prevent weights to increase exponentially. As a matter of fact, we included an additional term in the cost function, a weighted penalty that added a fraction of the sum of the absolute values of the weight.

The resulting cost function for binary classification:

$$C = \frac{1}{N} \sum_{i \in N} (d_i \log(y_i) + (1 - d_i) \log(1 - y_i)) + \lambda \sum_{i=1} |w_i| \tag{8.13}$$

where N is the size of the training set, d_i is the desired output and y_i is the actual output for the i^{th} input, and w_i is the weight associated with the i^{th} neuron.

L1 regularization had the effect of constraining the absolute value for the weights and setting some of them to 0, to reduce complexity and overfitting. Another important technique used to escape from overfitting was early stopping. Early stopping consists in performing class prediction on the testing set at each epoch and storing the results and the current values for the weights while training the network. In the end, we kept the set of synaptic weights which yielded the highest accuracy. This way we would avoid overfitting, since, even if after some epochs the network performed too well on the training data and poorly on the testing data, we could use the previous weights which worked better on the testing set.

CHAPTER 9

EXPERIMENTAL RESULTS

We ran three different classifiers for every dataset using 10-fold cross-validation with stratified sampling to preserve label distributions, since our class distribution was extremely unbalanced. We obtained different performance for each classifier according to the dataset used to train and evaluate the model, but all of them could classify at least one dataset with reasonable accuracy if compared to related work.

9.1 Classification using Behavioral Features

We trained many models using the computed user features, convinced that suspicious users would have different feature distributions than genuine ones. We represented each user as a vector of normalized features, and we fed them to the machine learning algorithms. We wanted to exploit them to find behavioral patterns that could be used to identify suspicious users.

9.1.1 Bayesian Classifier

We tried both Naive Bayes and Kernel Naive Bayes classifiers and we obtained better results with the kernel estimator. This is due to the fact that the kernel could better estimate the density of the features, as continuous values between 0 and 1. We could detect suspicious users with almost 70% recall. Actually, it was also pretty good also at recognizing genuine users, with more than 96% recall. However, since the class distribution was very unbalanced, the

TABLE IV: BEHAVIORAL FEATURES - CM KERNEL NAIVE BAYES.

		Predicted Class		
	Accuracy 95.97%	Suspicious	Genuine	Class Recall
True Class	Suspicious	723	319	69.39%
	Genuine	1143	34087	96.76%
	Class Precision	38.75%	99.07%	

misprediction errors on good users affected a lot the precision for suspicious users, taking it down to less than 40%.

9.1.2 SVM

We ran SVM algorithm using both dot and RBF kernels. The linear kernel performed a lot better than RBF, as we expected, since it is crucial to tune the gamma parameter to make RBF work correctly. With respect to the Bayesian classifier, SVM could better detect genuine users, hardly mistaking them for malicious ones with a recall higher than 98%, but it was less accurate at recognizing suspicious users, with almost 58% recall. The overall accuracy was also higher, but due to the fact that our dataset was very unbalanced, this was not a good quality indicator for the classifier.

9.1.3 Neural Network

The Neural Network was the one performing worst for class prediction using behavioral features. It could not properly detect suspicious users by analyzing their features, reaching a

TABLE V: BEHAVIORAL FEATURES - CM SVM DOT KERNEL.

		Predicted Class		
	Accuracy 97.06%	Suspicious	Genuine	Class Recall
True Class	Suspicious	598	444	57.39%
	Genuine	621	34609	98.24%
	Class Precision	49.06%	98.73%	

TABLE VI: BEHAVIORAL FEATURES - CM NEURAL NETWORK.

		Predicted Class		
	Accuracy 97.86%	Suspicious	Genuine	Class Recall
True Class	Suspicious	325	717	31.19%
	Genuine	61	35169	99.83%
	Class Precision	84.20%	98.00	

very low recall of 31%. Basically, it assigned the genuine label to the great majority of the users. Indeed, it had the highest recall for normal users, very close to 100% and yielded the highest overall accuracy.

9.2 Classification using Linguistic Features

We trained our models for class prediction using the linguistic features extracted from the messages posted by the users. The goal was to determine whether suspicious users followed specific linguistic patterns which could be used to identify them. We used the bag-of-words approach, representing each user as a vector of word occurrences. We also tried to use TF and TF-IDF representation, but they lead to lower accuracy. During text preprocessing, we computed bigrams too, but they did not improve the overall performance. Moreover, we did not exploit POS because they usually do not help in this kind of tasks.

