UNIVERSITY OF TRENTO - Italy

Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

FINAL DISSERTATION

RECOMMENDING FOLLOWERS TO IMPROVE REPUTATION ON SOCIAL MEDIA

Supervisor
Mauro Brunato

Co-supervisor
Claudio Giuliano

Student
Andrei Catalin Coman

Academic year 2016/2017

Acknowledgements

This period has represented for me one of the most intense I have ever experienced. It has allowed me to grow both in scientific and personal terms. First of all, I would like to thank all the people who belong to the Future Media Unit (Fondazione Bruno Kessler) who have allowed me to be part of their group. Thank you, Francesco Corcoglioniti, Roberto Zanoli, Yaroslav Nechaev, and Marco Fossati.

I'd like to give special thanks to the head of the unit, Claudio Giuliano, who allowed me to face this great challenge. Without his guidance and persistent help, this dissertation would not have been possible.

In addition, a thank to Professor Mauro Brunato who has always been there to discuss the work I was carrying out. Thank you for your valuable cooperation and support.

I would also like to thank my parents Daniel and Emilia, who have offered me their support in all these years. Thanks to the love, advice, and education they have given me, they allowed me to be what I am today. I would like to thank my little sister Laura, who has brought a breath of freshness to our family that only a child is capable of.

I would like to thank my relatives, which include my grandmother Elisabeta, my aunt Lucica, my uncle Viorel, and finally my cousins Alexandra, Ramona, and Irina.

Finally, special thanks go to my friends. They represented the essential piece of this academic journey. I want to thank you very much Cosimo, Elia, Giacomo, Stefano, Andrea, and Lorenzo. You have proved to be positive and valuable people who have supported me both in tough times and in the most beautiful ones.

Contents

Abstract **2**

1 Introduction **3**

2 Background **5**
2.1 Twitter 5
2.2 Pokedem and @esseredeltoro 6

3 System description **9**

4 Recommender systems **12**

5 Learning algorithms **14**
5.1 Supervised learning 14
5.1.1 Support Vector Machines 14
5.2 Unsupervised learning 16
5.2.1 One Class K-Nearest Neighbor 17

6 User modeling **18**

7 Results and conclusion **22**
7.1 Learning algorithms: offline and online evaluation 22
7.1.1 Offline evaluation 23
7.1.2 Online evaluation 25
7.2 Conclusion 28

Bibliografy **29**

Abstract

These days, social media presence has become more and more important for businesses. However, growing a social media account and improving its reputation by gathering followers are far from being simple tasks, especially for professionals and small businesses lacking the necessary skills and resources.

Context and motivation

In a not too distant past, businesses needed a personal website to promote themselves. Social media overruled this need, providing them a lot of potential customers aggregated in a single place. This led companies to intensely revise their marketing strategies to keep pace with the new phenomenon and make social media platforms integral part of their business core.

Marketing and public-relations specialists had to adapt themselves to the new reality, relocating part of their job to the online space. Nowadays, managing a social account has become a real profession. The new professional figure born from the explosion of social platforms is known as social media account manager. The latter has the responsibility of leading the social media strategy on behalf of a company. This typically implies managing customer service, publishing valuable content and analyze trends, with the ultimate goal of maximizing customer engagement.

However, not all companies have the necessary resources to hire this kind of professionals or to reprofile part of employees to create and keep their online presence. This has given rise to the need for software tools and associated techniques that could take care of most or all the actions a social media account manager can do.

Goal

The purpose of this thesis is to create an engine capable of recommending the social media manager a top-K list of users that if approached through the *following action* are likely to follow back the account that is being managed. In order for such a system to do these suggestions, it needs to learn from previous follow actions carried out by the account manager. The system must be able to learn the type of users that have been followed. This way, it will be capable of proposing new recommendations that are similar to those made in the past by the social media manager.

Personal contribution

This dissertation focuses on the study and creation of a content-based recommender system where recommended items are social media users, described with a rich profile that includes both domain-independent *basic user attributes* and domain-specific *specialized user attributes*. Dealing with a large amount of users' data, extract relevant information and outline a suitable representation is an important part of this work.

Two different learning algorithms are analyzed, one based on supervised learning and the other on unsupervised learning. The performance of these algorithms is then measured by the number of followers each of them gathered. Additional comparisons with a baseline recommender engine are being taken to assess their behaviour.

1 Introduction

Social media has had a great impact on our culture because it completely revolutionized the way people communicate and socialize. Over time lots of social media platforms were born. Some consolidated and managed to create well-known places where people can interact and share content.

The ever-growing popularity of social media has had a significant impact on the advertising industry. Many companies have completely revised their marketing strategies as they started to see these platforms as outstanding candidates for advertising their products. Companies' social accounts have turned into landmarks for customers, comparable to customer services and care.

Although the account offered a direct channel for reaching customers, companies began to consider it too limited, as it was not able to meet all their ambitions. They started to outline new ways to expand themselves, with the sole aim of reaching other potential customers. The first strategy was to take advantage of the paid promotion offered by the social media. This method allows companies to distribute their content, i. e. advertisements that can be either textual or visual, to users who do not necessarily follow the company's account. However, as companies began to abuse of this marketing strategy, users got used to this type of content and simply started to skip any sponsored post.

As this strategy became less effective, companies began to outline new strategies based on influencer marketing. The latter is based on the identification of so-called influencers, i. e. individuals who can influence a large number of people, and the orientation of the company's marketing around them. This new social figure is capable of reaching numerous users scattered across social networks. One thing worth noting is that once a certain degree of popularity has been reached, the influencer does not have to make too much effort to distribute its content. This is because he/she has been able to create a very strong link with followers, so much so that he/she induced them to regularly visit the account/page to check if new content has been posted.

However, achieving this level of popularity is far from being a simple task. In order for an account to become influential, it is necessary to fully understand the domain in which it will operate, analyze the competitors, and last but not least, define a clear goal and a strategy to achieve it.

A user that aspires to become an influencer, can outline different goals to achieve with his/her social media account. Some of these may include:

- *Engagement and interaction.* The creation of relevant content is intended to stimulate followers in interacting through different actions, such as like, comment, share and so on. To provide the engagement experience, the account must be very responsive, i. e. it must be willing to listen to followers' requests, respond to comments and the various questions asked by fans.

- *Traffic.* It consists of increasing the number of visitors following the target account on a regular basis. This is a much sought after factor by companies, as users loyal to the target account, who visit it regularly are more likely to be influenced.

- *Revenue.* Posting content endorsed by various sponsors can allow the target account to generate revenue. However, gains are often related to other parameters. One of the most relevant is the degree of engagement achieved with followers.

- *Followers.* The goal is to maximize the number of followers the target account has. Having a good base of active followers provides the influencer a considerable negotiating power with a potential sponsor.

In this work, the focus is on followers' gaining problem in a Twitter scenario. A user who aspires to become an influencer may decide to manage the account autonomously. However, there are two problems to be addressed. The first one is to actively search for users to follow, who are likely to follow back the target account. The second problem concerns the evaluation of these users, which consists in looking at some attributes such as the number of followers, likes, the content posted and so on, in order to make a decision on whether to follow him.

One way to easily earn new followers is by buying them. There are several services that offer fake users purchase. However, this is only a shortcut that will not benefit the account in the long term. This practice is obviously not compatible with the terms of use of any social network and could endanger the account. Another reason for not using these services is that having fake users means that they do not provide any value to the account. They do not generate any interaction as they are only mere numbers. A potential sponsor is not just interested in the number but also in the degree of engagement that an influencer can establish with followers. Excluding this unethical practice, user search and evaluation remains a time-consuming activity. Therefore, the need arises for some tools capable of addressing those problems.

This thesis focuses on the *following action* that must be carried out by a wannabe influencer or a social media manager on behalf of a particular account. In the following chapters, a learning engine will be outlined that aims to learn the type of users that have been previously followed by the account manager. Finally, the system will try to emulate the manager's task and provide a list of top-K users that are more likely to follow back the target account, if approached through the *following action*.

This type of system is called a *recommender system*. The one that will be described in the following chapters is based on machine learning algorithms. The reason why this engine has been created, is that it will be integrated into a larger application called **Pokedem**, developed by the Future Media Unit at Fondazione Bruno Kessler[1] research center which aims to help social media managers in their daily activities by reducing the workload that these activities involve. The task of the system consists of providing users recommendations to the account manager who operates on behalf of a Twitter account called **Essere del Toro (@esseredeltoro)**, on which the experiments were carried out. It is a community-like account for supporters of Torino Football Club, a professional football team playing in the Italian top football division (Serie A).

Performances of the machine learning based engine will be compared to those obtained by a previous manual user following phase that has been done by the social media manager. A further comparison will be made with a baseline recommender system, represented by the one that Twitter provides in its graphical interface. The evaluation metric is based on the conversion rate value, i. e. the ratio between the amount of followers gained and the total number of users followed.

After this introductory chapter, the thesis is divided into:

- *Chapter 2 - Background.* A preliminary part describing the context in which the work is placed.
- *Chapter 3 - System description.* A panoramic overview of system's structure is being presented in this chapter.
- *Chapter 4 - Recommender systems.* This chapter offers a brief introduction on recommendation systems, the contexts in which they are used and the type of engines that can be outlined.
- *Chapter 5 - Learning algorithms.* A purely theoretical part describing the algorithms used for the design of two different recommender systems.
- *Chapter 6 - User modeling.* This represents the core of the work. The extraction and representation of user information is addressed in this chapter.
- *Chapter 7 - Results and conclusions.* Analysis of algorithms performance and comparison with the baseline version.

