PROYECTO FIN DE MÁSTER

Título:	Desarrollo de un Sistema de Análisis de Grafos de Influencia Social para Big Social Data
Título (inglés):	Development of a Social Influence Graph Analytics System for Big Social Data
Autor:	Pablo Álvarez García
Tutor:	Juan Fernando Sánchez Rada
Ponente:	Carlos Ángel Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Tomás Robles Valladares
Vocal:	Mercedes Garijo Ayestarán
Secretario:	Joaquín Salvachua Rodríguez
Suplente:	Francisco González Vidal

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

PROYECTO FIN DE MÁSTER

DEVELOPMENT OF A SOCIAL INFLUENCE GRAPH ANALYTICS SYSTEM FOR BIG SOCIAL DATA

Pablo Álvarez García

Junio de 2016

Resumen

El objetivo de este proyecto es doble: 1) caracterizar el contexto social, 2) crear una plataforma para obtener y analizar el contexto social, es decir, el contexto de los usuarios y su contenido en las redes sociales. En particular, nuestro objetivo es detectar automáticamente posibles influenciadores y evaluar sus capacidades de impacto y relevancia en un tema determinado.

Los principales componentes de la plataforma son los algoritmos que permiten analizar el contexto social. Este trabajo proporciona una visión general del estado del arte en este tipo de algoritmos, así como una descripción de los algoritmos implementados. La plataforma también proporciona una interfaz para cargar los usuarios y tweets objetivo en el sistema.

Además, se ha implementado una interfaz web a través de la cual los usuarios pueden acceder a los resultados del análisis. p.ej. para estudiar la influencia de usuarios o contenidos específicos, así como la evolución de la influencia en el tiempo.

Por último, se presentan las conclusiones extraídas de este trabajo, las posibles líneas de trabajo futuro y continuación del proyecto y los pasos a seguir en términos de desarrollo y el uso de la plataforma.

Palabras clave: Big Data, Social Data, Contexto Social, OrientDB, Influencia Social, Twitter, Python

Abstract

The aim of this project is twofold: 1) to characterize social context, 2) to create a platform to collect and analyse social context, i.e. context of users and content in social media. In particular, we aim to automatically detect possible influencers and assess their relevance and impact capabilities in a given topic.

The main aspects of the platform are the algorithms to analyse the social context. This work provides an overview of the state of the art in such algorithms, as well as a description of the algorithms implemented. The platform also provides an interface to load target users and tweets into the system.

Additionally, we have implemented a web interface through which users can access the results of the analysis. e.g. to study the influence of specific users or content, as well as the evolution of influence over time.

Finally, we present the conclusions drawn from work, the possible lines of continuation of the project and the next steps in terms of development and use of the platform.

Keywords: Big Data, Social Data, Social Context, OrientDB, Social Influence, Twitter, Python

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter provides an introduction to the problem which will be approached in this project. It provides an overview of the objectives and goals of the project and a summary of the structure of the document.

1.1 Context

In recent years, social networks have experienced exponential growth in both number of users and market relevance. Social networks concentrate large amounts of users that ar willing to exchange personal information in order to access the services of the social media site. Large amounts of data are waiting in social networks to be exploited by companies. Because of this, companies have found in them a source of information about their products and a very powerful marketing platform. The main objectives of these companies are the users known as "influencers"; users who have credibility on a particular topic, and their presence and influence in social networks can become an interesting prescriptor for a brand. Through them, companies can track public opinion about your products or your reputation and get good publicity from users with great influence.

The problem is that exploiting these benefits is not a simple task. Companies need to analyse social network in order to find the most influential users or the most relevant content. SNA (Social Network Analysis) comes to answer that need. SNA is a technique of quantitative research for social networks, based on the pillars of graph theory. SNA employs the use of representations of networks of actors, to understand social phenomena. Compared to other analysis techniques, SNA has a strong relational character, and focuses on understanding the interactions between actors in a network and the structure of the network itself. In social networking environments, relationships are even more important than the content itself, as increased connectivity means greater diffusion capacity.

1.2 Goals

This project aims to create a modular platform to collect and analyse data from social networks to automatically detect possible "influencers" and assess their capacity of impact and their relevance in the topic being analysed. To achieve this, the main goals of our project are:

- To define and characterize the concept of Social Context.
- To create a platform that:
  - Is able to extract and process social media information.
  - Is able to characterize users and content in order to be able to use this data in other applications.

- Is modular and extensible.
- Is capable of retrieving, storing and processing large amounts of information.
- Has an interface that allows users to interact with it.

1.3 Structure

In this section we will provide a brief overview of all the chapters of this Master Thesis. It has been structured as follows:

Chapter 1 provides an introduction to the problem which will be approached in this project. It provides an overview of the objectives and goals of the project and a summary of the structure of the document.

Chapter 2 introduces and defines the concept of Social Network Analysis. In this chapter we will explain the state of the art in Social Network Analysis (SNA) and different concepts associated with social networks.

Chapter 3 will define and characterize the concept of Social Context, presenting the most relevant metrics for users and contents. First, we will explain the different components of the Social Context and the metrics associated, particularized for Twitter. Second, we will explain how the influence, one of the most complex metrics, can be obtained. And last, we will present the algorithm that we have implemented to obtain it in this project.

Chapter 4 introduces the requirements analysis for the project. It is important to perform a requirements analysis to make sure the final solution will be adequate for real life applications, and to broaden the variables taken into consideration, making it less likely to miss a key aspect in the design process.

Chapter 5 introduces which technologies have made this project possible. First of all the programming language. Second of all, the technology that allows to virtualize the components and deploy it with a stable configuration. Third, the storage support for the system, where all the social media data and results of the process will be stored. Finally, the technologies that have been used to develop the web interfaces that enable the RESTful API and the interaction with the users.

Chapter 6 describes the architecture of the system. We will explain how the different modules are structured, their roles and how they communicate between them. Finally, we will explain how the user can access the platform to load data or retrieve information.

Chapter 7 describes how the different modules of the system have been implemented

and which technologies have been used.

Chapter 8 will present the conclusions obtained from this project and the lines of future work and improvement.

Finally, the appendix provide useful related information, especially covering the installation and configuration of the tools used in this thesis.

CHAPTER 2

Social Network Analysis

This chapter introduces and defines the concept of Social Network Analysis. In this chapter we will explain the state of the art in Social Network Analysis (SNA) and the different concepts associated with social networks.

2.1 Introduction

We can define Social Context as the set of characteristics of social networks linked to an act of communication, content or user. One of the main parts of our work was to characterize this social context, define it and analyse its components. In order to understand social context, some concepts about social media and social networks are needed.

2.2 Social Media Data

One of the definitions of social media [1] is: "Social Media is the use of electronic and Internet tools for the purpose of sharing and discussing information and experiences with other human beings in more efficient ways."

Traditional media, such as newspapers, television and radio, follow a unidirectional delivery paradigm, from business to consumer. The information is produced from media sources or advertisers and transmitted to media consumers. Different from this traditional way, web technologies are more like consumer to consumer services. They allow users to interact and collaborate with each other in a social media dialogue of user-generated content in a virtual community.

Social media such as blogs, microblogs, discussion forums and multimedia sharing site are increasingly used for users to communicate breaking news, participate in events and connect to each other any time, from anywhere. The social media sites play a very important role in current web applications, because they provide rich information of human interaction and collective behaviour, thus attracting much attention from disciplines including sociology, business, psychology, politics, computer science, economy, etc. [2].

Among the various formats of data exchanged in social media, text plays an important role. The information in most social media sites is stored and shared in text format. For example, microblogging services allow users to post small amounts of text for communicating breaking news, information sharing, and participating in events. This emerging media is the example of how social media has become a powerful communication channel.

This data contains valuable information for different uses. For example, companies which want to know more about their customers or users, can extract data from social media in order to create targeted marketing. This use of the data can bring benefits to the company, but analysing it and extracting useful information from social media is not an easy task. There are many approaches in this matter, with different tools and techniques,

and all of them are grouped under the name of Social Network Analysis (SNA).

2.3 Social Network Analysis

Social network analysis, sometimes also referred to as "structural analysis" [3], is not a formal theory, but rather a broad set of tools, techniques and strategies for investigating social structures. Network studies is a topic that has gained increasing importance in recent years [4], and the fact that the Internet is one largest networks is not unrelated to this. Social network theory directly influences the way researchers nowadays think and formulate ideas on the Web and other network structures, such as those shown in enterprise interactions. Even within the field of sociology, network studies are becoming increasingly important.

The traditional individualistic social theory and data analysis considers individual actors making choices without taking the behaviour of others into consideration. This individualistic approach ignores the social context of the actor. One could say that properties of actors are the prime concern here [5]. In SNA, however, the relationships between actors become the first priority, and individual properties are only secondary. Relational data are the focus of the investigations. It should be pointed out, however, that individual characteristics as well as relational links are necessary in order to fully understand social phenomena.

Another important aspect of SNA is the study of how structural regularities influence actors' behaviour [6]. It is clear that ideas originating in SNA can offer added value to investigations in many disciplines, in particular those mentioned previously. One distinguishes two main forms of SNA: the ego network analysis, and the global network analysis. In "ego" studies the network of one person is analysed. In global network analyses one tries to find all relations between the participants in the network. SNA, although considered here mainly within the field of sociology, is an interdisciplinary technique developed under many influences, the most important ones coming from mathematics and computer science.

To understand the different concepts and functioning of SNA, it is necessary to explain some basic concepts of network structures. In subsection 2.3.2 we will explain some specific concepts associated with networks, and later we will particularize for social networks.

2.3.1 Basics of Network Structure

A network or graph is a set of nodes and edges. The nodes represent elements or individuals in the network, with their properties, and edges represent the relationships between them. Edges can also have their own properties.

There are two main types of graphs, depending on the direction of their edges: directed and undirected. Directed graphs are graphs where the edges have a direction associated with them. An undirected graph is a graph in which edges have no orientation. Most social networks are directed graphs.

2.3.1.1 Network Structures

Beyond nodes and edges, there are some basic structures that are important to know for describing and understanding networks [6]. These include descriptions of nodes, their connections, and their role in the network.

Subnetworks

So far, we have considered the entire graph or network, looking at how many nodes and edges it has and how to describe them. Often, there are parts of the network that are interesting as well. When we are considering a subset of the nodes and edges in a graph, it is called a subnetwork. Some of the simplest subnetworks are singletons. These are nodes that have no edges. While these nodes are not very "social", they are still part of a social network. In fact, it is very common to find singletons in online social networks. Often, these represent people who signed up for an account to access some part of the site other than the social networking features, or people who signed up but never actively participated. We also are interested in small groups of nodes. When looking at two nodes and their relationship, it is called a dyad, and a group of three nodes is called a triad.