9.2.1 Bayesian Classifier

We used Naive Bayes classifier both with Gaussian and kernel estimators and, again, Kernel Naive Bayes was the one performing best. However, the overall performance was slightly worse with respect to features classification, meaning that with our dataset the algorithm could better predict class labels using behavioral features with respect to linguistic ones. However, it yielded almost 90% recall for genuine users and 67% for suspicious ones. The overall accuracy was significantly lower because it was not as good at identifying normal users as the model which worked with behavioral features.

9.2.2 SVM

While SVM could predict the class labels for users with reasonable accuracy when using behavioral features, it was pretty bad at detecting suspicious users using linguistic features. The linear kernel was again the one performing the classification task better. However, it yielded a low 28% recall for suspicious users. Similarly to what happened when we used the neural

TABLE VII: LINGUISTIC FEATURES - CM KERNEL NAIVE BAYES.

		Predicted Class		
	Accuracy 88.97%	Suspicious	Genuine	Class Recall
True Class	Suspicious	637	316	66.84%
	Genuine	3613	31051	89.58%
	Class Precision	14.99%	98.99%	

TABLE VIII: LINGUISTIC FEATURES - CM SVM DOT KERNEL.

		Predicted Class		
	Accuracy 97.83%	Suspicious	Genuine	Class Recall
True Class	Suspicious	292	750	28.02%
	Genuine	37	35193	99.89%
	Class Precision	88.75%	97.91%	

network with behavioral features, the classifier assigned the genuine label to the great majority of users, reaching the highest recall for normal users.

9.2.3 Neural Network

The neural network was the classifier which performed worst when using behavioral features. On the other hand, it was by far the best with linguistic features. Not only it reached a very

TABLE IX: LINGUISTIC FEATURES - CM NEURAL NETWORK.

		Predicted Class		
	Accuracy 98.74%	Suspicious	Genuine	Class Recall
True Class	Suspicious	775	267	74.38%
	Genuine	189	35041	99.46%
	Class Precision	80.39%	99.24%	

high recall for suspicious users of 74%, but also yielded a precision of 80%. Differently from other classifiers, it could properly identify both suspicious users and genuine ones. Indeed, it is worth mentioning that both recall and precision for normal users were above 99%. Hence, we can state that the neural network found linguistic patterns which could be used to distinguish between suspicious and genuine users accurately.

9.3 Classification using Combined Features

Finally, we tried to perform supervised learning using both linguistic and behavioral features. In order to do that, we simply appended the feature vector associated with each user to the document vector containing the occurrences of each word for the user. Since the behavioral features were almost 30 and the size of the dictionary was a lot larger, we expected the addition of those features to modify the performance obtained by performing text classification slightly. As a matter of fact, in our final vector, the linguistic features outnumbered the behavioral features by far. Therefore the contribution of behavioral features to class prediction was

TABLE X: COMBINED FEATURES - CM KERNEL NAIVE BAYES.

		Predicted Class		
	Accuracy 90.15%	Suspicious	Genuine	Class Recall
True Class	Suspicious	669	286	70.05%
	Genuine	3225	31450	90.70%
	Class Precision	17.18%	99.10%	

small. However, since behavioral patterns matched linguistic patterns, we boosted the overall performance obtained with text classification.

9.3.1 Bayesian Classifier

We observer that, by adding behavioral feature values to the word occurrences vector for each user, we slightly improved the performance. Once again, the algorithm using kernel estimation performed best. Recall increased to 70% for suspicious users and above 90% for genuine ones. Overall accuracy and precision for both classes were improved too. However, the results obtained exploiting behavioral features only for classification were better, showing that Bayesian classifier was more accurate with behavioral features than with linguistic ones to distinguish between genuine and suspicious users.

9.3.2 SVM

SVM performed a lot better with behavioral features with respect to linguistic features. Hence, we expected to improve the accuracy of text classification, but still obtaining worse

TABLE XI: COMBINED FEATURES - CM SVM DOT KERNEL.