[1] **Fondazione Bruno Kessler**: https://www.fbk.eu/

2 Background

2.1 Twitter

Twitter is one of the most popular social networking sites. It can be put under the microblogging category as it combines blogging and instant messaging. In fact, users can broadcast everything about themselves by means of small messages called "tweets", which are restricted to 140 characters. Time and tweets are strictly related. Content can be created and consumed very rapidly with high frequency.

These characteristics made of Twitter one of the most dynamic social platforms available today. Current events and Twitter have become synonymous. In fact, by considering a sphere of great interest such as daily news, most people prefer to follow them directly on Twitter because of the ease of retrieving frequently updated information from popular newspapers.

Despite having one-sixth of the users compared to its major competitor [2], its dynamism managed to draw attention on lots of celebrities and politicians. The reason is that influencers have seen on Twitter a direct communication channel with users that are following them. Businesses also had the same intuition. They started to see this platform as the goose that laid the golden egg, and outlined several strategies to get social as soon as possible. However, this does not mean that doing marketing on Twitter is effortless. Building a strong base of followers, engage them and keep brand's awareness are still complex tasks. Time and resources must be spent to reach and preserve a brand's social media presence.

For a better understanding on the work presented in the following chapters, some Twitter's terminology must be introduced.

- **Tweet**. Content posted by a user. It can range from simple text to more complex media such as images or videos. Terms like *post* and *status* are also used to describe a *tweet*.
- **Retweet**. A reposted or forwarded *tweet* from another user. It represents the *share* mechanism that can be seen on other platforms.
- **Reply**. A user replies to a *tweet*. On other platforms it represents the *comment* mechanism.
- **Direct Messages**. The way users can have private conversations. Messages can only be seen by those directly involved in the conversation.
- **Timeline**. Main section of the website's UI[1] where all *tweets* are collected and represented in reverse chronological order. Most recent *tweets* appear first, while the oldest appear last.
- **Like**. A user shows appreciation for a specific *tweet* without replying.
- **Follower**. A user who follows other users.
- **Followee**. A user who is being followed by other users. In twitter jargon when a *followee* is being followed by a *follower*, it becomes a "friend" of him.
- **Hashtag**. Word preceded by a hash/pound sign (e.g. #torinofc). Its main use is to organize *tweets* by a specific theme or content.
- **Mention**. A user can be referenced within a *tweet* by simply prepending an at sign (@) to the username (e.g. @TorinoFC_1906).

[1] **UI**: User Interface.

- **List**. A curated group of users where they can talk about common interests. The posted content is available only to members of the list.

When Twitter was launched in 2006, friendship relation was not well defined. The reason behind this decision is due to the will of the founders to distinguish the new social media from other existing platforms. Unlike other networks where initially you could only be or not be friend with a user, Twitter provided a unidirectional follow relation between a *follower* and a *followee*.

Within this kind of relationship, the connection is established by the *follower*, which then receives on his/her *timeline* all the contents posted by the *followee*, and interacts by means of a *retweet*, *reply* or *like* action. The true friendship relationship only occurs when the *followee* follows back the *follower*.

One of the main reasons why influencers and businesses firstly adopted Twitter as their primal online communication channel was the ease of engaging and broadcast their own contents to a larger audience, making their account more popular and reputable. Today, however, this valuable relationship mechanism has been adopted by almost all competitors.

2.2 Pokedem and @esseredeltoro

Pokedem is an application developed by the Future Media Unit[2] at Fondazione Bruno Kessler research center that helps a social media manager in his/her daily activities by reducing the burden, required time and skills that are needed to carry out these activities. This way, the manager can focus on activities that require human judgment and creativity [4].

The goal of this application is to overrule trivial social media automation tools that usually only cover basic tasks, such as post scheduling and optimization for different social networks. The aim is to prove that it is possible to create a complex system capable of proposing real content and actions to be carried out on social media.

The application consists of three major components, which are:

- *Recommendations* (Figure 2.1). A ranked, always up-to-date list of recommended social media actions, with explanations of why they are recommended. The account manager may reject, execute immediately, or schedule a recommended action for later execution at an optimal time chosen by the system. The list of possible recommended actions includes:
  - Tweet/Retweet.
  - Follow/Unfollow.
  - Like/Unlike.

- *User Profiling* (Figure 2.2). User profiles are collected by Pokedem to support recommendations. Profiles are computed for the target Twitter audience of the account, identified by navigating the social network from representative seed accounts. Within the chapters, this component is also called profiling service.

- *Analytics* (Figure 2.3). Numerous charts are provided, which monitor the impact of the actions that have been carried out by the account manager. Furthermore, a comparison between the managed account and competitors is provided. Comparisons can be made on different dimensions, such as the number of likes received, retweets and so on, to identify metrics that need improvement.

To support the development and testing of the application's functionality, a Twitter account called Essere del Toro (@esseredeltoro) has been created. The account (Figure 2.4) is presented as a non-personal, community-like account for supporters of Torino Football Club (@TorinoFC_1906), a professional football team playing in the Italian top football division (Serie A). The account only posts

[2]**Future Media Unit**: http://futuro.media/

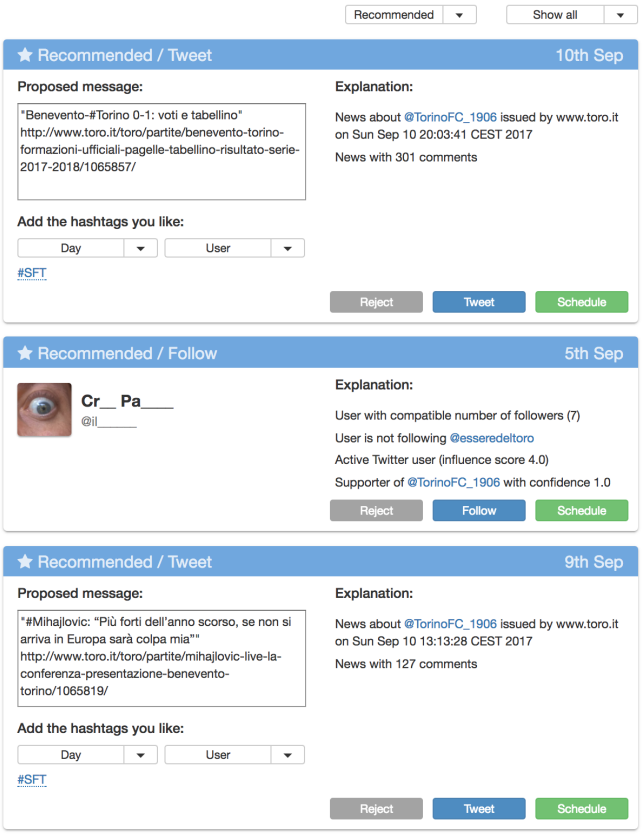

Figure 2.1: Pokedem's recommendations

Screen Name	Created Time	Gender	Type	Location	Inf. Score	Active	Team Name
@Ma_____	2017-03-31 02:41	♂	Person	Milano	–	Yes	TORINO FC
@Lu_____	2017-03-31 02:41	♀	Person	Milano	1	Yes	TORINO FC
@Cl_____	2017-03-31 02:41	♀	Person	Milano	–	Yes	TORINO FC
@Ss_____	2017-03-31 02:41	♂	Person	Milano	–	Yes	TORINO FC
@se_____	2017-03-31 02:41	♀	Person	Milano	1	Yes	TORINO FC
@An_____	2017-03-31 00:20	♀	Person	Pescara	1	Yes	TORINO FC
@ma_____	2017-03-30 21:50	♂	Person	Milano	–	Yes	TORINO FC
@Fr_____	2017-03-30 18:46	♂	Person	Barletta-Andria-Trani	1	Yes	TORINO FC
@An_____	2017-03-30 18:27	♂	Person	Milano	–	Yes	TORINO FC
@al_____	2017-03-30 18:27	♂	Person	Milano	0	Yes	TORINO FC

Figure 2.2: Pokedem's user profiling

content about the team, following Pokedem's recommendations, with the ultimate goal of increasing its popularity.

The account has started from a completely empty state, where no followers were present. As a starting point, an account manager began to manually search for potential users to follow. If initially the choice was based on a single criterion, i. e. to be somehow tied to the football team, over time different heuristics were defined to allow the selection of best candidates to be followed, i. e. those who were most likely to follow back the target account back, if approached through a *following action*.

The difficulty in finding and evaluating users has led to the need for the creation of a recommendation system based on machine learning techniques. This is the exact position in which this thesis is situated, i. e. the one that seeks to exploit all the users followed by the social media manager on behalf of @esseredelteltoro to create a system capable of replicating these actions. The objective was therefore to create a module capable of replacing the previous strategy, while maintaining the same performance.