Cliques

Groups of nodes of any size have properties that are interesting. One of particular interest is whether or not all nodes in a group are connected to one another. When this happens, it is called a clique. The term is the same as the one we use to refer to, for example, a group of people who are all strongly connected and tend to talk mostly to one another (e.g., "Alice is part of a clique at school"). For a graph or subgraph to be a clique, every node must be connected to every other.

Clusters

We are also interested in clusters of nodes. While there is no strict definition of a cluster like there is for a clique, we can describe properties of clusters using some network measures, like density. There are a variety of methods to automatically identify clusters based on the network structure.

Egocentric Networks

One of the most important types of subgraphs we will consider is the egocentric network. This is a network we pull out by selecting a node and all of its connections until some degree or depth. Often, the central node and its edges are excluded and only the node's neighbours and their connections are considered. This helps make the graph more readable. Egocentric networks can extend out further. Egocentric networks are used to understand nodes and their role in the network.

2.3.1.2 Links, paths and connectedness

A link is a connection between different nodes in the network. This connections and measures of their closeness are important network characteristics that we will explain in this section.

Paths

A path is a series of nodes that can be traversed following edges between them. Often, we are only interested in the shortest path from one node to another. Note that there may be multiple shortest paths between two nodes. Shortest paths will be an important measure we consider in network analysis and are sometimes called geodesic distances.

Connectedness

Paths are used to determine a graph property called connectedness. Two nodes in a graph are called connected if there is a path between them in the network. There does not need to be a direct edge, though that would count. Any path through a series of nodes will work. An entire graph is called connected if all pairs of nodes are connected. In an undirected graph, this is relatively straightforward. A path is found by following edges between nodes. In a directed graph, edges may only go in one direction. Thus, while there may be a set of edges that connect two nodes, those edges may not all point in the right direction. If there are edges that can be followed in the correct direction to find a path between every pair of nodes, the directed graph is called strongly connected. If a path cannot be found between all pairs of nodes using the direction of the edges, but paths can be found if the nodes, edges, and network measures directed edges are treated as undirected, then the graph is called weakly connected. If a graph is not connected, it may have subgraphs that are connected. These are called connected components.

Bridges and hubs

There are two basic concepts that we can use to identify particularly important edges and nodes. The first is a bridge. Intuitively, a bridge is an edge that connects two otherwise separate groups of nodes in the network. Formally, a bridge is an edge that, if removed, will increase the number of connected components in a graph. Hubs are important nodes rather than edges. They do not have a definition as strict as that of a bridge, but the term is used to refer to the most connected nodes in the network.

2.3.2 Social Network Concepts

The term social network has entered common language and is understood to describe circles of friends, acquaintances, colleagues, and so on. We are going to define social network as any graph whose nodes are people and the edges are some kind of relationship between individuals, e.g. a network of cellphone users and their calls. According to Tichy, Noel M. et al. [7], there are three sets of properties of networks are of particular interest:

- Transactional Content: what is exchanged by the social objects. For instance, two employees may exchange information or affect.

- Nature of the Links: this property refers to the strength and qualitative nature of the relation between two social objects.

- Structural Characteristics: this property refers to the overall pattern of relationships between the system's actors. For instance, clustering, network density, and the existence of special nodes in the network are all structural characteristics.

2.3.2.1 Transactional content

What is exchanged when two actors are linked. Four types of transactional contents can be distinguished: (1) exchange of affect (liking, friendship), (2) exchange of influence or power, (3) exchange of information, and (4) exchange of goods or services. Social networks can, then, be developed for each content type. These networks may or may not overlap and an individual's position in the networks may vary. For example, the information exchange network might be decentralized and fully connected, while the influence network might be centralized, with interactions mediated by the formal supervisor

2.3.2.2 Nature of the links

The linkages between pairs of individuals, explained in 2.3.1.2, can be described in terms of several characteristics:

- Intensity: the strength of the relation as indicated by the degree to which individuals honor obligations or forego personal costs to carry out obligations, or by the number of contacts in a unit of time.
- Reciprocity: the degree to which individuals report the same (or similar) intensities with each other for a content area.
- Clarity of expectations: the degree to which individuals agree about appropriate behavior in their relations to one another.
- Multiplexity: individuals have multiple roles, such as worker, husband, community member, and group member. Multiplexity identifies the degree to which a pair is linked by multiple roles. The more role requirements linking one person to another, the stronger the linkage.

2.3.2.3 Structural characteristics

Structural characteristics can be divided into four levels:

- External network: in what ways is the focal unit linked with external domains? Given some external linkage, in what ways is the set of actors linked?
- Total internal network: given a set of actors that make up the network, in what ways are they linked.
- Clusters within the network: areas of the network where actors are more closely linked to each other than they are to the rest of the network are termed clusters. There are various types of clusters: formally prescribed work groups, emergent coalitions, and cliques. A coalition is a temporary alliance of actors who come together for a limited purpose . Cliques are more permanent informal associations and exist for a broader range of purposes. For exampl:, task, social, and career.
- Individuals as special nodes within the network: not all individuals are equally important in social networks. Key nodes exist to link a focal unit to other areas within the organization (liaison), as well as to areas outside the organization (gatekeepers). Individuals can also be uncoupled from the rest of the network (isolates).

2.4 Twitter

Twitter launched as a micro-blogging website in March 2006 which allows users to post status updates of up to 140 characters, also known popularly as tweets [8]. Since its launch, Twitter has amassed a large user base and now has over 310 million users (May, 2016). Twitter allows its users to post short status messages called tweets. Tweets can be posted (tweeted) from various sources which include the Twitter website, Twitter mobile applications as well as several third party applications/websites (after authentication). Users also have the control over the privacy features and they can choose to either make their tweets public which make the tweets visible to any one or make them private which restricts the access to only some users who obtain permission from the user. Users can follow other users on twitter which gives them access to their tweets on their homepage on Twitter.

Twitter allows several other features. It allows users to reply to tweets of other users by clicking on the reply button on the tweet of the user who one wants to reply to. This is a way to say something back in response to a user's tweet. In addition to this, users can also mention other users in their tweets by adding '@' to the username of another user in a tweet. A mention is a way to refer to some other user. Another popular concept of Twitter is retweeting. A retweet is an event of sharing someone else's tweet to our followers. Retweet plays an important part in the dissemination of information on twitter. Users can also add a hashtag in their tweets by adding a '#' sign before relevant keywords. This is used to categorize those tweets to show more easily in twitter search. Very popular hash tags on Twitter become trending topics on Twitter.

An important feature of Twitter that separates it from other social networking sites like Facebook is that the relationship of following and being followed are not necessarily bidirectional. Following someone is equivalent to subscribing to a blog; the follower gets all the status updates of the user that he follows.

One key characteristic that emerges from the network of Twitter users is the Social Graph. A social graph is a graph derived from the connections between the users. These connections can be of many forms. The most straightforward social graph that can be created from twitter is a graph that contains following and being followed relationship among users. There have been several researches [9] [10] [11] focused towards studying these social graphs and finding some features from such graphs. There are a few properties common to many social graphs: the small-world property, power law degree distributions and network transitivity (two users who have a common neighbour are more likely to be connected together rather than with some other user who with whom they don't share a

neighbour).

The social graphs generally contain a clustered structure meaning that certain users form a tightly knit group with very low connectivity between different such groups [8]. These clusters may also contain other similarity features like similar tweets or locations etc. A community in a social graph can be described as a group of vertices that have more edges between them than any other vertex that belongs to other group in the social graph.

CHAPTER 3

Characterizing Social Context

This chapter will define and characterize the concept of Social Context, presenting the most relevant metrics for users and contents. First, we will explain the different components of the Social Context and the metrics associated, particularized for Twitter. Second, we will explain how the influence, one of the most complex metrics, can be obtained. And last, we will present the algorithm that we have implemented to obtained it in this project.

3.1 Overview

In this chapter, we will focus in characterizing social context, explaining its main components, their characteristics and their relationships.

We will use Twitter as the reference social network. All these concepts can be extrapolated to any other microblogging network, and other social media like Facebook, although some restrictions apply. However, other examples tend to suffer of lack of widespread adoption or of a more restrictive interfaces and policies.

There are two main components in social context: users and content. The following subsections present a series of features of both users and content, which we analyse separately. Note that, technically, any information from the social network that is not present in the bare textual content could be considered part of its social context. However, these metrics tend to include richer aspects from the social network, exploiting the graph of relationships and interactions between users and content. Some of these aspects, the more general ones, are already provided by the social network site through its API, such as the number of mentions, favourites or replies. More specific or intensive metrics need to be computed by third parties. This approach is more flexible and powerful, but is obviously limited by the request rate and access restrictions imposed by the social network site.

3.2 User

A user is any entity capable of interacting with other entities in a social network. User information can be split in three groups: profile information, behaviour indicators and information about the user network.

3.2.1 Profile

This group includes all the parameters related with the user's profile. Obtaining these metrics is generally a simple process and can be obtained directly from the profile. The metrics of this group tend to be static or vary slightly in time. The most relevant ones are:

Biography

Biography often contains descriptive information about the user. The contents of the biography of a user can provide very useful when searching for keyword. It allows us to detect or filter effectively certain topic related users, increasing the likelihood that

these users are more influential or popular in those topics.

Account age

The age since the account was created. Account age can influence the popularity or relationships of the user. It is more likely that an older account has a high level of relevance and popularity, as it has been able to improve and develop its relevance and popularity over time, establishing new relationships and increasing its network. Moreover, recent accounts may also be popular, but these tend to be isolated cases. It is a secondary metric and rarely used.

Number of followers

This is the number of followers that a certain user has. This number indicates the amount of users that get updates about the user's activity. This is a good metric to estimate the popularity of the user and the network of users who will receive its messages. On the other side, a high number of followers does not always imply high relevance, so we can find users with more followers than others but with less relevance [12].

Number of friends

This is the number of user that the user is following. This is a less used metric than the number of followers and less influential. Tt does not provide much information by itself, since a user who follows many accounts does not need to have relevance. Furthermore, by combining this metric with the number of followers, we can know how well interconnected a user is with his environment [12].

Associated groups

Social media like Facebook allow the creation of user groups, and users may choose to join one or more groups. In other media Twitter, this may be a list of user lists the user has been added to. The difference between both cases would be that in the former groups are global and users choose their own groups, whereas in the latter each user has their own lists and chooses who to add to them. In other words, one is an active selection and the other is a passive one.