		Predicted Class		
	Accuracy 98.12%	Suspicious	Genuine	Class Recall
True Class	Suspicious	400	642	38.39%
	Genuine	40	35190	99.89%
	Class Precision	90.91%	98.21%	

performance than when using behavioral features only. Indeed, we raised precision and recall for suspicious users respectively to almost 91% and 38%. However, when comparing the obtained performance with the ones yielded by features classification, it was clear that SVM was by far better suited to distinguish between suspicious and genuine users using behavioral features than with linguistic ones. Since linear kernel SVM performed better than RBF SVM on both the previous tasks, it had better performance also combining the features.

9.3.3 Neural Network

Differently from both Bayesian classifier and SVM, the neural network performed a lot better with linguistic features with respect to behavioral ones. In fact, it was the best at text classification and the worst at behavioral classification. However, when combining behavioral features with linguistic features, the algorithm performed even better. Precision and recall for both classes slightly improved, yielding respectively 75% and 81% for suspicious class. Exploiting behavioral features helped the network to identify suspicious and genuine users

TABLE XII: COMBINED FEATURES - CM NEURAL NETWORK.

		Predicted Class		
	Accuracy 98.79%	Suspicious	Genuine	Class Recall
True Class	Suspicious	783	259	75.14%
	Genuine	180	35050	99.49%
	Class Precision	81.31%	99.27%	

better. Indeed, the model learned on this dataset was the one which performed the classification task most accurately.

9.4 Classification Results Overview

We give a final overview of the results obtained with different algorithms and different datasets. We observed that Kernel Naive Bayes and SVM algorithms worked a lot better using behavioral features than using linguistic ones. On the other hand, the Neural Network did the exact opposite, detecting both suspicious and genuine users accurately when using the text associated with each user and performing poorly on behavioral features. The difficulty in distinguishing between suspicious users and normal ones laid in the fact that suspicious users have become very good at spamming opinions, disguising as good users and making it harder to detect them. As a matter of fact, they often showed a dual behavior which did not help our classifiers to catch the differences with normal users. They might post normal messages with genuine content and, sometimes, malicious ones, or they could be associated with accounts with

TABLE XIII: KERNEL BAYES - PERFORMANCE.

Kernel Naive Bayes	Suspicious Class			Genuine Class		
	Precision	Recall	Fscore	Precision	Recall	Fscore
Behavioral Features	38.75%	69.39%	49.73%	99.07%	96.76%	97.90%
Linguistic Features	14.99%	66.84%	24.49%	98.99%	89.58%	94.05%
Combined Features	17.18%	70.05%	27.59%	99.10%	90.70%	94.71%

good reputation, which could have been bought and then used to propagate misinformation. However, our classifiers were able to detect some abnormal linguistic and behavioral patterns associated with the users we defined as suspicious and exploited these patterns to distinguish them from genuine ones properly.

TABLE XIV: SVM DOT KERNEL - PERFORMANCE.

SVM Dot Kernel	Suspicious Class			Genuine Class		
	Precision	Recall	Fscore	Precision	Recall	Fscore
Behavioral Features	49.06%	57.39%	52.90%	98.73%	98.24%	98.48%
Linguistic Features	88.75%	28.02%	42.59%	97.91%	99.89%	98.89%
Combined Features	90.91%	38.39%	53.98%	98.21%	99.89%	99.04%

TABLE XV: NEURAL NETWORK - PERFORMANCE.

Neural Network	Suspicious Class			Genuine Class		
	Precision	Recall	Fscore	Precision	Recall	Fscore
Behavioral Features	84.20%	31.29%	45.63%	98.00%	99.83%	98.91%
Linguistic Features	80.39%	74.38%	77.27%	99.24%	99.46%	99.35%
Combined Features	81.31%	75.14%	78.10%	99.27%	99.49%	99.38%

CHAPTER 10

DETECTION OF GROUPS OF COLLUDING USERS

A group of spammers has the workforce to post many messages which could influence people's opinion on target subjects. They might promote or discredit products or services without revealing their hidden intentions. Moreover, when cooperating with others, malicious users are harder to detect since they do not directly show such abnormal behavior as single spammers do. For instance, they do not exceed the maximum number of posted URLs, mentions, tweets. As a matter of fact, they can distribute the load of tweets to be posted, each member sharing just a few messages, reducing the chance of being discovered. We consider a group of suspicious users as a set of userids [3]. There could be either a single person or different users behind multiple ids. However, in our work, we briefly investigated the problem of groups of colluding users, without focusing on how many people manipulated the accounts of a suspicious group.