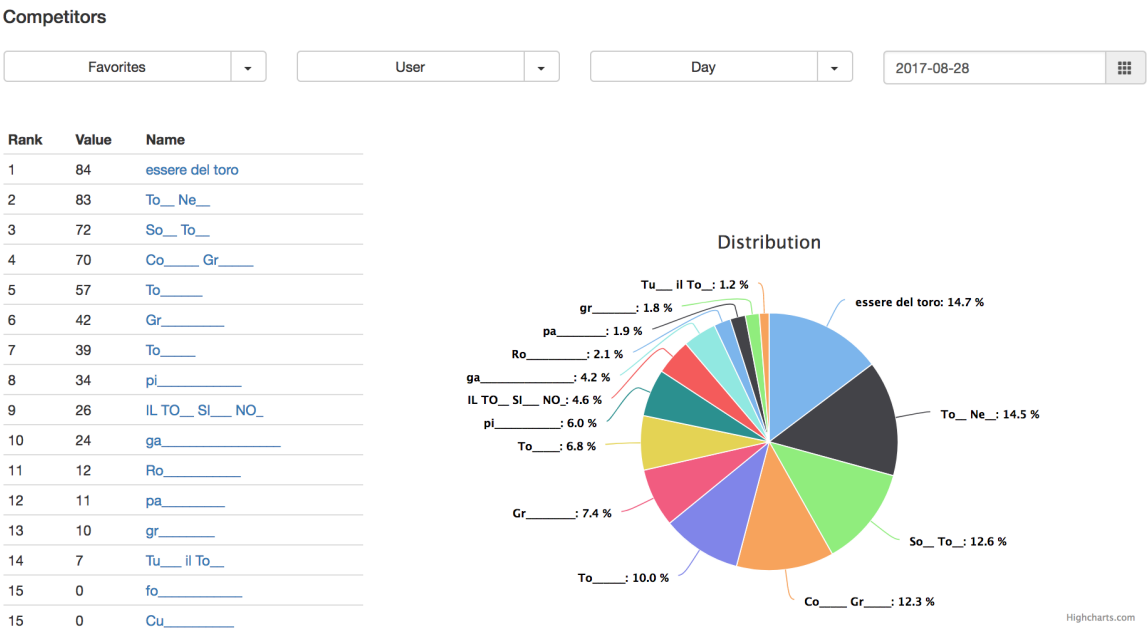

Figure 2.3: Pokedem's competitors comparison

Figure 2.4: Essere del Toro (@esseredeltoro) Twitter account

3 System description

This chapter aims to show the overall structure of the recommendation system that will be outlined in the following chapters. The system will be integrated within the Pokedem application as a support for the social media manager, who based on the recommendations provided by the engine, will have to carry out his/her following strategy directly on Essere del Toro's Twitter account.

The problem that this work addresses is a recommendation problem where:

i. the target/active user is the social media account manager.

ii. the system has to suggest a top-K list of potential followers ranked by the estimated likelihood that they will follow the target account, if approached through the *following action*.

iii. the system may leverage data about users already following the target account.

iv. the feedback about the recommendation is not provided directly by the account manager, but depends on the decision of the recommended user to convert into a follower.

The structure of the system that seeks to overcome this problem can be seen in Figure 3.1[1]. The modules that compose it are described below.

Social media manager. The social media manager is the one who carries out the following actions on those users that are suggested by the recommender engine. This figure played an important role in the creation of the user data set. In fact, as an initial phase, the account manager has started to actively look for potential users to follow, i. e. users considered more likely to follow the target account, in this case @esseredeltoro, if approached through the *following action*. To narrow the search area, a simple heuristic has been adopted, which consisted of identifying the accounts supporting Torino F.C. team, that have consolidated and held a considerable number of followers. The set of followers of these accounts provided a number of potential users from which the following phase could begin.

Although initially there were no particular constraints on the users to follow, over time different heuristics have been defined, including factors like:

- If the number of followers is much higher compared to the target account, the user should not be followed. This is because a rather large account will never be willing to follow a smaller one. An inverse reasoning can be made over the number of friends. An account that follows many other users will be more like to follow the target account.

- A user is considered a supporter of the team, if, in addition to following the official team account, he/she follows other users or pages related to it, or he/she follows several football players belonging to the football club. The use of team-related hashtags is another element that indicates how close he/she is to the team. It often happens that authentic supporters set their header image or even their profile image with a team-related image, such as the stadium or badge. The degree of interest towards a user to follow is therefore based on his/her level of team supporter.

- The activity level of a user is evaluated on the basis of the amount of interactions he/she has generated in the short-term period. Of course, more active users are preferred over those who seldom user their Twitter account.

[1] **Icons**: All the icons used in this scheme can be found on https://www.flaticon.com/ and the rights are reserved to their respective owners.

- The date on which the user joined the platform represents an additional criterion of choice. A user who has just landed on the social network is more likely to explore the various mechanisms offered and thus discover the various accounts that might interest him/her. The target account could be one of these.

Data collection is one of the most cumbersome steps towards the creation of a recommendation system. It is also clear that user search and evaluation of the various parameters are time-consuming activities. That's why the need for a system capable of learning from the hard work done by the account manager and then replicating his actions has arisen. The aim of this thesis is to overcome this gap by creating a recommendation system that leverages machine learning techniques to achieve this goal.

Users data set. This module represents all users who have been followed by the account manager over a particular period of time. Once the following action has been performed, a certain amount of time is given to the followed user so that he/she can follow back the target account. If the user decides to convert within this period of time, then he/she is considered as a positive case, otherwise, he/she is treated as a negative case.

Composite features extractor. The purpose of this module is to encode the information that represents a user. It takes a particular user as input and delegates the job to its sub-modules, whose task is to extract specific features from the set of information forming the user. The whole can be seen as a master-slave management template, where the master part is represented by the composite features extractor, while the slave part is composed of the sub-modules. The last task of the composite features extractor is to build and return as output the vector of features representing the user. This vector is a collection of sub-vectors that have been return from each sub-module. Details of how and which information is extracted together with how it is represented are described in chapter 6.

Features dictionary. The set of features that are extracted from the data set users form a dictionary that will then be used during the encoding phase of new users. These users may have features that are not present in the dictionary. However, in order for them to be compared with those in the data set, it is essential that they are encoded with the same features.

Encoded users data set. This module represents the set of users who have been encoded via the composite features extractor. In this module, users have an adequate representation that can be used by the learning algorithms during the learning phase.

Learning algorithms. This module represents the learning algorithms that have been used to learn the type of users the account manager has followed in the past. The aim is to learn from the actions that have been previously carried out in order to make recommendations at a later stage. Details describing how these algorithms work are presented in chapter 5.

Training model. The output of the learning phase is contained in this module. It consists of a mathematical model, or more simply a mathematical function, that seeks to map users into a discrete set of classes. In this work, a binary classification problem is being addressed, i. e. the set of classes consists of a positive case that represents the user who has converted to a follower, while the negative case is represented by the user who has decided not to follow back the target account.

Profiling Service. This module is a component of the Pokedem application, that can be called with two different types of requests. The first one consists of asking for profiling information about a user. The second one consists of asking for new users to be classified. As time goes by, the number of users collected increases. Before they can be delivered to the classification step, these users pass through a profiling phase, which enriches each user with useful information such as gender, location, level of activity, influence score and favourite team.

Encoded users. This is the dual of the *Encoded users data set* module. The aim is to emphasize the encoding phase during the features extraction stage. This module contains the users coming from the profiling service, not the data set users.

Classification. In this module, users are classified as potential users to be followed, or as unimportant

users. At this stage, all users classified as positive cases, i. e. those who are most likely to follow the target account once they have been approached through the *following action*, are ranked by a confidence score that comes from the predictive model.

User Interface. Within Pokedem there is a dedicated page for the various recommendations that must be carried out by the account manager. Users classified as potential followers are presented in descending order of confidence, i. e. users on whom the classifier has had most confidence are at the top of the list, while those on which the classifier has shown most uncertainty are displayed at the bottom of the list.