3.2.2 Behaviour

This group includes all the parameters related with the user's behaviour. Obtaining these metrics is not always an straight forward process and could require some calculations. The metrics of this group are less static than the profile attributes. The most relevant ones are:

Recent activity

In an environment where messages have a very short lifespan, is very important to be an active user and perform publications often. Knowing the recent activity of a user allows us to filter active accounts and also adds value to users with a constant activity rate. Moreover, this measure loses its value in isolation, because, as in the case of the number of tweets, a very active user does not have to be an influential one.

Number of tweets

The number of post the user has published since the account was created.This metric gives us an idea of the level of activity of this user over time. A large volume of tweets makes it more likely that a user stands out, but it is not always the case. On the other hand, this metric is not indicative of popularity or relevance by itself. It may be the case of a user with large number of tweets who is not influential or has little impact on the network. A "noisy" user does not have to be influential.

Number of mentions

The number of times that the user has been named and tagged in a publication. The number of mentions that a user has is usually considered to be a good indicator of popularity and influence, since it involves interaction with the rest of the users. A user with a large number of mentions is probably very popular and influential in his network. On the other hand, a user who does not have a large number of mentions is not necessary a low influential one, which makes this metric only add positive value to the relevance of a user [12].

Number of retweets

The amount of times that tweets created by the user are shared. The number of retweets a user receives is a good indication of his popularity and relevance. A high number of retweets implies that the followers of the user usually find interesting or important the content published by the user, which makes him more influential in the network. Besides indicating a high degree of relevance, this metric also has a high correlation with other metrics such as the number of followers, the number of mentions or generated traffic.

3.2.3 Network

This group includes all the parameters related with the user's network, in particular the parameters that act as proxy for the user's influence and his relevance in this network, e.g.

betweenness, connectedness, in/out-degree, etc. explained in Chapter 2. This also includes the relationships with other users and content in the network. The network can be constructed in different ways as well, using either user-to-user or user-to-content interactions. Two examples would be using follower/friend relationships and using the comments network. It is common to combine the metrics above with metrics of the content generated by users and their closest network.

The most important metric in this group is the influence of the user. This metric measures the "amount of attention" that a user receives from the rest of the users. An influential user will receive more attention from other users and his content will be more relevant A good indicator of the relevance of a user is the influence of the users in his network. An influential user generally surrounds and connects with other influential users. Therefore, better connected users tend to be more influential [12].

3.3 Content: Tweets

Content in social media is the information created and shared by the users. Tweets are the default content in Twitter. Content information can be split in three groups: attributes, propagation and network.

3.3.1 Attributes

Content attributes are data associated with the content being shared:

Creator

The same content shared by different people has a very different impact on the network. Oftentimes, the audience only depends on the original creator, their network of followers. These factors greatly affect user reactions.

Keywords

Keywords can be used to group different topics together, and to extract concepts and entities from text. When analysing the relevance of a tweet on a certain topic, keyword research is very important. On the other hand, this method not always works perfectly because it is common that a tweet related to a topic does not include keywords or use synonyms. This can cause that we may filter some valuable tweets in our search [13].

Tweet emotion/sentiment

The emotion or sentiment expressed in content can affect other people's reactions. Often the sentiment of the tweet affects directly to its relevance. The problem with this metric is that it is dependent of the subject and context [13].

Tweet date

Content is ephemeral, its value or potential rapidly degrades with time. In other cases, such as social news, the content is tied to specific real life events or contexts. An old tweet will have little to no impact today.

Topics

As with keywords, topics are very useful to group content. Moreover, it is a crucial part for emotion recognition, which is domain dependent.

Media

Textual content may also have multimedia attached. For instance, a Tweet with a picture.

External links

Tweets often have links to content from outside the social network. For instance, a link to a newspaper.

3.3.2 Propagation

This kind of information characterises how content is shared and received by other users. Most of these features are dynamic, and we can analyse both the value at a specific point in time or the evolution of this value. Some examples would be: time to achieve a certain amount of reshares, editions to the content over time or evolution of favourites over time. The most important metrics in this group are:

Number of retweets

Like in the case of a user, a tweet with a large number of retweets has great popularity and indicates that the users that retweeted it find it interesting, making it a good indicator of influence.

Number of favourites

The amount of times that a tweet has been marked as "favourite" by a user. A

high favourite count indicate that a tweet is very popular. On the other hand, a low favourite count does not imply low popularity. In general, people prefer to use a retweet to indicate interest in a tweet, rather than favourite. It is a secondary metric and it can be replaced by the number of retweets in most cases.

Number of replies

Also known as comments in some networks, this feature measures direct reactions from other users. In contrast to favourites and reshares, which have an inherently positive charge, replies may be negative and include criticism.

Number of views

In media like Facebook or Youtube, it is possible to determine how many people have accessed the content. This feature is important when combined with others, as it is a proxy measure of neutral reactions.

3.3.3 Network

How content is related to other content and users in a social network often offers valuable insights about the content. The interactions between users and content forms a graph or network that can be studied. This concepts have been already introduced in Chapter 2. This information can be used to detect social media phenomena, such as cascades or memes. These are some of the relationships that can be used, alone or in combination, to form this network: Creation/Modification/Deletion (User - Content); Reshare/Favourite/View (User - Content); and Mention/Reply (Content- Content).

The most important metrics in this group is the tweet influence.This metric measures the “amount of attention” that a tweet receives from the users. Influential tweets will receive more attention and will be more likely to be shared. When determining the relevance of a tweet, it is really important to take into account the influence of the user who posts it, the influence of the users that read the tweet and the influence of the users that retweet it. A tweet published by an influential user and retweeted by influential users will generally be a very influential one.

3.4 User and Tweet Influence

From all of the metrics explained above, the influence of a user or a tweet is one of the most complex. It can not be obtained directly and it is necessary to estimate it from other

metrics. Nowadays there is no clear way to calculate it, but there are different approaches and different companies are using their own methods to calculate it. In the next sections we are going to study the different approaches available.

We have conducted a small study to find out what are the principal applications that use SNA to measure influence on Twitter. We will present a list of the principal solutions, with the metrics they use and the importance of each in their calculations.

3.4.1 Social Authority

It is a solution from Moz company, dedicated to developing applications to improve online marketing companies [14]. It is a tool to calculate the influence of Twitter users, giving them a score called Social Authority. To calculate this score, they employ a private algorithm, but they have published the metrics they use. Their main metrics are the number of retweets of the user, the retweet rate of the published tweets by the user, the user's recent activity and the age of the tweets. As secondary metrics, to adjust the values, they use the number of followers and friends of the users, the number of mentions, etc.

3.4.2 Follower Wonk

This is another product of Moz company and it allows to analyse the network of a user to understand better their followers and improve his marketing strategy [15]. This application extracts basic information about each user (account age, number of tweets, tweets per week, etc) in addition to an analysis of the content of their biography searching for keywords. This search allow you to search the company target audience to expand or improve their relations in the social network.

3.4.3 Klout Score

It is a measurement tool of influence with great acceptance and it is for many the "standard ranking of the influence" [16]. This tool measure the influence of a user on a scale from 0 to 100. The score is calculated by combining the activities of the user in up to 12 different social networks. To do this, the company uses more than 30 algorithms that are not publicly available and they update the scores daily. To calculate the score, they combine three factors that they have called "real scope", "amplification" and "impact on the network":

- Real scope: It is not about the number of followers, but the number of users on which

the user exerts influence. To determine it, they analyse how people who follow the user interact with him and with his content.

- Amplification; This metric means how much a user influences over the people in his network. They measure it considering how the messages of the user lead to conversation or are replicated. Each of these actions are considered a sign of authority and quality of the user content.

- Impact on the network: This measure calculates the influence of the followers of the user who are within his true scope. In other words, it measures the "quality" of the followers of the user.

3.4.4 PeerIndex

It is defined as a tool for measuring and understanding the "social capital" that a user has achieved in the online environment. In this case, they focus more on the concept of notion of authority and reputation rather than the influence. PeerIndex set the following 8 large topics (subdivided into 8000 more specific subtopics):

- Art, media and entertainment
- Technology and Internet
- Science and environment
- Health
- Sports
- Current topics, politics and society
- Business and economy
- Leisure and lifestyle

Within each topic, Peerindex rate its users from 0 to 100 based on three components:

- Authority: "The measure of confidence". It is the way in which the rest of the people rely on the recommendations and opinions of the user evaluated, both generally and on specific topics.

- Audience: it is equivalent to the parameters of "real scope" and "amplification" of Klout. It measures the impact of user actions over his followers and the way in which they link his content.

- Activity: Represents the update frequency of the user status in relation to the rest of the PeerIndex community around a topic. This update frequency is valued according to the nature of the topic, where it is not always positive a higher level of activity.

The information needed to score each user is obtained through a semantic analysis of the content of the URLs shared by the user in his tweets, allowing to identify the five most important topics for him. Again, the mechanisms used to calculate the score are private for reasons of competitiveness.

3.4.5 Twitter Rank Algorithm

After studying how the companies calculate the relevance and influence of tweets and users, we have studied the algorithm proposed by Noro, T., Ru, F., Xiao, F. and Tokuda, T. in "Twitter user rank using keyword search" [17] that it is currently in development [18]. This is an algorithm allows to calculate the influence of tweets and users that are the result of a search, assuming that their influence is a value linked to the context.

During the development of the platform, we have made an implementation of this algorithm. Below, we will explain how this algorithm works, what information uses and the data obtained. It will be divided in two sections: User relevance and Tweet relevance.

3.4.5.1 User relevance

In their work, they describe a method for finding relevant users im the target topic (called topic-related users). In their research, they have the following assumptions about the relevant users.

1. Good topic-related users usually post valuable tweets on the target topic.

2. The valuable tweets on the topic draw the attention of many users.

3. Each user pays attention to the tweets the user retweets or replies to.

4. Each user also pays attention to the tweets posted by the user's friends (followees).

Based on these assumptions, we define the user relevance score of each user u as follows:

$$UserRel(u) = TR(u)^{w_r} \times UI(u)^{w_i} \times FR(u)^{w_f} \tag{3.1}$$

$TR(u)$, $UI(u)$ and $FR(u)$ are respectively "tweet rate (TR) score", "user influence (UI) score", and "follow relation (FR) score" of user u ranging between 0 and 1. w_r, w_i, and w_f are non- negative values where the sum of the values is equal to 1. The TR score is based on the tweet frequency, and reflects the first assumption. The UI score is based on the tweet, retweet, and reply activities and the follow relation, and reflects the second, third, and fourth assumptions. The FR score is based on the follow relation, and reflects the second and fourth assumptions.