10.1 Extraction of Groups of Suspicious Users

We proposed a simple graph and content-based approach to detect groups of suspicious users who might be cooperating. First, we used Twitter APIs [2] to obtain the list of followers and friends for each user who was labeled as suspicious. Clearly, we could not get such information for those accounts which were suspended by Twitter. We built a directed graph exploiting the relationships between users: each vertex represented a user, while each directed edge modeled the following relationship, in which if a user a followed a user b, an edge was drawn from a to

b. Then, we extracted a sub-graph from our previously created graph by removing all nodes referring to users who did not belong to our set of suspicious users. As a matter of fact, we wanted to discover groups of possibly malicious users and we were not interested in genuine ones for this analysis. Moreover, we needed proof that the members of the discovered groups were actually cooperating, posting the same content and propagating the same information to achieve some common goal. We expected them to collude and post specific messages on target subjects to influence the opinion of the other users on Twitter. Hence, we reduced again our graph, including only connected users who posted similar messages. As we did during the computation of our features, we focused on users posting duplicate and near-duplicate tweets: we measured the cosine similarity to compute the similarity between documents, and we used the 0.7 as a threshold to determine whether two tweets were near-duplicate or not. From a set of more than one thousand suspicious users, we obtained about 40 groups of users who could be possibly working together.

10.2 Groups Evaluation

We extracted about 40 groups composed of suspicious users from our dataset. They were pretty small in size: the great majority was formed by four or fewer members, and the biggest one counted ten users. We now present the most relevant topics shared by members of the same groups:

- the biggest group posted many tweets about astrology, claiming that it could be exploited to predict and prevent cancer.

- some groups, even composed of just a few members, advertised fake clinics for cancer treatment or special therapies to cure cancer.
- many groups proposed natural remedies, including foods, as a solution for cancer prevention and cancer cure, referring to untrustworthy sources.
- other groups posted some dubious claims about particular objects or activities causing cancer without supporting them with evidence.

We could have exploited some other features as timestamps and time windows, posted URLs and mentions to better understand the relationships between users. However, even if we did not perform an in-depth analysis to detect groups of colluding users as in related works, we were able to find small groups of suspicious users who were possibly cooperating to propagate information and achieve a common goal. In our dataset, they mostly tried to propose dubious solutions for cancer treatment and prevention and to warn people, discrediting some products which were said to cause cancer.

CHAPTER 11

CONCLUSION

This study focused on finding suspicious users who posted dubious claims about cancer prevention and treatment on Twitter. As social media are becoming more and more popular and used, nowadays, they represent an important source of information. On Twitter, users may share posts about their lives and their opinions on any subject, communicating with their friends with short messages and allowing to spread information very fast. Malicious users may exploit these features, by posting deceptive messages to promote or discredit target product or service with hidden intentions. They usually target trending topics, like cancer. Therefore, it is crucial to distinguish genuine users from malicious ones. Since we did not have either the knowledge or the authority to separate utterly wrong claims from legitimate ones, we just defined users with abnormal behavior as suspicious.

We collected a set of more than 60 thousand tweets posted in October 2017. Due to the complexity of manual labeling, we extracted a set of behavioral features for each user and combined them to compute a spam score, i.e., the likelihood of a user being suspicious. We exploited that spam score to create a ranking which was used to assign labels to users more efficiently. In our labeling process, we did not have to go through the whole dataset, but we only analyzed the set of users belonging to a list we generated from the ranking.

Then, we ran multiple classifiers on three different datasets, in which users were respectively described by their behavioral features, linguistic features, and combined behavioral and

linguistic features. We found out that machine learning algorithms could quite accurately detect suspicious users by learning from those features. Apparently, suspicious users followed some abnormal linguistic and behavioral patterns which allowed our classifiers to identify and distinguish them from genuine ones. However, due to the dual behavior they manifested, some normal users were mistaken for suspicious ones by our algorithms.

Finally, we briefly investigated the problem of detecting groups of colluding users. By analyzing the content of their posts and their relationships on Twitter, we were able to find some sets of users who could be possibly working together, propagating misinformation to achieve a common goal.