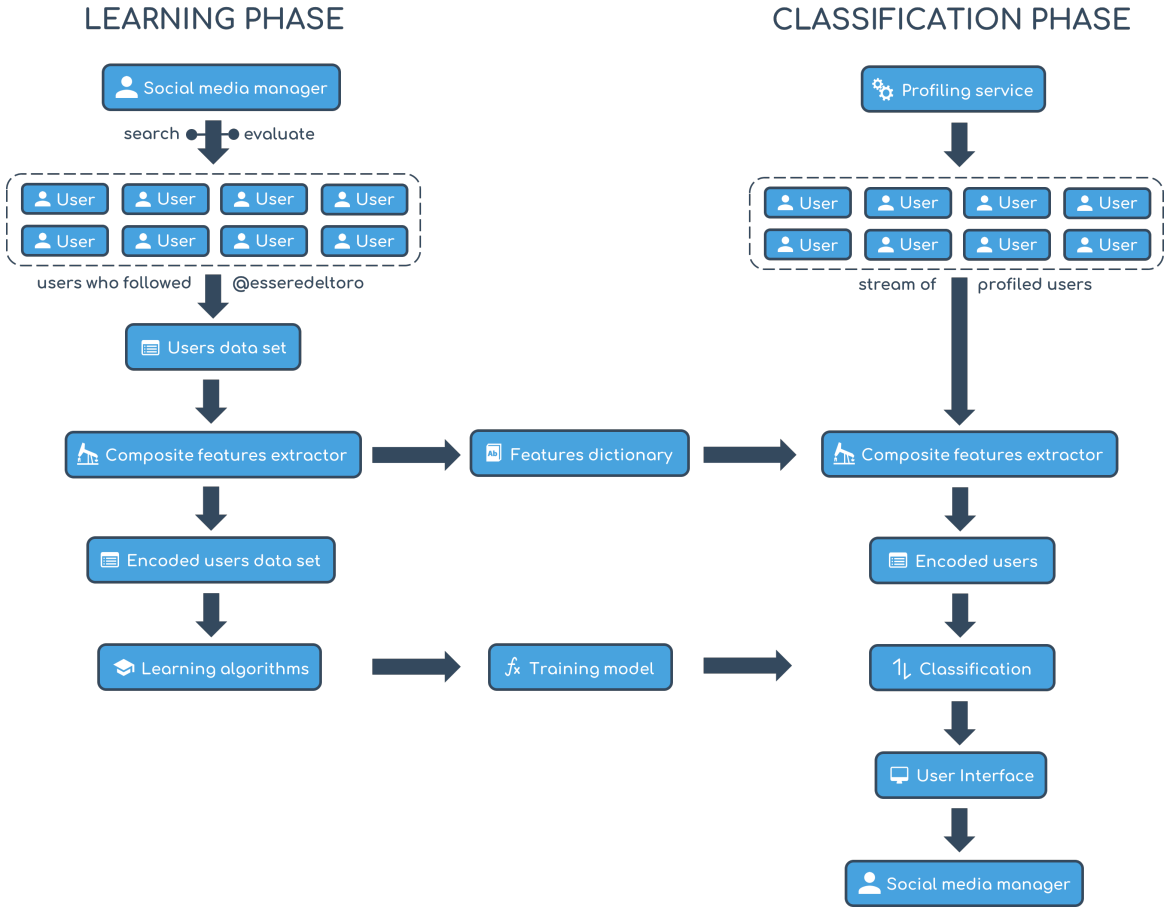

Figure 3.1: Overall structure

4 Recommender systems

The amount and variety of information available on the Web has grown exponentially over the last decade, leading to the so-called information overload problem. The large availability of information, instead of producing a benefit, has had the opposite effect, i. e. people have started to feel somehow overwhelmed.

Nowadays we expect that information, whether textual or visual, is always easy to reach and, in particular, reflects our interests and tastes. This need has given rise to numerous studies, which have led to the creation of systems capable of filtering information and extrapolating only the most relevant one.

Such systems are called **recommender systems**, a combination of software tools and techniques which seek to provide meaningful suggestions to users for items that might interest them. Suggestions are linked to a decision process that the user has to make, such as which items to buy, which music to listen to, or which online newspaper to read, and so on. The term *item* is used to indicate the system output, i. e. what is recommended to the user [6].

The aim of a recommendation system is to deliver some added value to the user. For instance, searching for a certain item can be associated with a time cost that a user is forced to spend. The value that a recommendation system can bring is the reduction of the time needed to find the item combined with the quality of the recommendation, which is something that is inherent to the user's interests.

The characteristics of the items together with the domain in which they are placed influence how a recommender engine will be created. For such an engine to bring satisfactory results, it is necessary to understand all the domain's specific facets, its requirements, challenges, and limitations.

Among the most common domains where recommendation systems are used, the following can be outlined:

- *E-commerce.* The term *recommendation* is often associated with this area, where platforms such as Amazon, Alibaba, and eBay were born. The engines that are used here, seek to create a relationship between products that are sold and buyers in order to choose the most suitable product to recommend to a particular user.
- *Entertainment.* With the ever-greater growth in on-demand content, recommendation systems have become essential. They are mainly used in movie and music related fields. Movie platforms such as Netflix, Hulu, and Amazon Prime Video take full advantage of these systems. Rating prediction algorithms are often used, which aim to recommend to a user the content that is most likely to receive a high rating. Music platforms such as Spotify, Apple Music, and SoundCloud are able to recommend new songs and playlists to the user by analyzing songs that the he/she has listened to in the past.
- *Content.* In recent years newspapers have increasingly digitized their content, migrating from classical paper to the online space. Google News, for instance, is a platform that collects the main news coming from newspapers and then proposes them in a list of articles. By looking at the history and content of articles opened by a user in the past, it is possible to determine which articles to show first in such a way that they are more pertinent to his/her interests.

Recommender systems are a subset of **information filtering systems**. Filtering information means giving the end user, information that is best suited for him/her, i. e., information that is most likely to reflect his/her interests and tastes. However, since each user may have different goals and characteristics from the others, it is not possible to generalize and the system must adapt to each individual

case. A further problem that needs to be addressed, is the decay of information over time. In fact, data describing a user may become obsolete, as his/her interests may change rapidly. A recommender engine must be extremely dynamic and constantly updated, to make accurate suggestions.

In order to be able to customise the recommendations, a recommender system can use a range of information concerning the user. The quantity, quality, and type of information used in combination with the way it is modeled affect the design of the system. An essential component of a recommender system is the **user model**. This term refers to information that constitutes the user, such as his preferences and needs. The way these data are encoded and the implementation details will be presented in chapter 6.

In literature, different categories of recommendation systems have been outlined, which differ on the type of information they use to perform the recommendations [6]. These are:

- *Content-based.* The system that uses content-based filtering takes care of recommending to a specific user, items that are somehow similar to those he previously liked or rated as positive. The item history, as well as how many items he/she has rated, plays an important role in this type of recommendation technique. Thinking of a movie streaming platform, a user could love horror and action movies, and find not very interesting genres such as drama and romance. As time goes by, the user will explore the films offered by the platform and start to rate some movies. Ratings will most probably reflect his preferences, thus allowing a recommendation system to create a model representing him/her. Once the user has been profiled, the system will only try to recommend movies that might really interest him/her, i. e. films that are very likely to be highly rated.

- *Collaborative-filtering.* This is one of the most widely implemented techniques in recommendation systems. Users' tastes are described by the way they have rated items over time. By considering two users Alice and Bob, where Alice represents the user to whom the system must make recommendations, and Bob represents a user very similar to Alice in terms of shared tastes. The items that Bob has positively evaluated in the past and that are not present among Alice's items, become the main candidates for recommendations to be made to Alice. This type of analysis is often called "people-to-people correlation".

- *Demographic.* Demographic factors take an important role in this type of system. The system is designed to exploit the various niches in order to diversify the recommended items. Factors such as the country of origin and language spoken by people, as well as the target age of the users, are some aspects that are taken into account by such a recommender system.

- *Knowledge-based.* A knowledge-based system leverages knowledge of a particular user's preferences and needs and recommends items accordingly. In this type of system, the quality of a recommendation derives from how much it meets the needs of a user.

- *Community-based.* This type of system finds its purpose mainly in social platforms. It leverages the preferences of the target user's friends. The intuition is that people often trust friends' recommendations more than those from anonymous users. Information about the preferences of the target user's friends is collected and used as a metric for recommendations for new items. The recommendation is based on ratings received from friends.

- *Hybrid recommender systems.* A recommendation system is called hybrid when it combines two or more recommendation techniques. The need for a hybrid system stems from the idea that a certain system of recommendation R_1 lacks something that is instead present in the system R_2 and vice versa. This may be the case for a collaborative filtering system (R_1), which is unable to recommend a new item that has never been rated. This problem can be solved with a content-based system (R_2) where the item, described by some attributes, is simply compared with the one that the user has already rated in the past.

5 Learning algorithms

Learning algorithms represent the core of machine learning, a branch of Computer Science that studies the ability of computers and machines, in general, to learn from data, without having to explicitly program them. In order to better understand what is meant by learning algorithms, it is necessary to quote one of the most influential people in this field, Tom M. Mitchell [5], who has provided a formal definition of algorithms studied in the field of machine learning:

*"A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."*

This work focuses on the classification problem. This type of problem is concerned with associating the class to which a given input example belongs to. Classes, often called labels, come from a discrete set of possible values. The problem of classification is tackled here by two learning methods, namely **supervised learning** and **unsupervised learning**.

5.1 Supervised learning

This section describes the functioning of supervised learning, a learning method in which the system is asked to predict an outcome based on some input data. The data provided to the learning problem is known as a *data set*, i. e. a set of m samples:

$$D = \{(x_i, y_i) : i = 1..., m\}$$

Independent variables x_i are represented by numerical tuples, usually in the form of vectors $\vec{x} \in R^n$, and they represent the algorithm input. The dependent variables y_i, which are the output of the algorithm, in a classification problem, can assume values in a discrete set of values C.

On the basis of the examples supplied, the system must provide a map between input data and the answer produced as output. This type of association can be compared to a mathematical function, which when applied to a given input provides a certain output.

$$f_0 : R^n \mapsto C$$

$$f_0 : \vec{x} \mapsto y$$

In literature this function is often called *model*. The latter is unknown a priori, so the only way to proceed is to try to find a new function f_1 which can sufficiently approximate f_0. This function must represent as far as possible a functional dependency of y-values on the corresponding x-values.

However, in order to be useful, this functional dependency must be generalizable, i. e. it must be able to provide correct answers, even when new $\vec{x} \in R^n$ instances are provided, and those instances were not present in the original data set, on which the training phase was carried out.

5.1.1 Support Vector Machines

A Support Vector Machine (SVM) is a supervised learning algorithm that can be used for two types of problems: the classification problem, where inputs are associated with discrete classes, and the regression problem, where inputs are associated with a real value. This section deals with how support vector machines are applied to a classification problem.

An SVM is trained with labelled data and aims to learn a linear model. This type of model deals with finding the optimal hyperplane that separates the data [3]. In a three-dimensional space a hyperplane is represented by a two-dimensional plane, while in a two-dimensional space, a hyperplane is represented by a line.

For a simpler understanding, we will focus on two-dimensional space. Each input example has only two features, x_1 and x_2 , so that it can be easily represented in the Cartesian plane (Figure 5.1).