Tweet Rate (TR) Score

The topic-related users post many tweets relevant to the target topic. However, there are some users who post many relevant tweets and much more irrelevant tweets (some users post hundreds of tweets on various topics every day). In order to exclude such users, we consider the tweet rate instead of the number of each user's tweets searched. In calculation of the TR score, we count not only each user's original tweets but also retweets as the user's own tweets. Some of the topic-related users usually retweet tweets relevant to the topic originally posted by others, which means they play a role of "filter" searching for valuable relevant tweets and sharing them with their followers.

This metric measures the proportion of tweets related to the topic that a user posts or retweets. We use the following formula to calculate it:

$$TR(u) = \frac{|t|t\epsilon T \wedge t.user.id{=}u.id|}{|Total(u))|} \tag{3.2}$$

Where t is a tweet posted by the user that is relevant to the topic and $Total(u)$ are the total amount of tweets posted by the user u during the topic search duration.

User Influence (UI) Score

The basic idea is:

1. Users who post many tweets on the target topic paid attention to by many users are good topic-related users.
2. Tweets of the good topic-related users are often paid attention to by other good topic-related users.

How much each tweet is paid attention to by others is measured according to the retweet and reply activities and the follow relation. Based on this idea, we define not only the UI score of each user but also "tweet influence (TI) score" of each tweet. The UI score of each user is calculated using the TI score of the user's tweets and retweets, and the TI score of each tweet is calculated using the UI score of users who pay attention to the tweet. The UI score and the TI score are defined as follows:

$$u = B_t^T t \qquad t = B_a^T u \tag{3.3}$$

u and t indicate a column vector of the UI score and a column vector of the TI score respectively. B_t is the tweet-to-user relation matrix based on what (tweet) is posted or retweeted by whom (user), and B_a is the user-to-tweet relation matrix based on who pays attention to what. To derive the two relation matrices B_t and B_a, we create a reference graph consisting of user nodes, tweet nodes, and directed edges each of which connects a user node and a tweet node, called "tweet activity relation graph". The tweet activity relation graph is represented as combination of three adjacency matrices A_t, A_r, and A_s.

$$A_t(t_i, u_j) = \begin{cases} 1 & \text{if } t_i \text{ is posted/retweeted by } u_j \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

$$A_r(u_j, t_i) = \begin{cases} 1 & \text{if } u_j \text{ retweets/replies to } t_i \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

$$u_0 = (\frac{1}{|U|}, \frac{1}{|U|}, ..., \frac{1}{|U|}) \qquad t_0 = (\frac{1}{|T|}, \frac{1}{|T|}, ..., \frac{1}{|T|}) \qquad k = 1 \tag{3.6}$$

$$Repeat \qquad t_k = B_a^T u_{k-1} \qquad u_k = B_t^T t_k \tag{3.7}$$

$$k = k + 1 \qquad until \quad k = 10000 \tag{3.8}$$

$$A_s(u_j, t_i) = \begin{cases} 1 & \text{if } u_j \text{ follows at least 1 user who post/retweets } t_i \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

t_i and u_j indicates the i-th tweet and the j-th user respectively. At represents what is posted or retweeted by whom, and A_r and A_s represent who retweets or replies

to what and who sees what respectively. It is natural for users to reply if they are mentioned by others, and some users often retweet tweets mentioning themselves. These activities are ignored in creation of the tweet activity relation graph since they do not always depend on topics. Activities replying to themselves are also ignored. The adjacency matrices are transformed into the two relation matrices B_t and B_a as follows:

$$B_t(t_i, u_j) = \frac{A_t(t_i, u_j)}{\sum_k A_t(t_i, u_k)} \tag{3.10}$$

$$B_a(u_j, t_i) = \begin{cases} \frac{A_r(u_j,t_i)}{\sum_k A_r(u_j,t_k)}(1-d) + \frac{A_s(u_j,t_i)}{\sum_k A_s(u_j,t_k)}d & \text{if } \sum_k A_r(u_j, t_k) \neq 0 \\ \frac{A_s(u_j,t_i)}{\sum_k A_s(u_j,t_k)} & \text{otherwise} \end{cases} \tag{3.11}$$

d is a damping factor of $0 < d < 1$. The matrix B_a reflects the third and the forth assumptions about the topic-related users. Each user pay attention to tweets the user retweets or replies to, and the user also watches all tweets at a certain rate of d regardless of the user's activities. Tweets posted or retweeted by the user's friends are more likely to be seen than the other tweets. The UI score and the TI score are calculated using the power iteration method as shown in Figure 2. u_k and t_k indicate the UI score and the TI score at the k-th iteration respectively. U and T are a set of user nodes and a set of tweet nodes in the tweet activity relation graph. Lastly the UI score is normalized so that the largest value should be 1.

$$UI(u_j) = \frac{u(j)}{max_k u(k)} \tag{3.12}$$

Follow Relation (FR) Score

The FR score is calculated based on the follow relation using PageRank [19]. A reference graph consisting of user nodes and directed edges each of which connects two of the user nodes, called "follow relation graph", is created from the follow relation. The graph is represented as the following adjacency matrix:

$$A_f(u_i, u_j) = \begin{cases} 1 & \text{if } u_i \text{ follows } u_j \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

$$B_f(u_i, u_j) = \begin{cases} \frac{A_f(u_i,t_j)}{\sum_k A_f(u_i,u_k)}(1-d) + \frac{d}{|U|} & \text{if } \sum_k A_f(u_i, u_k) \neq 0 \\ \frac{1}{|U|} & \text{otherwise} \end{cases} \tag{3.14}$$

$$f = B_f^T f \tag{3.15}$$

u_i and u_j indicates the i-th user and the j-th user respectively, and d is a damping factor. U is a set of user nodes in the follow relation graph, and f is a column vector of the FR score. The FR score is normalized so that the largest value should be 1.

$$FR(u_i) = \frac{f(i)}{max_k f(k)} \tag{3.16}$$

3.4.6 Tweet relevance

They describe a method for finding relevant tweets to the target topic. The method consists of two parts: calculation of the Voice score and the Impact score of each user based on the user activity, and ranking tweets based on who posted, retweeted, or replied to each of the tweets in the system.

Voice and Impact Score Calculation

In order to judge the relevance of each tweet to the target topic, we have the following assumptions about tweets relevant to the topic (relevant tweets).

1. The relevant tweets are posted or retweeted by the topic-related users.
2. The relevant tweets are paid attention to (retweeted or replied to) by many topic-related users.
3. Tweets posted, retweeted, or replied to by good topic-related users are more relevant to the topic.

These assumptions are close to the idea of the TI score defined above. Tweets with high TI score are paid attention to by users with high UI score, and users with high UI score post and retweet tweets with high TI score. However, we need tweets posted in a certain time period and the follow relation among the users appearing in the tweets to calculate the TI score, which means that it is difficult to apply the calculation to tweet data streams since the score of newly-arrived tweets cannot be calculated on the fly. Instead, we estimate the TI score of newly-arrived tweets based on the TI score and the UI score derived from the past tweet data.

We have two ideas for estimating the score of a newly-arrived tweet. According to the definition of the TI score, the score of the newly-arrived tweet can be estimated from the UI score of the users who retweeted or replied to the tweet. The other idea of the score estimation is considering the TI score assigned to the past tweets and

retweets of the users who posted or retweeted the newly-arrived tweet. For the score estimation, we define two kinds of score of each user, called "Impact" and "Voice". The Impact score is used for the estimation based on the first idea, and the Voice score is used for the estimation based on the second idea. The Impact score of user u (appearing in the tweets collected in the search) is defined as follows:

$$Impact(u) = \begin{cases} \frac{UI(u)}{|Relate(u)|+\sigma_i} \times (1-d) + \frac{UI(u)}{|T|} \times d & \text{if } |Relate(u)| > 0 \\ \frac{UI(u)}{|T|} & \text{otherwise} \end{cases} \tag{3.17}$$

$Relate(u)$ indicates a set of tweets the user u retweeted or replied to, and T indicates a set of all tweets obtained in the search (i.e. a set of tweet nodes in the tweet activity relation graph). σ_i is a smoothing parameter ($i \geq 0$) and d is the damping factor used in Ba equation. Unlike the case of the user relevance score calculation, we use the unnormalized UI score for the Impact score calculation. Some users frequently retweet other users' tweets without taking a moment to read them, and such users have little impact on other users' tweets. The definition of the Impact score reflects this idea. In the definition of the Voice score of user u (appearing in the tweets collected in the search), we consider the score for the user's original tweets ($Voice_t$) and the score for the user's retweets ($Voice_r$) separately. This is because some users post valuable original tweets related to the target topic and some other users search for and retweet valuable tweets posted by other users (some users do both).

$$Voice_t(u) = \frac{1}{|Tweet(u)| + \sigma_v} \sum_{t \epsilon Tweet(u)} TI(t) \tag{3.18}$$

$$Voice_r(u) = \frac{1}{|Retweet(u)| + \sigma_v} \sum_{t \epsilon Retweet(u)} TI(t) \tag{3.19}$$

$Tweet(u)$ and $Retweet(u)$ indicate a set of the user u's original tweets and a set of the user's retweets respectively. $Poster(t)$ is a set of users who posted or retweeted the tweet t. σ_v is a smoothing parameter ($\sigma_v \geq 0$). The $Voice_t$ score is not defined here if the user's original tweet set is empty, and the $Voice_r$ score is not defined if the user's retweet set is empty. We will describe how to deal with the users with the undefined Voice score in the next subsection. The Voice score is the average TI score for the user's original tweets or retweets. The TI score of tweets retweeted or replied to by many users is likely to be high, which means the $Voice_r$ score of users who just retweet only a few tweets retweeted or replied to by many users may be higher than expected.

Tweet Ranking

Previously, we collect tweets posted or retweeted by the topic-related users and retweets of the topic-related users' tweets, calculate the tweet relevance score of each tweet from the Voice score and the Impact score, then select the top-M tweets ranked by the tweet relevance score as the tweets relevant to the target topic. The tweet relevance score is defined separately according to the version of the Voice score. We use the following formula:

$$TweetRel(t) = \alpha \times VR(t) + (1 - \alpha) \times IR(t) \tag{3.20}$$

$$VR(t) = Voice(t.user) \tag{3.21}$$

$$IR(t) = \sum_{u \epsilon Related(t)} Impact(u) \tag{3.22}$$

$VR(t)$ and $IR(t)$ are respectively "Voice-based Relevance (VR) score" and "Impact-based Relevance (IR) score". Alpha ranges between 0 and 1. $Voice(u)$ indicates the Voice score of the user u ($Voice_t$ if u is the original poster of the tweet t, and $Voice_r$ if u retweeted t). $Poster(t)$ and $Related(t)$ are a set of users who posted or retweeted the tweet t and a set of users who retweeted or replied to the tweet t respectively. $t.user$ is the original poster of the tweet t. The VR score and the IR score can be considered as the estimated TI score calculated from the Voice score and the estimated TI score calculated from the Impact score respectively.