11.1 Future Work

In our work, we used an alternative method to perform manual labeling on our dataset, without having to analyze every single user, which turned out to be effective in identifying suspicious users. We generated a ranking by combining a set of behavioral features extracted from our dataset and by computing a spam score. Future work may focus on improving the accuracy and the effectiveness of similar rankings, especially adjusting the weights associated to each attribute to compute the final spam score. If accurate enough, this could represent a valid alternative to the complex and time-consuming manual labeling process in spam detection.

Moreover, we performed text classification with a neural network. Another approach using linguistic features is the vector representation of words, also called "word embedding". It exploits Vector Space Models, which basically represent words in a continuous space where semantically similar words are mapped to nearby points [62]. Future work might deal with

the word vector representation, which could even provide better results at text classification with respect to the methods we used. Furthermore, more advanced linguistic features might be exploited to boost the performance of text classification.

Finally, we tried a simple procedure to detect groups of suspicious users who could be cooperating to spread misinformation. However, we only exploited the content of tweets and the relationships between users to determine whether they were working together or not. In future work, the detection procedure could be made more effective, by exploiting other features as timestamps, locations, and URLs posted, which could allow to identify colluding users more accurately.

CITED LITERATURE

1. Wang, A. H.: Don't follow me: Spam detection in twitter. In Security and cryptography (SECRYPT), proceedings of the 2010 international conference on, pages 1–10. IEEE, 2010.

2. Twitter docs. `https://dev.twitter.com/docs`. Accessed: 2018-04-10.

3. Liu, B.: Sentiment analysis: Mining opinions, sentiments, and emotions. Cambridge University Press, 2015.

4. Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M., Castellanos, M., and Ghosh, R.: Spotting opinion spammers using behavioral footprints. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 632–640. ACM, 2013.

5. Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V.: Detecting spammers on twitter. In Collaboration, electronic messaging, anti-abuse and spam conference (CEAS), volume 6, page 12, 2010.

6. Grier, C., Thomas, K., Paxson, V., and Zhang, M.: @ spam: the underground on 140 characters or less. In Proceedings of the 17th ACM conference on Computer and communications security, pages 27–37. ACM, 2010.

7. Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., and Ghosh, R.: Exploiting burstiness in reviews for review spammer detection. Icwsm, 13:175–184, 2013.

8. Li, H., Chen, Z., Mukherjee, A., Liu, B., and Shao, J.: Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In ICWSM, pages 634–637, 2015.

9. Jindal, N. and Liu, B.: Opinion spam and analysis. In Proceedings of the 2008 International Conference on Web Search and Data Mining, pages 219–230. ACM, 2008.

10. Mukherjee, A., Venkataraman, V., Liu, B., and Glance, N. S.: What yelp fake review filter might be doing? In ICWSM, 2013.

CITED LITERATURE (continued)

11. Mukherjee, A., Venkataraman, V., Liu, B., and Glance, N.: Fake review detection: Classification and analysis of real and pseudo reviews. Technical Report UIC-CS-2013–03, University of Illinois at Chicago, Tech. Rep., 2013.

12. Mukherjee, A., Liu, B., Wang, J., Glance, N., and Jindal, N.: Detecting group review spam. In Proceedings of the 20th international conference companion on World wide web, pages 93–94. ACM, 2011.

13. Zhang, X., Zhu, S., and Liang, W.: Detecting spam and promoting campaigns in the twitter social network. In Data Mining (ICDM), 2012 IEEE 12th International Conference on, pages 1194–1199. IEEE, 2012.

14. Chu, Z., Widjaja, I., and Wang, H.: Detecting social spam campaigns on twitter. In International Conference on Applied Cryptography and Network Security, pages 455–472. Springer, 2012.

15. Lee, K., Caverlee, J., Cheng, Z., and Sui, D. Z.: Content-driven detection of campaigns in social media. In Proceedings of the 20th ACM international conference on Information and knowledge management, pages 551–556. ACM, 2011.

16. Qian, T. and Liu, B.: Identifying multiple userids of the same author. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1124–1135, 2013.

17. Twitter. `https://en.wikipedia.org/wiki/Twitter`. Accessed: 2018-04-10.

18. Twitter to test doubling tweet length to 280 characters. `https://www.nytimes.com/2017/09/26/technology/twitter-280-characters.html`. Accessed: 2018-04-10.