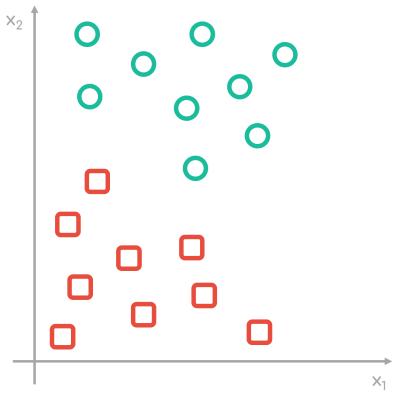

Figure 5.1: Examples in Cartesian plane

The data set D can be seen as:

$$D = \{(x_i, y_i) \mid x_i \in R^n,\ y_i \in \{-1, +1\}\}\ \forall i : i = 1...m$$

Looking at the figure it can be observed that the data is linearly separable. The line separating the two classes can be expressed as:

$$f(x) = \theta^T x + \theta_0 \tag{5.1}$$

where θ is known as the *weight* vector and θ_0 as the bias. By varying the two parameters, it is possible to define infinite number of lines/planes (Figure 5.2).

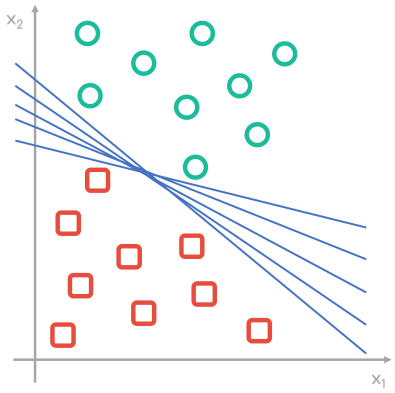

Figure 5.2: Infinite number of planes

However, we are interested in the hyperplane that maximizes the margin between the data set examples (Figure 5.3). Among all hyperplanes, the one chosen takes the name of canonical hyperplane, which is defined by the following equation:

$$|\theta^T x + \theta_0| = 1 \tag{5.2}$$

This equation results in two hyperplanes, where the first hyperplane H_0 is defined by $H_0 : \theta^T x + \theta_0 = 1$ and the second hyperplane H_1 is defined by $H_1 : \theta^T x + \theta_0 = -1$.

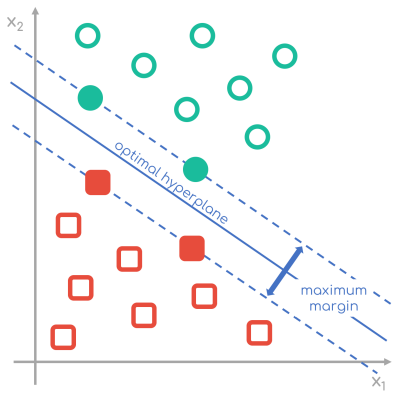

Figure 5.3: Maximum margin

The aim is to find examples within the data set, that are closest to the hyperplane, giving rise to so-called support vectors. To do this, it is necessary to indicate how the distance between a given point x and the hyperplane (θ, θ_0) is calculated:

$$distance(x, (\theta, \theta_0)) = \frac{|\theta^T x + \theta_0|}{||\theta||} = \frac{1}{||\theta||} \tag{5.3}$$

Once the distance has been calculated, the margin can be calculated as follows:

$$M = 2 \cdot distance = 2 \cdot \frac{1}{||\theta||} = \frac{2}{||\theta||} \tag{5.4}$$

It is therefore a matter of solving an optimization problem, where among all possible hyperplanes we will choose the one with the smallest $||\theta||$, as it is the one that maximizes the distance and therefore the margin.

5.2 Unsupervised learning

Due to the unbalanced distribution of classes within the training, where 80% are negative examples and 20% are positive examples, it was decided to try with an unsupervised learning algorithm.

In unsupervised learning, input data is unlabeled. This means that the examples given to the learner do not have any labels. The task of a learning algorithm is to infer an eventual hidden structure or distribution that could be present in the data. This section deals with the one-class classification problem.

A one-class classification problem is often known as unary classification. Compared to the classic problem of binary classification, where training is composed of both positive and negative examples, in the unary classification problem, training is composed only of positive examples.

This kind of approach can be seen as an outlier detector. When a new example is presented to the learner, it is determined whether it is somehow similar to the examples it has already seen. If this is the case, then the provided example can be considered as belonging to the positive class, with which the learning algorithm was trained.

The similarity is measured using the cosine similarity formula. Given two vectors A and B, the similarity is measured as the the cosine of the angle between them. It is defined as follows:

$$cos(\theta) = \frac{A \cdot B}{||A||_2 ||B||_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.5}$$

5.2.1 One Class K-Nearest Neighbor

For a neighbor-identification algorithm to work, three main elements must be provided. The first one is how many neighbors the algorithm has to find. This value is indicated by the k parameter. The second one is a similarity measure between elements. The cosine similarity was used during experiments. Finally, a threshold on the similarity value to identify the best neighbors. This value is indicated by the γ parameter

A first classification attempt has been made using the formula that can be found in [7]. Given a new example x in input that must be classified, the distance D_{xy} to the nearest neighbor y is computed. Next, the distance D_{yz} of the nearest neighbour z of y is computed. Finally, to decide whether the input example x belongs to the target class, the ratio between D_{xy} and D_{yz} is computed. If this result is below a set threshold γ, then the example belongs to the target class, otherwise it is considered as an outlier. The formula can be more easily seen in this way:

$$\frac{D_{xy}}{D_{yz}} < \gamma \tag{5.6}$$

where the distance D is computed as 1 - cosine similarity. This way, as the similarity increases, the distance decreases and vice versa.

However, since this algorithm did not produce great results, it was decided to create a variant. The evaluation of these two versions can be found in chapter 7. The one that has been used in this work can be described by the following steps:

i. A new example is given in input. This example and those in the train are encoded by their own feature vector.

ii. For each training vector, the similarity with the input vector is then computed.

iii. Training vectors are then ordered by maximum similarity with the input vector. After that, the top-K are selected.

iv. If for each example in this top-K list the similarity is above a certain threshold t, then the input vector is considered to belong to the positive class, otherwise it is considered an outlier.

v. If the input vector is classified as positive, it is also possible to determine a score metric by averaging the similarities of the elements in the top-K list. This will be used to make the ranking among all the new classified examples.

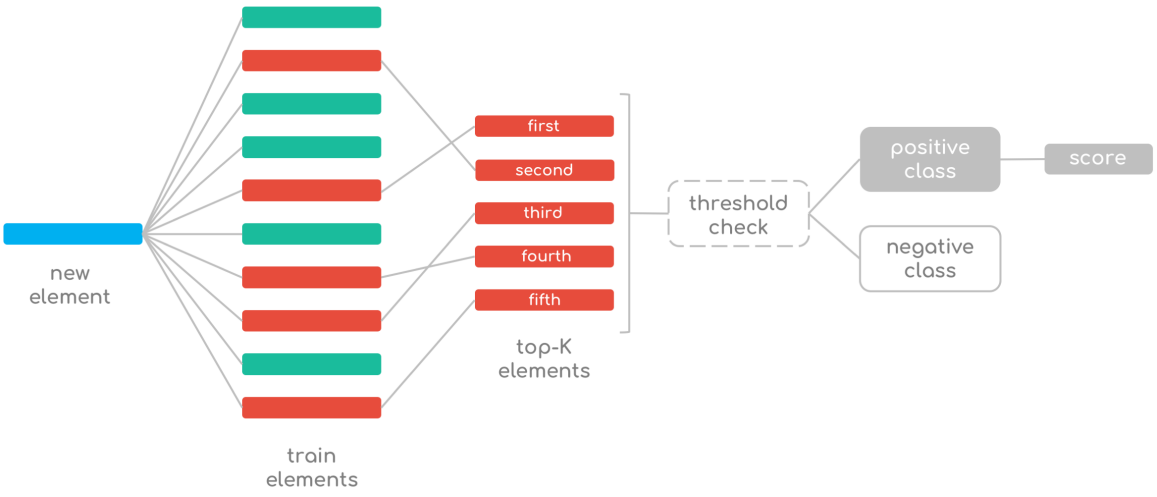

Figure 5.4: One Class K-Nearest Neighbors

6 User modeling

The recommendation system outlined in this work is based on a learning algorithm, which makes use of users' information to learn and then make recommendations. In order for such an algorithm to work properly, it is necessary to encode the data describing a particular user, which consists of extracting relevant information of a user and creating his abstract representation. Prior knowledge is usually necessary in order to choose the most appropriate information to be extracted. In addition, trade-offs can also arise. In fact, extracting too little information can prevent the system from learning the task, while extracting too much of it also means bringing inside information that can be very noisy. This chapter deals with the feature extraction problem, outlining how this thesis has addressed it.

Feature extraction task is often linked to user modeling which consists of the creation of a vector space of users. This means that a user is represented by a vector $\vec{x} \in R^n$, i. e., an n-dimensional vector, where each dimension represents an attribute of which it is composed. In Data Science jargon, attributes are often called features. Each feature in the row vector is described by an index, which indicates its position within the vector, and by a weight that indicates its importance.

Features that make up this vector may belong to two main categories according to whether or not they depend on the domain.