CHAPTER 4

Requirements Analysis

This chapter introduces the requirements analysis for the project. It is important to perform a requirements analysis to make sure the final solution will be adequate for real life applications, and to broaden the variables taken into consideration, making it less likely to miss a key aspect in the design process.

4.1 Overview

In this chapter, we are going to present some of the possible use cases of our system, in order to define the requirements for our platform. This use cases will help us finding the necessities of our potential users.

4.2 Use Cases

Influence analysis has many possible uses and interests nowadays, varying from a user who wants to know his influence or his "score", to a company which wants to analyse the influence of his messages, marketing or products. We can set several use cases depending on the future user of the system. In the following sections, five different use cases will be presented.

4.2.1 UC1: Brand Monitoring

Enterprise SA. is an average marketing company. They have decided that they need a way to get feedback online of their products in the market. Specifically, they want to know what people are saying about their products in Twitter. They decide to use SCANER for this analysis. They begin to collect tweets with the keywords of their products and load them into the system through the API. SCANER will extract the information about the users and tweets related. It will give a ranking of the most relevant positive and negative tweets in the keyword search, with the corresponding emotion and the information of the user. Now, Enterprise SA. can have a list of the most influential tweets about their products, meaning that they can know what the most influential users think about their products and what are the most influential opinions. This will help Enterprise SA. to improve them.

4.2.2 UC2: Emotion Analysis

A set of researchers want to analyse the spread of emotions in Twitter and they decide to use SCANER. They load a set of tweets from an important event into the platform. SCANER will extract the information about related users and tweets. It will annotate the emotions of the different tweets, and their relationships with the other tweets. With this information, the developers can study the original tweets with their emotions, and how this tweets spread through the network.

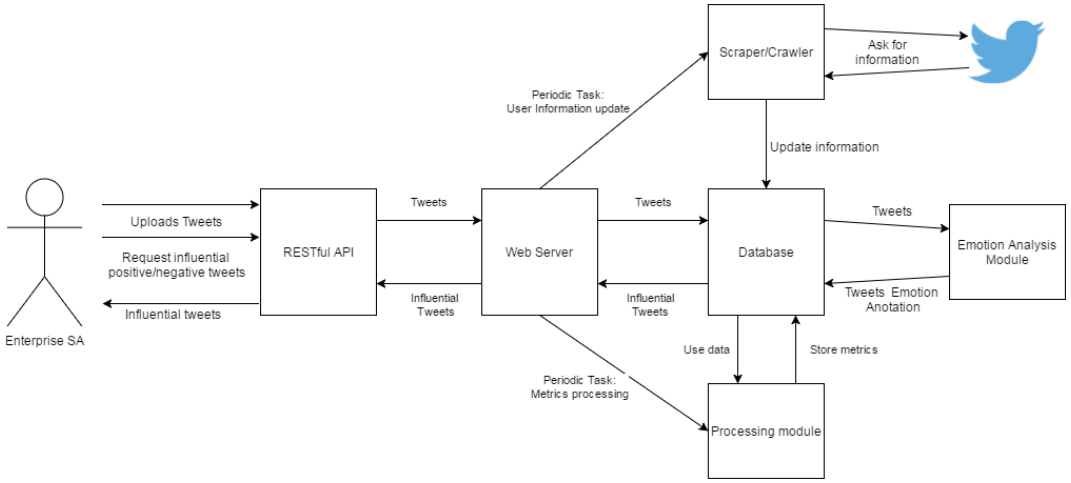

Figure 4.1: UC1: Brand Monitoring

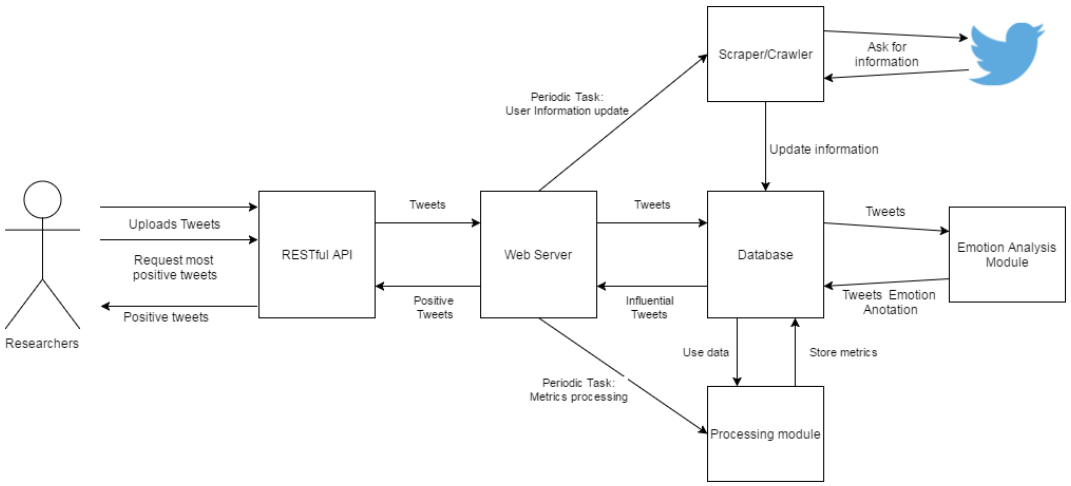

Figure 4.2: UC2: Emotion Analysis

4.2.3 UC3: Targeting

Enterprise SA. wants to launch a new product in a determined sector, and they want to have the most influential users of Twitter in this sector to do advertising of the product. They decide to use SCANER for this. They load a set of tweets from a hashtag search of the sector. SCANER will extract the information about the users and tweets related to this search. Once the data is analysed, Enterprise SA. will have a ranking of the more influential users in the sector, so the can contact them for their marketing campaign.

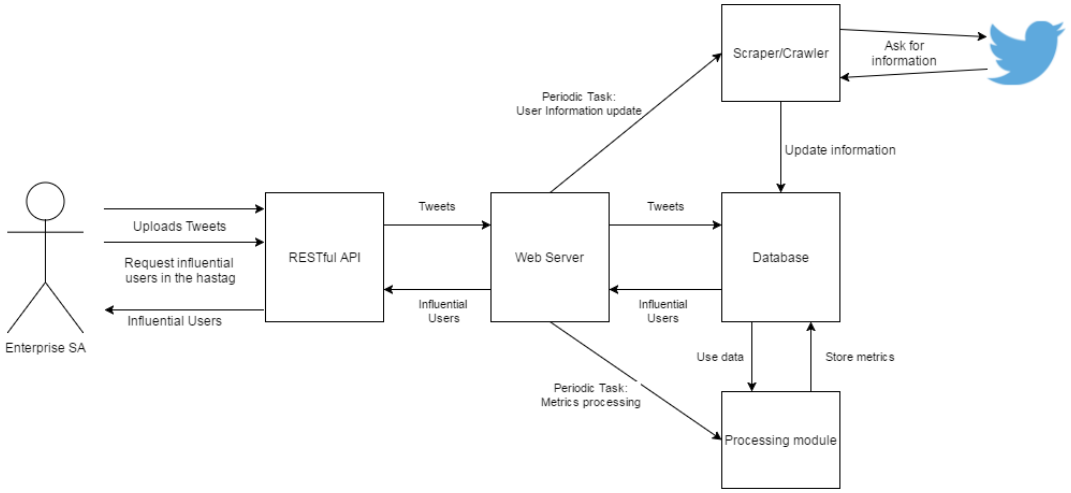

Figure 4.3: UC3: Targeting

4.2.4 UC4: Task Monitoring

Enterprise SA. wants to know the status of a targeting process that has being initiated through a request. They will connect to the server and use the task status REST call with their task id. The server will return the status of the task, and if it is finished, the results.

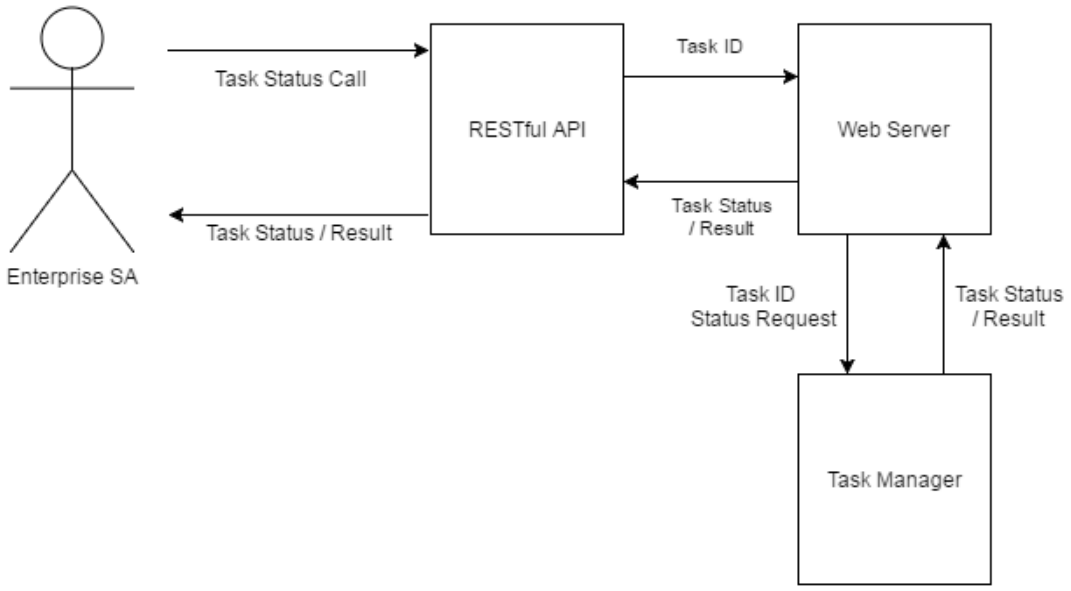

Figure 4.4: UC4: Task Monitoring

4.2.5 UC5: Spread of Content

Online News is an online newspaper that uses Twitter to share their most import news. They want to analyse the spread of their content and they uses SCANER for it. They will load a set of tweets and retweets of their tweets into the platform. SCANER will extract the information about the users and tweets related, and will create the network with all the relationships. Online News now can study which users are sharing most of their content, which content is more popular, and the real spread of the news.