19. Kwak, H., Lee, C., Park, H., and Moon, S.: What is twitter, a social network or a news media? In Proceedings of the 19th international conference on World wide web, pages 591–600. ACM, 2010.

20. Java, A., Song, X., Finin, T., and Tseng, B.: Why we twitter: understanding microblogging usage and communities. In Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, pages 56–65. ACM, 2007.

21. Huberman, B. A., Romero, D. M., and Wu, F.: Social networks that matter: Twitter under the microscope. arXiv preprint arXiv:0812.1045, 2008.

CITED LITERATURE (continued)

22. Pak, A. and Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In LREc, volume 10, 2010.

23. Wang, H., Can, D., Kazemzadeh, A., Bar, F., and Narayanan, S.: A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In Proceedings of the ACL 2012 System Demonstrations, pages 115–120. Association for Computational Linguistics, 2012.

24. Breast cancer awareness month. `http://www.nationalbreastcancer.org/breast-cancer-awareness-month`. Accessed: 2018-04-10.

25. The twitter rules. `https://help.twitter.com/en/rules-and-policies/twitter-rules`. Accessed: 2018-04-10.

26. Kouloumpis, E., Wilson, T., and Moore, J. D.: Twitter sentiment analysis: The good the bad and the omg! Icwsm, 11(538-541):164, 2011.

27. Hemalatha, I., Varma, G. S., and Govardhan, A.: Preprocessing the informal text for efficient sentiment analysis. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 1(2):58–61, 2012.

28. Snowball stemmer. `http://snowball.tartarus.org/`. Accessed: 2018-04-10.

29. Orange. `https://orange.biolab.si`. Accessed: 2018-04-10.

30. Orange text mining documentatio. `https://media.readthedocs.org/pdf/orange3-text/latest/orange3-text.pdf`. Accessed: 2018-04-10.

31. Bird, S. and Loper, E.: Nltk: the natural language toolkit. In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions, page 31. Association for Computational Linguistics, 2004.

32. Big data polito. `https://bigdata.polito.it/`. Accessed: 2018-04-10.

33. Pyspark documentation. `http://spark.apache.org/docs/2.1.0/api/python/pyspark.html`. Accessed: 2018-04-10.

34. Apache spark. `https://spark.apache.org/`. Accessed: 2018-04-10.

CITED LITERATURE (continued)

35. Hdfs architecture guide. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Accessed: 2018-04-10.

36. Spark rdd programming guide. https://spark.apache.org/docs/latest/rdd-programming-guide.html. Accessed: 2018-04-10.

37. Blei, D. M., Ng, A. Y., and Jordan, M. I.: Latent dirichlet allocation. Journal of machine Learning research, 3(Jan):993–1022, 2003.

38. Zhao, W. X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., and Li, X.: Comparing twitter and traditional media using topic models. In European Conference on Information Retrieval, pages 338–349. Springer, 2011.

39. Chen, Z. and Liu, B.: Mining topics in documents: standing on the shoulders of big data. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1116–1125. ACM, 2014.

40. Hong, L. and Davison, B. D.: Empirical study of topic modeling in twitter. In Proceedings of the first workshop on social media analytics, pages 80–88. ACM, 2010.

41. Jónsson, E. and Stolee, J.: An evaluation of topic modelling techniques for twitter.

42. Steyvers, M., Smyth, P., Rosen-Zvi, M., and Griffiths, T.: Probabilistic author-topic models for information discovery. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 306–315. ACM, 2004.

43. Mehrotra, R., Sanner, S., Buntine, W., and Xie, L.: Improving lda topic models for microblogs via tweet pooling and automatic labeling. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pages 889–892. ACM, 2013.

44. Dirichlet distribution. https://en.wikipedia.org/wiki/Dirichlet_distribution. Accessed: 2018-04-10.

45. Bayesian inference. https://en.wikipedia.org/wiki/Bayesian_inference. Accessed: 2018-04-10.

46. Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A.: Optimizing semantic coherence in topic models. In Proceedings of the conference on empirical

CITED LITERATURE (continued)

methods in natural language processing, pages 262–272. Association for Computational Linguistics, 2011.

47. Perplexity. https://en.wikipedia.org/wiki/Perplexity. Accessed: 2018-04-10.

48. Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., and Blei, D. M.: Reading tea leaves: How humans interpret topic models. In Advances in neural information processing systems, pages 288–296, 2009.