Domain independent features. The following domain-general features are considered, each of them encoded as one or more binary elements in the user vector:

- *Number of followers.*
- *Number of friends.*
- *Number of favourites.*
- *Number of statuses.*
- *Number of lists.*

All these features are originally represented by a number, i. e., a counter. However, numbers must be mapped to a finite number of values. A simple way to do this is by applying the logarithm to this number and round the result down to the nearest whole. This way, a series of equivalence classes can be outlined, which for simplicity are called buckets, where the numbers will fall into. This step can be easily abstracted through a function, which takes the number and base of the logarithm as parameters and returns the corresponding bucket number:

$$f(attribute\ value, logarithm\ base) : log_{base}(attribute\ value) \mapsto corresponding\ bucket\ number \quad (6.1)$$

The attributes listed above are also used to extract some features that take into account the amount of days since a user joined Twitter. These features are indicated as:

- *Number of followers relative to joined year.*
- *Number of friends relative to joined year.*
- *Number of favourites relative to joined year.*
- *Number of statuses relative to joined year.*
- *Number of lists relative to joined year.*

The intuition is that comparing the numerical value and the time taken to reach it can provide useful information about the user under analysis. The feature is extracted by making the ratio between the value of the attribute and the number of days since the user joined Twitter. The result is then discretized using the above function (6.1).

Additional features can be extracted from these numbers. The following features are based on the comparison between the user under analysis, i. e., the user from whom the information is extracted, and the target account. These features are indicated as:

- *Comparison between followers number of the user under analysis and the target account.* From this comparison it is possible to outline two different cases. In the first case the target account has more followers than the user under analysis. It can be empirically verified that users who have more followers than the target account are less likely to follow it. In the second case, the user under analysis has more followers than the target one. The idea behind is that the user who is followed by a larger account in terms of followers, is more likely to follow it back. These two cases are encoded with two specific features. In both cases, however, the ratio between the largest and the smallest number is computed. The logarithm is then applied to this result, to obtain a set of discrete classes.

- *Comparison between friends number of the user under analysis and the target account.* This comparison is very similar to the previous one. It can be inferred that when the user under analysis follows many accounts, i. e. has a lot of friends, it is a kind of user that may be disposed to follow the target account also. When the opposite case happens, i. e. the target account has a higher number of friends compared to the user under analysis, the latter could assume that the target account is a kind of user that makes use of mass following strategy to gather new followers. As a result, his/her willingness to follow back decreases. For the feature encoding part, the same procedure used above is applied here also.

The following features are extracted, taking into account the set of users with whom the target user has had an interaction on the social media. They are:

- *Set of followers.* This set will occupy a subspace within the user vector. Each user in this set will have a corresponding feature that represents him/her in the vector. The assumption behind this extraction is that accounts that are followed by the same set of users are somehow similar.

- *Set of friends.* This subspace consists of the set of users that are followed by the target user. Each vector dimension corresponds to one user within the set. The idea is that it is possible to approximate the interests of a user by means of the users he/she follows.

- *Overlapping between followers and friends.* This set is composed by simply doing the intersection between the two previous sets. In this case, the focus is on those users with whom a two-way relationship has been created, i. e., those who have followed back the user under analysis, or those who have been followed back later by him/her. These users will have a greater weight because, in addition to being included in the above-mentioned sets, they are also part of this set.

- *Set of users who have been retweeted.* Users within this set are extracted from the last 200 retweets of the user under analysis. This feature focuses on the user to whom the retweet was made and not on the content that was posted.

- *Set of users who received a like.* This feature is similar to the previous one. The subspace it occupies is made up of users who have received a like on at least one of their tweets from the user under analysis.

- *Set of mentioned users.* Identical to the above two features except for the interaction mechanism that is being analysed.

There are also two additional features that can be extracted but do not fit into one of the above feature families. These features are:

- *Numerical comparison between followers and friends.* This attribute could have been included in the counter family. However, the way it is extracted is different. Two features can be extracted from it. A first feature that describes which of the two sets has more users, and a second one that aims to quantify this difference. For the latter extraction, two cases can be outlined based on whether the number of followers is larger than friends or vice versa. In both cases, however, the ratio between the largest and the smallest number is made. To cover the various ratios, three main ranges have been defined. The first one provides nine buckets when the ratio is less than 10. The second one provides ten additional buckets to cover the range between 10 and 100. Finally, the last one provides one bucket to contain a ratio that exceeds 100.

- *Description and tweets hashtags.* By analyzing the user's description and last 200 tweets, it is possible to extract a set of hashtags that have been used. Each element of the set is considered as a feature.

The values used for the extraction of the following features come from the Profiling Service:

- *Gender.* This attribute is estimated by taking the first name of the user. This information can be extracted from the screen name on the social media. The name is then compared with two gazetteers of male and female names to determine its gender. As the analysis domain is football, male users are expected to be more likely to follow the target account.

- *Influence score.* This attribute tries to emulate the h-index influence measure, which is mainly used in academia. This measure takes into account the number of tweets posted by a user, and the amount of interactions he has obtained from them.

- *Activity level.* This attribute was initially defined as a simple binary attribute. A user was considered active only if he/she had posted something in the last 6 months. However, this feature has been extended. The activity level is now measured by splitting the initial 6-month period of time into sub-periods, looking at whether the user has posted something in the last hour, last day, last day, last week and so on. The simple binary version has therefore been complemented by a feature subspace that tries to grasp the granularity of the attribute. The activity level of a user is crucial for the following action. The assumption is that by following an active user, the probability that he/she will see the notification and follow back the target account increases.

- *Favourite team.* The way this attribute is derived, is by watching the football teams and players that a user follows. The presence of certain keywords among his posts may also provide an additional clue to the team that a particular user supports. The assumption is that a target account biased towards a team is more likely to be followed by supporters of that team.

- *Favourtie team confidence.* This attribute is closely related to the previous one. The profiling service calculates a confidence value on the team that a user supports. This value varies from 0 to 1. This range has been discretized, i. e. divided into four attributes that take into account sub ranges, such as 0 to 0.25, 0.25 to 0.5 and so on.

- *Location.* The value is encoded to the Italian province. The intuition is that the target account located in a given province is more likely to be followed by people from the same location.

Domain dependent features. The following domain-specific specialized user attributes are considered:

- *Description and tweets favourite team related hashtags.* Hashtags are extracted from the user description and last 200 tweets. A list of recurring hashtags related to @TorinoFC_1906 account was created by observing the tweets posted by supporters. For each extracted hashtag it is checked whether it is present in the list. If it is, it becomes a feature.

- *Description and tweets favourite team related n-grams.* By analyzing the user's description and last 200 tweets, it is possible to check if there are any words related to @TorinoFC_1906. These words were created by hand in an n-gram form. All words related to the team that have been found, are considered as features.

During the features extraction phase an attribute can be encoded into one or more features. This means that an attribute can occupy one or more dimensions in the feature vector. However, in order for a learning algorithm to work as intended, it is required to normalize all the extracted attributes. This way there are no more important attributes than others.

Taking as an example the attribute called "followers of a user", it can be encoded in a vector subspace $\vec{x}$ that can occupy one or more dimensions within the feature vector. The size of this subspace depends on the amount of followers this particular user has. However, it is necessary to rescale this subspace so that it does not weight more than other extracted attributes. The rescaling phase is called **normalization**.

There are several ways to normalize vectors. In this work, normalization is done using one of the most common methods, that is scaling the vector $\vec{x}$ to its unit vector $\hat{x}$. The following formula is used to do this:

$$\hat{x} = \frac{\vec{x}}{||\vec{x}||} = \left(\frac{\vec{x}_1}{||\vec{x}||}, \frac{\vec{x}_2}{||\vec{x}||}, \frac{\vec{x}_3}{||\vec{x}||}, \ \cdots, \frac{\vec{x}_{n-1}}{||\vec{x}||}, \frac{\vec{x}_n}{||\vec{x}||} \right) \tag{6.2}$$

Each component is divided by the magnitude of the feature vector. This way the magnitude $||\hat{x}||$ of the unit vector $\hat{x}$, equals 1.

7 Results and conclusion

This chapter will present the results obtained by the different recommendation systems that have been taken into account. The first part will analyze the performances of the two recommendation systems based on the learning algorithms described in chapter 5. Their performances will be measured on the training data (offline evaluation) and on the experiment carried out via Essere del Toro's Twitter account (online evaluation).

Comparisons will be made with two main baselines, the first one consists of users recommended by Twitter's recommender system, while the second one consists of randomly picked users from the set of classified users, both positive and negative. Finally, the results will also be compared with those obtained before the introduction of the machine learning techniques.

7.1 Learning algorithms: offline and online evaluation

This section presents the results obtained from the two learning algorithms described in chapter 5, that is, the supervised learning algorithm represented by Support Vector Machines (SVM) and the unsupervised learning algorithm based on One Class K-Nearest Neighbor (OC-KNN). The two algorithms will be tested under two conditions, namely:

- *Offline*. Performance is measured on the training data set.
- *Online*. Performance is measured in a real use case where the social media manager follows the algorithm's recommendations and performs them on behalf of @esseredeltoro.

As the classifiers have a binary output, which indicates whether the user received in input is to be recommended or not, a binary class confusion matrix (Figure 7.1) can be used to describe their performance.