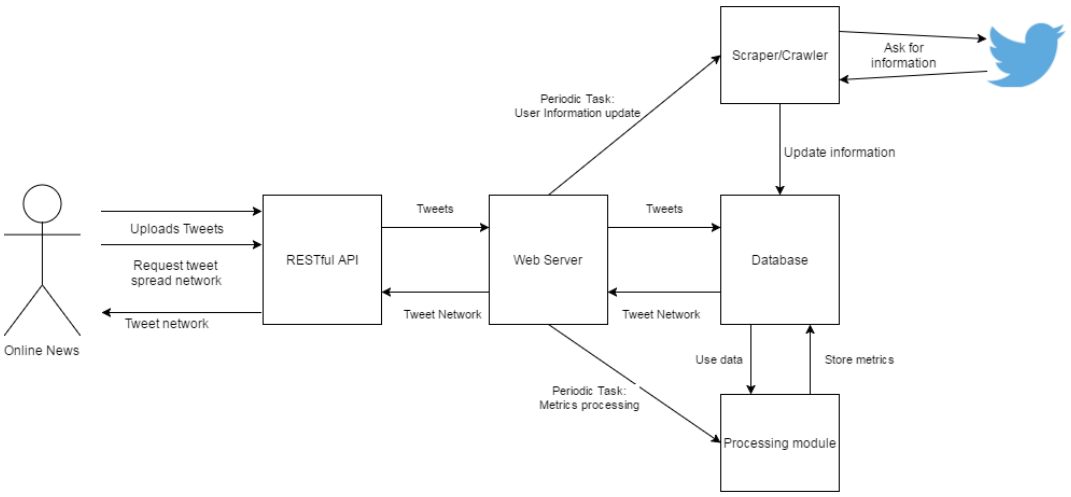

Figure 4.5: UC5: Spread of Content

4.3 Database requirements

As stated in Chapter 2, data from social media has a particular set of properties that make us require an storage system with specific requirements. These requirements are:

- Primary:
  - Advanced relationships capabilities: As said in chapter **??**, the data that will be used in the system is social media data. This data focuses on the relationships between the users and the content. With data with these characteristics, we require an infrastructure capable of manage these relationships and work with them in an efficient way.
  - Query speed: Due to the large amounts of data we are going to work with, we need a data infrastructure capable of delivering fast queries and data access.

  - Traverse speed: Due to the importance of the relationships in our data, it is crucial to be able to traverse through this relationships efficiently. Instead of query each record separately, we need to be able to query related records using their relationships, making it faster to access related data. It will be one of the main operations that will be performed with the data, so we need high speed.

  - Unstructured data optimization: In social media data, a large part of the data is unstructured data. The term unstructured data (or unstructured information) refers to information that either does not have a pre-defined data model or is not organized in a pre-defined manner. Unstructured information is typically text-heavy, but may contain data such as dates, numbers, and facts as well. Usually, the content generated by the users matches this description, because it can vary widely from user to user, or type of content to type of content. Due to this, we need and efficient and reliable way to store unstructured data in our system.

  - Open Source: Open Source software has a series of advantages such as costs (generally free), continually evolving, you are not tied to a particular vendor's system and you can adapt and modify the software for our own requirements.

- Secondary:

  - Python API package: As we have chosen Python as the programming language for the project, it is important to have a python library that allows to create an easy way to communicate with the database directly from the application, code in a reliable and efficient way.

  - Distributed: In case that the volume of data grows too large, or our computation resources for a single machine are limited, having a distributed database infrastructure could improve our performance and allow to use multiple less powerful computers to run the queries and operations.

4.4 Summary of requirements

After analysing the previous use cases, some clear requirements seem to stand out. Many of the user cases require similar features or certain characteristics in the architecture, pointing out the necessities or requirement of the system. In this section we will present those requirements, separated in two groups: functional and non-functional requirements.

4.4.1 Functional requirements

- **FR1:** The system must be able to link, relate, extract and process information from Twitter. We can observe this requirement in user cases UC1, UC2, UC3 y UC5.
- **FR2:** The system must update information in order to analyse spread and evolution. We can observe this te in user cases UC1, UC2 y UC5.
- **FR3:** The system must differentiate the data and classify it into different topics for different analysis. We can observe this requirement in user cases UC1, UC2, UC3 y UC5.
- **FR4:** The system must update information in order to analyse spread and evolution. We can observe this requirement in user cases UC1, UC2 y UC5.
- **FR5:** The system must be able to integrate other services such as the Emotion Analysis Module in order to extend its functionality. We can observe this requirement in user cases UC1, UC2 y UC3.
- **FR6:** The system must be able to process the data offline, if it was already stored in the database.
- **FR7:** The system must offer a mechanism to retrieve the processed data if this information can not be delivered in real time. We can observe this requirement in user cases UC4.
- **FR8:** The system must have a mechanism to load data available for final users. We can observe this requirement in user cases UC1, UC2, UC3 y UC5.

4.4.2 Non-functional requirements

- **NFR1:** The system mechanism to load the user data into the system must be user-friendly. We can observe this requirement in user cases UC1, UC2, UC3 y UC5.
- **NFR2:** The system must be able to process huge amounts of data, even when it is received simultaneously. We can observe this requirement in user cases UC1, UC2, UC3, UC4 y UC5.
- **NFR3:** The system must offer a modular architecture. We can appreciate this requirement in all of the user cases.

- **NFR4:** The system must comply with Twitter API limitations and policy. This requirement is not directly defined, but is present in all use cases.

CHAPTER 5

Enabling Technologies

This chapter introduces which technologies have made this project possible. First of all the programming language, explained in section 5.1. Second of all, the technology that allows to virtualize the system and deploy it with a stable configuration and fast will be presented in section 5.2. Third, the storage support for the system, where all the social media data, will be presented in section 5.3. Finally, the technologies that have been used to develop the web interfaces that enable the RESTful API and the interaction with the users.

5.1 Python

In this project we are going to use Python as the programming language for our software. In general, Python is an efficient language that has three main characteristics that improves its efficiency: dynamically typed, concise and compact [20]. It has an easy syntax, high readability, it is object oriented and extensible. All of these features aid in prototyping, shorten the development cycle and result in a cleaner, smarter and more effective code. It simplifies and accelerate the development process, also making it very accessible [21]. It is worth mentioning the large amount of libraries available for Python, which reduces the coding work.

However, it has some disadvantages, such as concurrency. Concurrency and parallelism, although completely possible in Python, are not designed-in for elegant use, as with JavaScript and Go. Other disadvantage is the speed, because Python is executed by an interpreter instead of compilation, which causes it to be slower than if it was compiled and then executed. However, for most applications, it is by far fast enough.

5.2 Docker

For the deployment of this project, we are going to use Docker. Docker is a platform that allows to package an application with all of its dependencies into a standardized unit for software development [22]. Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

Containers have similar resource isolation and allocation benefits as virtual machines but a different architectural approach allows them to be much more portable and efficient. These containers include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in userspace on the host operating system. They're also not tied to any specific infrastructure. The main advantages of this platform are the ability to accelerate developer onboarding, the easy sharing and distribution and the capacity to eliminate environment inconsistencies.

5.3 Database

For this project, we need a storage infrastructure that is able to store social media data in an efficient way. As said above, social media data is structured in a particular way, where the relationships are more important than the content itself. We are going to take a look on the different options available in the market and what requirements they fulfill.

5.3.1 Relational Databases

Relational databases are one of the fundamental building blocks of modern database architecture and are designed for fast storage and retrieval of large quantities of data [23].

In this type of databases, the data is stored in relational model, with rows and columns. Rows contain all of the information about one specific entry, and columns are all the separate data points. Each record conforms to fixed schema, meaning the columns must be decided and locked before data entry and each row must contain data for each column. This can be amended, but it involves altering the whole database and going offline. In these databases, the scaling is vertical. In essence, more data means a bigger and more powerful server. It is also possible to scale a relational database across multiple servers, but this is a difficult and time-consuming process. The vast majority of relational databases are ACID compliant. Relational databases mainly use SQL as the language for querying and maintaining the database.

A hefty part of designing a relational database is dividing the data elements into related tables. Once you're ready to start working with the data, you rely on relationships between the tables to pull the data together in meaningful ways [24]. In relational databases, relationships link two or more tables. There are three different types of association between tables:

- One-to-one: Both tables can have only one record on either side of the relationship. Each primary key value relates to only one (or no) record in the related table. Most one-to-one relationships are forced by business rules and don't flow naturally from the data. In the absence of such a rule, it is usually possible to combine both tables into one table without breaking any normalization rules.
- One-to-many: The primary key table contains only one record that relates to none, one, or many records in the related table.
- Many-to-many: Each record in both tables can relate to any number of records (or no

records) in the other table. Many-to-many relationships require a third table, known as an associate or linking table, because relational systems can't directly accommodate the relationship.

The database system relies on matching values found in both tables to form relationships. When a match is found, the system pulls the data from both tables to create a virtual record. Most of the time, the resulting record is dynamic, which means any change made to the virtual record will usually work its way back to the underlying table. A diagram of the structure of a relational database is shown in Figure 5.1.

The most popular relational databases in the market are Oracle Database, MySQL and Microsoft SQL Server [25].