49. Liu, B.: Web data mining: exploring hyperlinks, contents, and usage data. Springer Science & Business Media, 2007.

50. Cosine similarity. https://en.wikipedia.org/wiki/Cosine_similarity. Accessed: 2018-04-10.

51. Youtube policies. https://www.youtube.com/yt/about/policies/. Accessed: 2018-04-10.

52. Twitter lists. https://help.twitter.com/en/using-twitter/twitter-lists. Accessed: 2018-04-10.

53. Benesty, J., Chen, J., Huang, Y., and Cohen, I.: Pearson correlation coefficient. In Noise reduction in speech processing, pages 1–4. Springer, 2009.

54. Rapidminer. https://docs.rapidminer.com/. Accessed: 2018-04-10.

55. Pérez, A., Larrañaga, P., and Inza, I.: Bayesian classifiers based on kernel density estimation: Flexible classifiers. International Journal of Approximate Reasoning, 50(2):341–362, 2009.

56. Haykin, S. S., Haykin, S. S., Haykin, S. S., and Haykin, S. S.: Neural networks and learning machines, volume 3. Pearson Upper Saddle River, NJ, USA:, 2009.

57. Hepner, G., Logan, T., Ritter, N., and Bryant, N.: Artificial neural network classification using a minimal training set- comparison to conventional supervised classification. Photogrammetric Engineering and Remote Sensing, 56(4):469–473, 1990.

58. Dreiseitl, S. and Ohno-Machado, L.: Logistic regression and artificial neural network classification models: a methodology review. Journal of biomedical informatics, 35(5-6):352–359, 2002.

CITED LITERATURE (continued)

59. Kline, D. M. and Berardi, V. L.: Revisiting squared-error and cross-entropy functions for training neural network classifiers. Neural Computing & Applications, 14(4):310–318, 2005.

60. Why you should use cross-entropy instead of classification error or mean squared error for neural network classifier training. https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean Accessed: 2018-04-10.

61. Tang, D., Qin, B., and Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 conference on empirical methods in natural language processing, pages 1422–1432, 2015.

62. Vector representation of words. https://www.tensorflow.org/tutorials/word2vec. Accessed: 2018-04-10.

63. N-gram. https://en.wikipedia.org/wiki/N-gram. Accessed: 2018-04-10.

64. Part of speech. https://en.wikipedia.org/wiki/Part_of_speech. Accessed: 2018-04-10.

VITA

NAME	Massimo Piras
EDUCATION	
	Master of Science in Computer Science, University of Illinois at Chicago, May 2018, USA
	Specialization Degree in Data Science for Computer Engineering, Jul 2018, Polytechnic of Turin, Italy
	Bachelor's Degree in Computer Engineering, Jul 2016, Polytechnic of Turin, Italy
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	2016 - IELTS examination (7.5/9)
	A.Y. 2017/18 One Year of study abroad in Chicago, Illinois
	A.Y. 2016/17. Lessons and exams attended exclusively in English
SCHOLARSHIPS	
Fall 2017	full tuition waiver at UIC
Fall 2017	Italian scholarship for TOP-UIC students
TECHNICAL SKILLS	
Basic level	Assembly 8086 programming,
Average level	Matlab, UNIX shell and scripting, HTML, CSS, Javascript, PHP
Advanced level	GO, C, Python, Java, SQL and PL/SQL, IBM BPM Platforms
WORK EXPERIENCE AND PROJECTS	
Ongoing	GO implementation of a load balancer proxy for HTTP servers
Fall 2017	Neural Network classifier for Twitter Sentiment Analysis on US 2012 Presidential Election

VITA (continued)

Fall 2017	Java implementation of Apriori algorithm for frequent itemsets extraction with multiple minimum supports
Fall 2017	Matlab implementation of a Neural Network for digit classification using MNIST data set
Fall 2017	GO implementation of a recursive program for file archiving
2016 - 2017	Back end software developer
	Software development for a leading Italian electricity and gas manufacturer and distributor. Use of Oracle database for new application developments, as procedures, packages and triggers, and maintenance of previous applications. IBM BPM Integration Designer, software development tool that renders IT assets into service components for reuse in service-oriented architecture solutionsand Process Designer. IBM Procee Designer, graphical user interface tool used for modeling and implementing business processes and providing masks and interfaces for end users