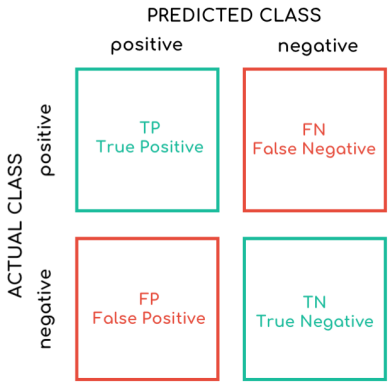

Figure 7.1: Binary class confusion matrix

Different metrics may be extracted from this table of confusion that have been used for performance evaluation. These include:

- *Accuracy*. Amount of examples correctly classified out of total examples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.1}$$

- *Precision*. How many of the examples classified as positive are actually positive.

$$Precision = \frac{TP}{TP + FP} \tag{7.2}$$

- *Recall.* How many of the actual positive examples were recalled by the classifier.

$$Recall = \frac{TP}{TP + FN} \tag{7.3}$$

- *F-score.* A measurement that takes into account both precision and recall to measure the accuracy of a predictive model. The general formula is as follows:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{7.4}$$

  In this work, the β value has been set to 1, leading to the following formula:

$$F_1 = (1 + 1^2) \cdot \frac{precision \cdot recall}{(1^2 \cdot precision) + recall} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{7.5}$$

  The same formula can be seen in terms of the harmonic mean of precision and recall, as follows:

$$F_1 = (2) \cdot \frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{7.6}$$

The **cross-validation** technique was used to evaluate the predictive models, which consists of partitioning the data set into a training set used to train the model and a test set used to evaluate it. To be more specific, a n-fold cross validation has been used in this work, which consists of splitting the examples in the data set into n partitions called folds. On rotation each fold is held as a test set and the remaining n-1 folds are used for model training. One of the advantages of this technique is that all available data is used for both testing and training.

After the cross validation phase, the **micro average method** is applied to obtain the results. This method consists of summing up the true positives (TP), false positives (FP) and false negatives (FN) of each fold and then applying the metrics listed above.

7.1.1 Offline evaluation

Concerning the **SVM classification** part, a library named LIBSVM [1] has been used. It is a very popular library that is not limited only to the academia. It has been included in the code of the recommender system that has been created and has also strongly influenced the format in which the data has been represented. Potential aspects such as scalability and the use of sparse vectors have contributed to the selection of this library.

The parameters that have been used are the default ones, except for:

- *Kernel type.* The **linear kernel** has been used in this work.
- *Probability estimation.* The probability estimation parameter has been enabled. It is used as a score when the predicted class is positive, i. e. the user is recommended.

With regard to the **OC-KNN classification** described in chapter 5, the following parameters have been used:

- *K-Nearest-Neighbors.* The k parameter has been set to 5.
- *Similarity threshold.* The similarity threshold γ has been set to 0.6.

These two parameters were chosen from a set of values that led to the best results in cross-validation with $n = 10$. The analysis table can be seen in Table 7.1.

k	similarity	Accuracy	Precision	Recall	F_1 - Score
5	0.75	0.7558	0.2333	0.0111	0.0213
5	0.70	0.6917	0.2035	0.1011	0.1351
5	0.65	0.5802	0.2619	0.4196	0.3225
5	0.60	0.4913	0.2824	0.7373	0.4084
7	0.75	0.7592	0.2083	0.0039	0.0078
7	0.70	0.7306	0.1839	0.0382	0.0633
7	0.65	0.5279	0.2821	0.6361	0.3909
7	0.60	0.5127	0.2833	0.6839	0.4006
9	0.75	0.7607	0.2000	0.0016	0.0031
9	0.70	0.7306	0.1839	0.0382	0.0633
9	0.65	0.5279	0.2821	0.6361	0.3909
9	0.60	0.5279	0.2821	0.6361	0.3909

Table 7.1: Performances with different values of k and γ

Both SVM and OC-KNN learning algorithms were tested in cross-validation with $n = 10$. Their confusion matrices can be observed in Figure 7.2 and Figure 7.3 respectively. In addition, a summary table shows the values obtained with the various evaluation metrics (Table 7.2).

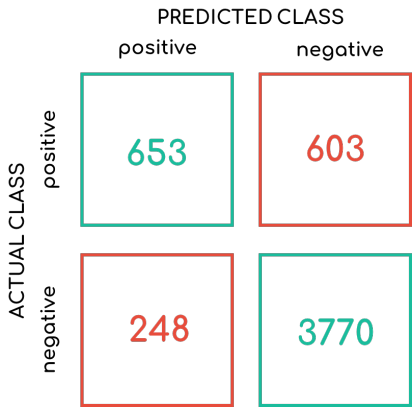

Figure 7.2: SVM confusion matrix

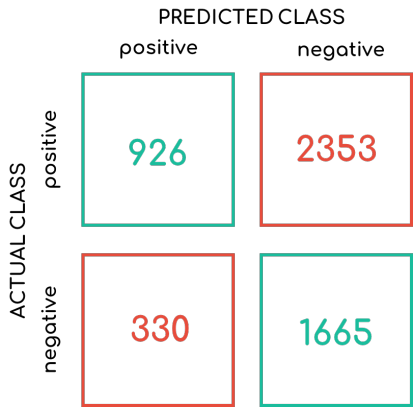

Figure 7.3: OC-KNN confusion matrix

	Accuracy	Precision	Recall	F_1 - Score
SVM	0.8386	0.7247	0.5199	0.6055
OC-KNN	0.4913	0.2824	0.7373	0.4084

Table 7.2: SVM and OC-KNN evaluation metrics results

Regarding the OC-KNN version presented in [7], the comparison with the OC-KNN variant used in this work can be seen in Table 7.3. For clarity, the version indicated in [7] will be indicated with letter A, while the variant used in this thesis will be indicated with letter B.

	Accuracy	Precision	Recall	F_1 - Score
A	0.6356	0.2566	0.2785	0.2675
B	0.4913	0.2824	0.7373	0.4084

Table 7.3: Version A and B of OC-KNN comparison

7.1.2 Online evaluation

This section will present the results obtained by experimenting with @esseredeltoro's Twitter account during the period between 19th of August and the 12th of September.

During the experimentation, it was noticed that the SVM classifier tended to classify a user as a potential follower only when it was very confident about the decision. This has led SVM to extract very few users from a large number of users to classify. For instance, at one stage of the experimentation around 23000 users have been classified. However, the SVM classifier has extracted only 200 users as potential followers. This means that the extraction percentage is less than 1% which is a very low value.

Because of this value, it was decided to try to randomly choose among all users classified by SVM, to verify if those who had been classified as negative, were actually uninteresting users, i. e. users not willing to follow the target account. This has given rise to a hypothetical recommendation system called **Random**, which has been used as a baseline.

It is also worth considering the reason that led to the introduction of an outlier detector such as OC-KNN. The intuition is that the distribution of users' classes within the training set, was not representative of those present in the real world. Figure 7.4 attempts to explain this problem. The main issue is that it is not clear who the false negative users are . The only way to discover them is by following new users, aware that there is still a risk of expanding in the wrong direction. Due to this problem, it was therefore decided to extract only positive examples and apply the one class algorithm described in chapter 5.

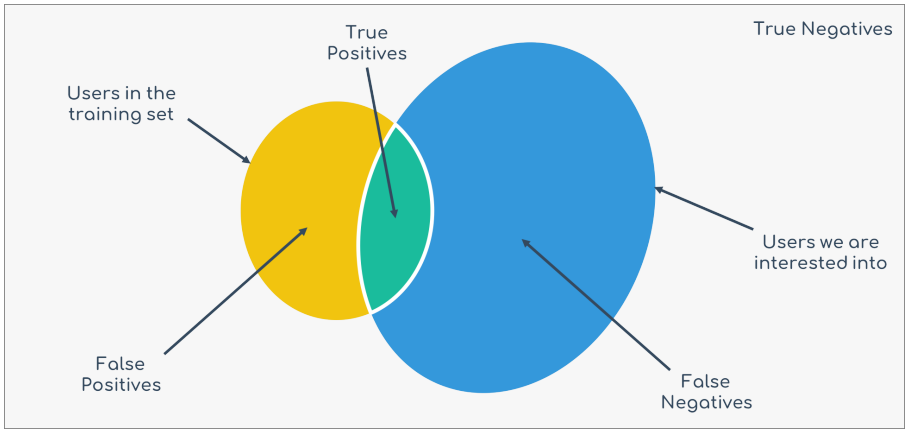

Figure 7.4: Users diagram

In addition to the three algorithms mentioned above, i. e. SVM, Random, and OC-KNN, the recommendation system offered by Twitter has been considered as a baseline. The account manager carried out the experiment with two main recommendation sources:

- *Pokedem UI.* The various strategies based on learning algorithms pass through the application. The list of recommendations is presented to the account manager who, before following them, is required to verify that the recommended users do not contain inappropriate content for a social network.

- *Twitter UI.* The main account page contains a section with users recommended by Twitter. The social media manager follows these recommendations, excluding any promoted accounts or accounts that contain content that is not suitable for a social network.

The following formula is used for the evaluation of all strategies mentioned above.

$$conversion\ rate = \frac{number\ of\ converted\ users}{total\ number\ of\ followed\ users} \tag{7.7}$$

Figure 7.5 shows the graph depicting the followers growth in the period in which the experiments took place. Twitter sets limits on the number of people that can be followed daily. A threshold of 50 users has been set for the daily use of the various strategies.