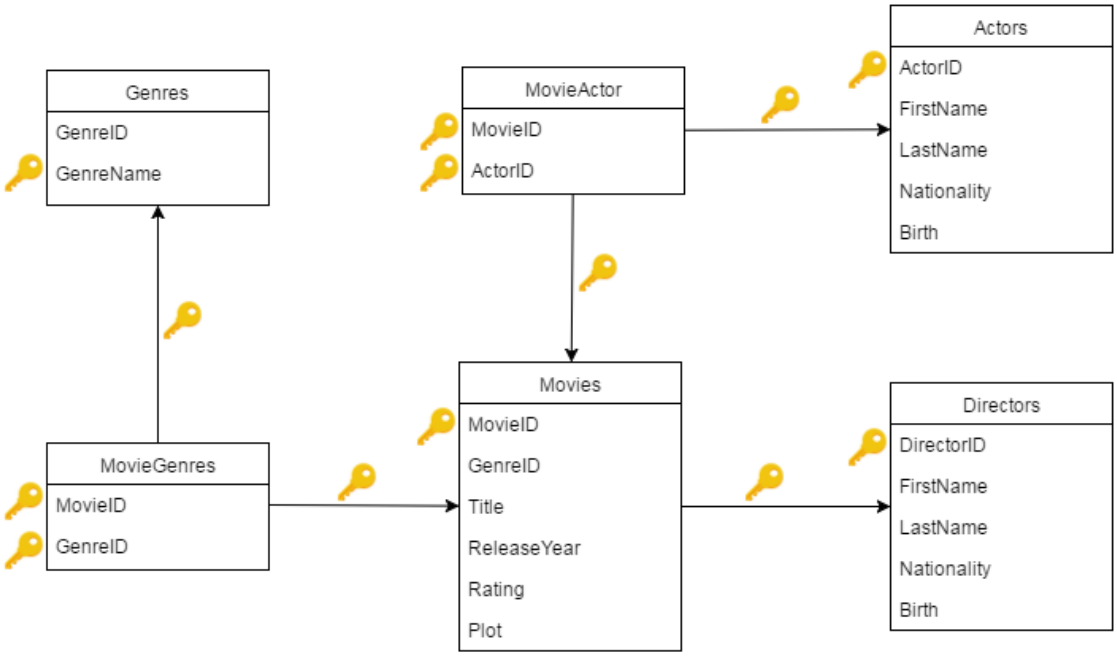

Figure 5.1: SQL Relationship Model

5.3.2 Document-oriented databases

Document-oriented databases can be considered to be next step to the key-value stores because they store more complex data than the key-value stores [26]. The central concept of a document-oriented database is the notion of a document. While each document-oriented database implementation differs on the details of this definition, in general, they all assume documents encapsulate and encode data (or information) in some standard formats or encodings. Documents are records that describe the data in the document, as well as the

actual data; and allow values to be nested documents or lists as well as scalar values, and the attribute names are dynamically defined for each document at runtime.

In this type of database, the relationships between entries can be created in two ways [27]:

- References Relationship: This type of relationship stores the data by including links or references, from one document to another. Applications can solve these references to access the related data in the structure of the document itself.
- Embedded Documents: This type of relationship stores in a single document structure, where the embedded documents are disposed in a field or an array. These denormalized data models allow data manipulation in a single database transaction.

This presents a set of advantages, such as an improvement in performance due to the fact that the different information of a record is stored contiguously in memory and that speeds up the access. The other main advantage of this structure is their ability to store unstructured data in a simple and efficient way. The most popular document-oriented database in the market is MongoDB [28].

5.3.3 Graph-oriented databases

Graph-oriented databases are a type of database in which the data structures for the schema and instances are modeled as a directed, possibly labeled, graph, or generalizations of the graph data structure, where data manipulation is expressed by graph-oriented operations and type constructors, and appropriate integrity constraints can be defined over the graph structure [29]. Graph-oriented databases are used in areas where information about data interconnectivity or topology is more important, or as important, as the data itself. In these applications, the data and relations among the data, are usually at the same level. Introducing graphs as a modeling tool has several advantages for this type of data:

- It allows for a more natural modeling of data. Graph structures are visible to the user and they allow a natural way of handling applications data, for example, hypertext or geographic data. Graphs have the advantage of being able to keep all the information about an entity in a single node and showing related information by arcs connected to it. A user can define some part of the database explicitly as a graph structure, allowing encapsulation and context definition.

- Queries can refer directly to this graph structure. Associated with graphs are specific graph operations in the query language algebra, such as finding shortest paths, determining certain subgraphs, and so forth. Explicit graphs and graph operations allow users to express a query at a high level of abstraction.

- For implementation, graph databases may provide special graph storage structures, and efficient graph algorithms for realizing specific operations.

Graphs databases contains connected entities (the nodes) which can hold any number of attributes (key-value-pairs). Nodes can be tagged with labels representing their different roles in the domain. In addition to contextualizing node and relationship properties, labels may also serve to attach metadata, index or constraint information, to certain nodes[30].

Figure 5.2: Graph-oriented Database Relationship Model

In this type of database, relationships provide directed, named semantically relevant connections between two node-entities. A relationship always has a direction, a type, a start node, and an end node. Like nodes, relationships can have any properties. In most cases, relationships have quantitative properties, such as weights, costs, distances, ratings, time intervals, or strengths. As relationships are stored efficiently, two nodes can share any number or type of relationships without sacrificing performance. Note that although they are directed, relationships can always be navigated regardless of direction. A diagram with the structure of a graph database is shown in Figure 5.2. There is one core consistent rule in a graph database: “No broken links”. Since a relationship always has a start and end

node, you can't delete a node without also deleting its associated relationships. You can also always assume that an existing relationship will never point to a non-existing endpoint.

The most popular graph-oriented database in the market is Neo4j [31].

5.3.4 Hybrid databases

Hybrid databases are NoSQL databases which main characteristic is that they combine two or more database schemas in one database system. The properties of these databases depends on the combination of schemas they use. One of the most used hybrid databases is OrientDB [32]. It is a combination of document-oriented and graph-oriented databases, combining the advantages of both schemas.

5.3.5 Comparison

Previous sections have introduced the types of databases. For this project, we have compared the most popular databases of each type in the context of social network information.

Relational Databases have the advantage of the query speed, but lack the relationship structure that we need to use in order to get efficient operations and they do not support unstructured data.

Document-oriented Databases have the advantages of accepting unstructured data and fast query speed, specially for adjacent data. The problem with this type of databases is that the lack the relationship structure aforementioned.

The results of this comparison are shown in Table 5.1.

After analysing the different options available for our database choice, we have decided to choose OrientDB. OrientDB is a hybrid graph-document oriented database. It combines the speed and flexibility of document-oriented databases with the advanced relationship capabilities of graph-oriented databases. Its main features are [33]:

- Apache 2.0 license
- ACID transactions
- Free of cost
- Gremlin Language for graph computing
- SQL language Syntax for graph computing

Table 5.1: DB comparative

DB	Advanced relationships	Query Speed	Traverse Speed	Unstructured data optimization	Open Source	Python API package	Distributed	License
MySQL	No	High	Low	No	Yes	Yes	Yes	GPLv2
Oracle	No	High	Low	No	No	Yes	No	Product License
MongoDB	No	High	Low	Yes	Yes	Yes	Yes	GNU AGPL v3.0
Neo4J	Yes	High	High	Partial	Yes	Yes	Yes	GPLv3
OrientDB	Yes	High	High	Yes	Yes	Yes	Yes	Apache2.0

- RESTful API
- Fast performance
- Multi Master Replication
- Sharding
- Official release APIs for JAva, .Net, PHP, Python and many others
- Developed in JAVA hence can be run in any Operative System

OrientDB focuses on performance and it has been built to extremely optimize data retrieval operations and traverse operation. All of this, combined with the features mentioned above, make OrientDB a good choice for university development.

5.4 Web framework: Flask

Flask is a micro web framework written in Python and based Werkzeug toolkit and Jinja2 template engine [34]. Flask is called a micro framework because it does not presume or force a developer to use a particular tool or library. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions [35]. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Examples of applications that make use of the Flask framework are Pinterest [36], LinkedIn, as well as the community web page for Flask itself.

5.5 Swagger

Swagger is a RESTful API framework, available in multiple programming languages and completely open source. Using Flask as base, it allows to easily create REST APIs with quick and flexible customization. It offers interactive documentation, client SDK generation and discoverability [37]. It is now part of the OpenAPI specification, which is a specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services [38].

The frameworks allows to create an RESTful API from a structure defined in a documentation file. It implements the different API calls, with field validation in the call and in the response, It also offers an interactive interface to try the API and create custom calls. This allows for easy and quick early deploy and development.

5.6 Celery

Celery is an asynchronous task queue/job queue based on distributed message passing [39]. It is focused on real-time operation, but supports scheduling as well.

The execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing, Eventlet, or gevent. Tasks can execute asynchronously (in the background) or synchronously (wait until ready).

Celery is easy to integrate with web frameworks, some of which even have integration

packages. It is written in Python, but the protocol can be implemented in any language. It can also operate with other languages using webhooks. It also has multi broker support. It supports RabbitMQ, Redis, MongoDB, etc.

5.7 Redis

Redis is an open source (BSD licensed)[40], in-memory data structure store, used as database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

In order to achieve its performance, Redis works with an in-memory dataset. Depending on your use case, you can persist it either by dumping the dataset to disk every once in a while, or by appending each command to a log. Persistence can be optionally disabled, if you just need a feature-rich, networked, in-memory cache.

Redis also supports trivial-to-setup master-slave asynchronous replication, with very fast non-blocking first synchronization, auto-reconnection with partial resynchronization on net split.

CHAPTER 6

Architecture

This chapter describes the architecture of the system. We will explain how the different modules are structured, their roles and how they communicate between them. Finally, we will explain how the user can access the platform to load data or retrieve information.

6.1 Overview

Based on the requirements identified in the previous section, a preliminary set of design criteria can be set for our influence calculating system:

1. Periodical Analysis: This criteria is set due to the requirement FR4. Social Network are very active sites. The content is ephemeral, and relevance change in time. Also, one of the features of our system is the temporal evolution analysis of the content or a user. This means that our system must be constantly updating the data and acquiring more information, in order to do constant analysis periodically.

2. Modular: The system should be modular in order to provide flexibility and further improvements or extended functionalities. It also helps to customize the system to the user needs.This answer the requirement FR5.

3. Topic centric: This criteria is set to fulfil requirement FR3. As said before, context is the key to understand the influence of a user or his content. Relevance is dependent of the context factors, such as the subject, time, scope, etc. This means that our system must make independent analysis to the same data, one for each topic this data is included in. Different topic analysis will offer different results for the same content.

4. Periodical Data Updates: Our system needs to be continuously connecting to Twitter in order to extract new data or update the existing one. This will allow to answer to the requirements FR1, FR2 and FR4.

5. Accessible by Web: Being a system based on Social Network Data Analysis, our users will demand access by web, so a web interface and API must be provided. This allows to fulfil requirements FR8 and NFR3.

6. Offline processing: The system should be able to process the data and deliver the results on demand when the calculation process is time consuming or can not be done in real time. This criteria answers to the requirement FR6, NFR2 and NFR3.

Taking into account these criteria, we have designed a modular architecture. The system is composed of different modules or sections with differentiated functions: REST API, Web Server, Processing Module, Scraper/Crawler, Emotion Annotation Module and the Graph Database Each one of these modules will be detailed in the following sections. The whole system will be connected permanently to the Internet in order to be able to extract

information from Twitter when needed and to be accessible by the users. This access will be done through the API of the system.