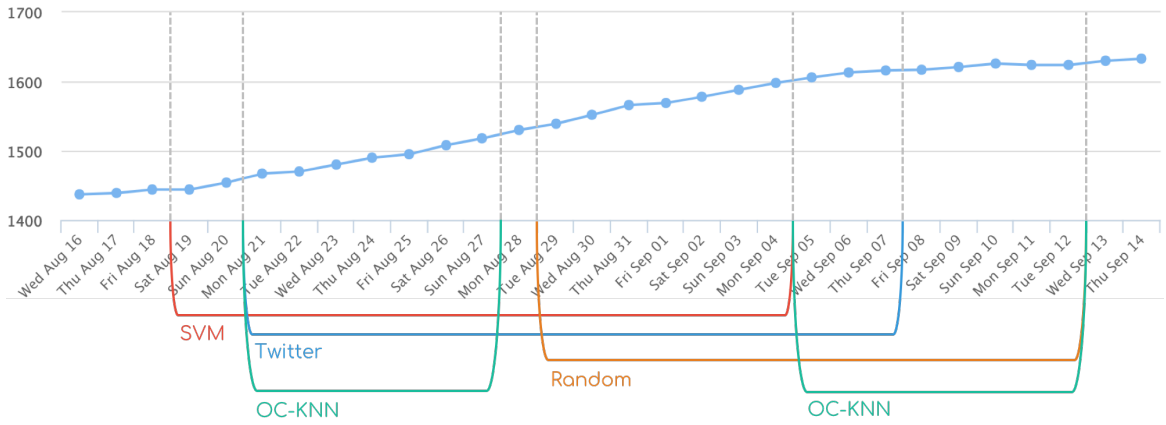

Figure 7.5: Followers growth period 19/08/2017 - 12/09/2017

As can be noticed, the different strategies do not start on the same day because some of them have been defined or reintroduced during the experimentation, when one or more strategies were already running. To overcome this problem and give the same weight to the various strategies, it was decided that each of them should reach 270 followed users.

Having established a total number of users to be followed by each strategy, it is possible to compare them. Figure 7.6 and 7.7 show the performance of the different strategies. The first one shows the number of users gained by each strategy, while the second one shows the conversion rate of each strategy.

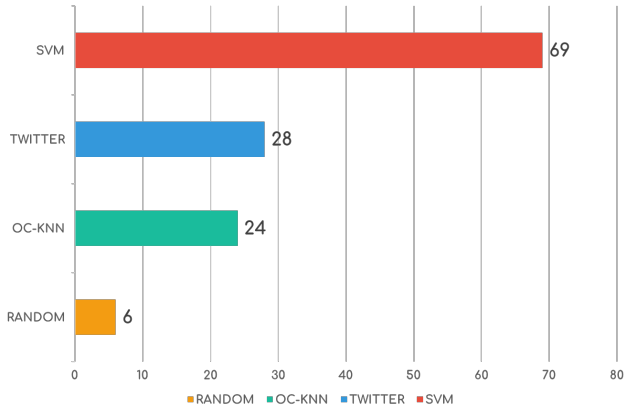

Figure 7.6: Followers gained

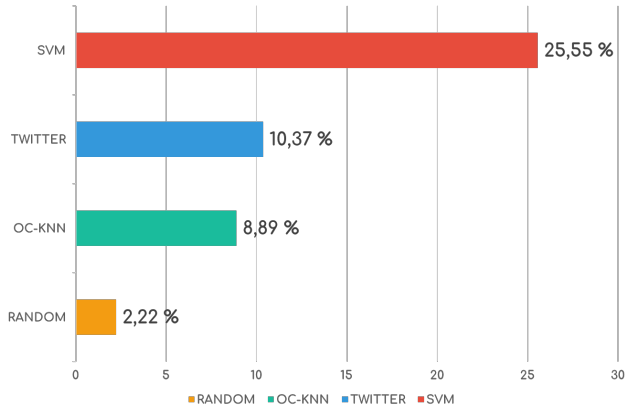

Figure 7.7: Followers conversion rate

The number of users gained by the various strategies does not cover the entire space of users gained during the experimental period. In fact, the total number of users gained in this period is equal to 184, but the strategies mentioned above cover only 127 of them. This difference is made up of users who started to follow @esseredelteltoro without a previous *following action* by the social media manager. It is therefore important to take these users into account as well, especially because they are a considerable proportion as shown in Figure 7.8. It is possible to split this set into two user categories, which are:

- those who follow the target account because they are part of the same domain, i.e football, and are interested in the content that is being posted by the account.
- those who follow the account because they simply apply a mass following strategy and they hope that the target account will follow them back in turn.

Analyzing the retention perspective for these two categories, i. e. the ability to keep them as followers, it is clear that the value of the first category will be higher than the value of the second one. The intuition is that it is more likely that users belonging to the second category will stop following the target account as they have not been followed back.

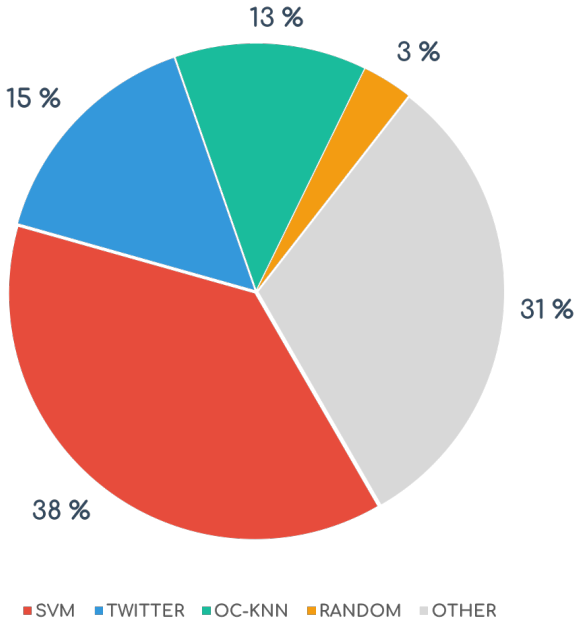

Figure 7.8: All gained followers split by strategies

7.2 Conclusion

The strategy prior the introduction of machine learning techniques was carried out in the first six months of the account's life, managing to obtain a 30% value of conversion rate. During this period, Twitter's recommendation system was used as a baseline for comparisons and obtained a conversion rate value of 8.39%. By combining those results and Figure 7.7 it is possible to notice (Figure 7.9) that the strategy based on manual searching for users and subsequent following by the social media manager obtained 4.45 percentage points more than the best recommendation system defined within this work. Twitter's recommendaer system has also improved slightly.

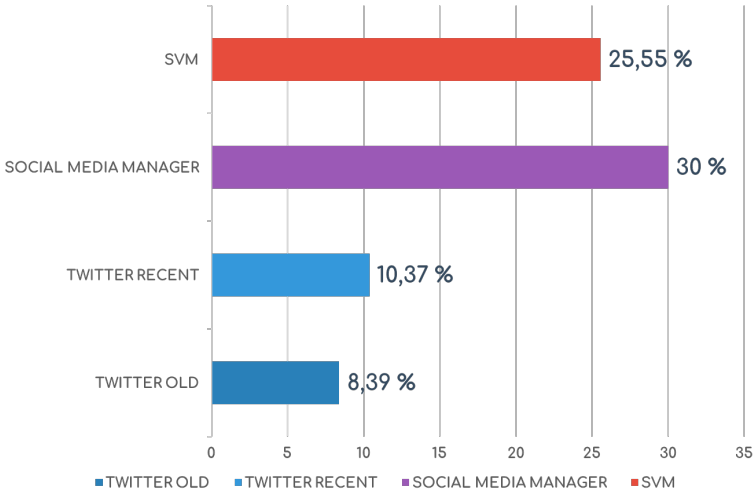

Figure 7.9: Previous strategy followers conversion rate

However, several factors must be taken into account in order to make a proper assessment, including:

- *Temporal space problem.* One of the major drawbacks that weighs heavily on the performance of the various strategies that have been outlined, is the period in which the users have been collected. When the data set was created and users were labeled as positive or negative examples, the status of the target account was significantly different from the current status, as well as the features and information of the users who followed it at that time. As the data set is the core of the learning algorithms, it is essential that data is as accurate as possible. The intuition is that features that nowadays are being extracted from users who have followed @esseredeltoro may have a different value compared to what they were at the moment of user conversion into follower.

  For instance, by considering a user who followed back the target account during the data set creation period, nowadays it may have significantly different attributes compared to those of the past. This means that the reason why the user had followed back the target account at that time was related to his own and @esseredeltoro's specific features, which are not the ones that are being extracted nowadays.

  Unfortunately, it is not possible to retrieve this type of information because no traces have been saved in the database. The only solution would be to restart with a new account and save all essential data, taking advantage of the experience gained from this experiment.

- *Curve of growth and users saturation.* Depending on the domain in which the account is placed, there is an upper bound of users that can be gained. Once an account is close to the hypothetical limit of users that it can gain, growth is also affected.

Given these circumstances, the experiment can be considered successful. The recommendation system that has been outlined in this work, is able to learn the type of users that have been followed by the social media manager and recommend new users to be followed that are similar to those followed in the past.

Bibliography

[1] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. 2013.

[2] Josh Constine. Facebook now has 2 billion monthly users... and responsibility. https://techcrunch.com/2017/06/27/facebook-2-billion-users/, 2017. [Online; accessed 19-August-2017].

[3] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[4] Claudio Giuliano, Francesco Corcoglioniti, Yaroslav Nechaev, and Roberto Zanoli. Pokedem: an automatic social media management application. 2017. DOI: 10.1145/3109859.3109980.

[5] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[6] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer, 2011.

[7] Malik Yousef1, Naim Najami, and Waleed Khalifa. A comparison study between one-class and two-class machine learning for microrna target detection. 2010. DOI: 10.4236/jbise.2010.33033.