A diagram of the architecture is shown in Figure 6.1.

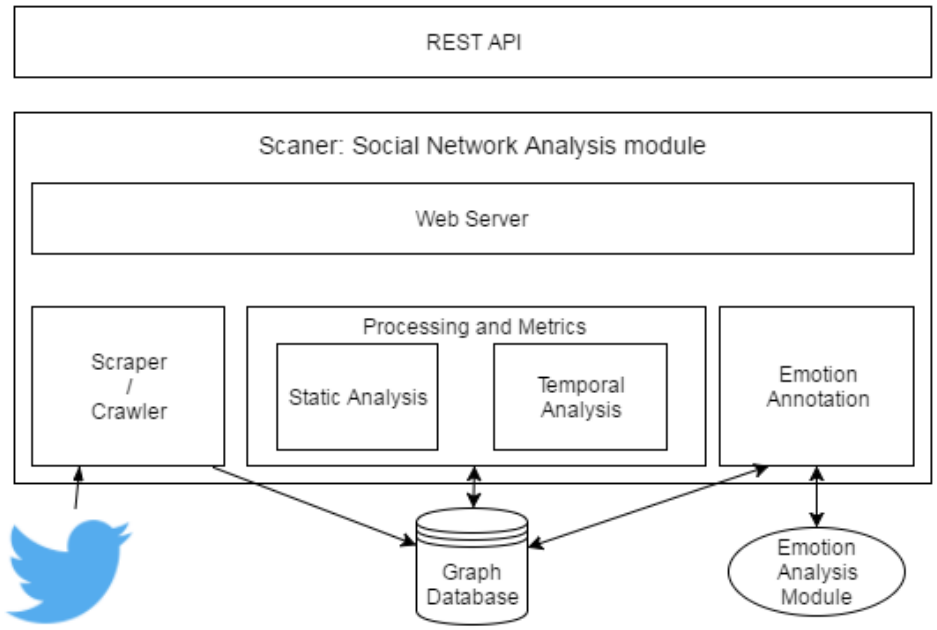

Figure 6.1: General Architecture

In the following sections, the different components of the system will be described.

6.2 Graph Database

All the data extracted from Twitter and the metrics calculated for the content and users of the extractions must be stored somewhere. In our case, we will use a graph database. The objective of this database is to provide an integrated centralized storage for the data that allows to connect the different information with powerful relationships and traverse capabilities.

The database integrates information relative to all the modules and different topics, allowing us to work with all of them and our relationships at the same time, and also offers a single access point to all the data for the API.

The structure of the data in the database is explained in section 7.3.

6.3 Processing module

The processing module is the software module that implements the influence algorithms and the metrics calculation. It is divided in two subsections: The Static Analysis and the Temporal Analysis.

6.3.1 Metrics and Analysis

Internally, metrics are classified in two different types: direct and indirect metrics. Direct metrics are directly obtainable from the extracted data, such as the number of followers a user has. The Social Context Analysis module obtains direct metrics as soon as new social media content is stored in the database, and these metrics are updated when new information arrive. For instance, the Social Context module is configured to refetch general information about users periodically, so these metrics would be updated as well.

Indirect metrics are obtained through data processing, for example User Influence. These metrics are calculated periodically, as they have a high processing cost and require accessing all the information in the database.

A flux diagram of the Processing module working is shown in figure 6.2.

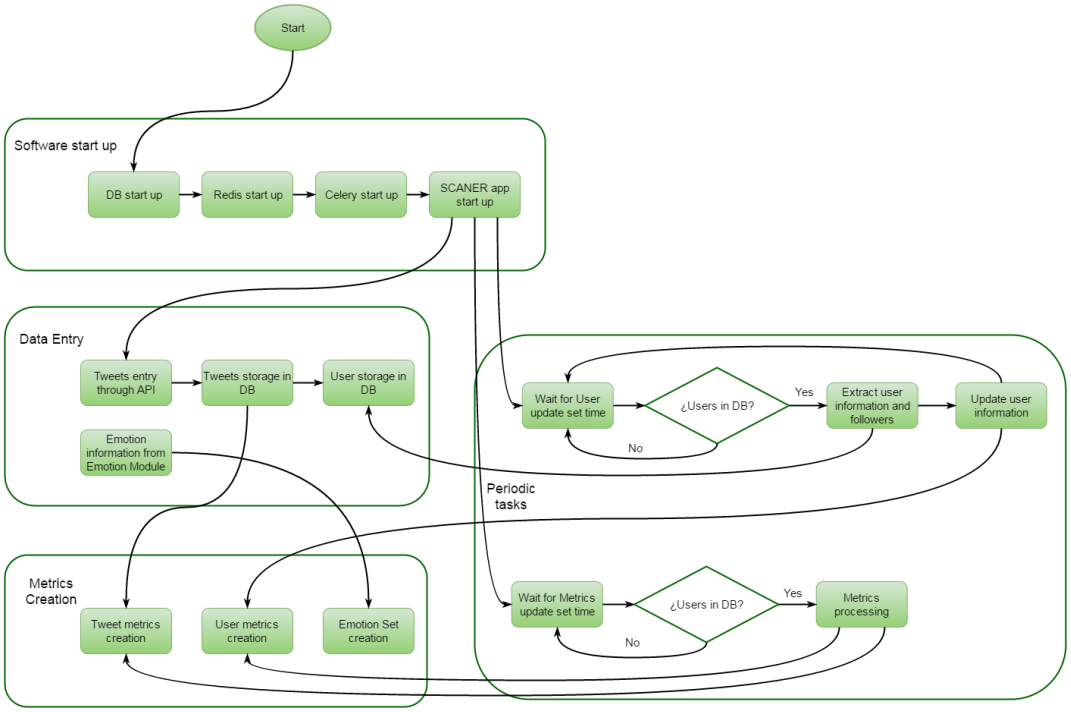

Figure 6.2: Flux Diagram

6.3.2 Static Analysis

This section is responsible for calculating the different indirect metrics of tweets and users periodically and to update and store their metrics each time it receives updated data. When the system receive new tweets, they are stored in the database. The Static Analysis section generate a new metrics object for the new tweet with its corresponding timestamp. This will allow the Temporal Analysis to keep track of the evolution in time of a tweet.

The Static Analysis software include the implementation of the algorithm explained in section 3.4. The program runs through the entire database, calculating the metrics for relevant tweets and users. After the calculation, it stores the new values in the database and updates the old values.

6.3.3 Temporal Analysis

This section is responsible for calculating and deliver metrics that depend on time. It can process the social media data stored in the database to evaluate the evolution in time of some parameters of the users or the tweets, such as number of followers, number of tweet published, number of retweets, influence, relevance, etc. This allow to get a record for the different users and tweets and analyse their growing or their spread.

This is an important feature, because it can be useful to analyse the evolution in time of some tweet to get information about lifespan of content in Twitter and the duration of the interest of the users for the content. This is specially interesting for marketing studies or special events tracking.

6.4 Web Server

The web server component is the software responsible for the implementation RESTful API and the interface with the user. The server also offers a graphical interface to access the RESTful API and make custom calls. This module will be responsible for making request to the database and retrieve the answers.

Through the API, the user can introduce new social media data in the system, look for different topics, users or tweets in the database, check their metrics and check the status of tasks queued in the server.

6.5 API

An application programming interface (API) is a set of routine definitions, protocols, and tools for building software and applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types, defining functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. A REST API is an API that is simply a specification of remote calls exposed to the API consumers.

RESTful APIs are a subset of REST interfaces that also satisfy 6 constraints:

- Client–server. By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.
- Stateless. Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.
- Cacheable. Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.
- Uniform interface. By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behaviour of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.
- Layered system. The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.
- Code on demand (optional). REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

Our RESTful API grants access to the contents stored in the database: tweets, users and metrics. It allows to load data in the database for different topics, and also, to retrieve the information of the different analysis performed by the system. A full description of the API methods is presented in the table 6.1.

Table 6.1: API Calls

Description	API call
Topics	
Obtain information of a particular topic	GET /topics/{topicID}
Obtain list of available topics	GET /topics
Obtain social network of a topic	GET /topics/{topicID}/network
Users	
Obtain list of available users	GET /users
Obtain information of a particular user	GET /users/{userId}
Obtain social network of a user	GET /users/{userId}/network
Obtain the emotion of a user	GET /users/{userId}/emotion
Obtain the sentiment of a user	GET /user/{userId}/sentiment
Obtain the metrics of a user	GET /user/{userId}/metrics
Tweets	
Obtain list of available tweets	GET /tweets
Obtain information of a particular tweet	GET /tweet/{tweetId}
Obtain the history of a particular tweet	GET /tweets/{tweetId}/history
Add a tweet to the database	POST /tweets
Delete a tweet from the database	DELETE /tweets/{tweetId}
Obtain the emotion of a tweet	GET /tweets/{tweetId}/emotion
Obtain the sentiment of a tweet	GET /tweets/{tweetId}/sentiment
Obtain the metrics of a tweet	GET /tweets/{tweetId}/metrics
Tasks	
Obtain the list of tasks	GET /tasks
Obtain the status of a particular task	GET /task/{taskId}

6.6 Task manager

Each one of the operations in the system will not be processed by a single process. With heavy processing task such as the indirect metrics calculation, the system could be blocked for several hours, making it unaccessible and not responsive. To fix this problem, our system uses a task manager system underneath.

This task manager will have a queue with all the processing tasks and will execute them in FIFO order. This will allow to run time consuming tasks without blocking all the system and retrieve the results later. It also separates the different processing modules and creates a single processing queue independently of the number of modules. This allow to add or remove modules without affecting the rest.

CHAPTER 7

Implementation

This chapter describes how the different modules of the system have been implemented and which technologies have been used.

7.1 Overview

In this section, we will describe the implementation of the most complex parts and the technologies used. The whole system is implemented in Python using Model-View-Controller pattern with different modules covering the different parts in the pattern. We will present the implementation description of the different modules explained in chapter 6.

7.2 Graph Database

To implement the database system, we have used OrientDB. In order to create the database model, we use a script that runs in the OrientDB console. This script creates the different classes that our database will contain, with all their attributes and types of relationships. It also creates the index that will speed up the queries.

To allow our server to communicate with the database, we use the python package pyorient, which implements an interface with OrientDB. Through this interface we can run queries, commands and extract the information we need from the database. We will also use the OrientDB web interface to check the data of our database, change settings and visualize the structure and network of the data in the integrated visualizer.

7.3 Database modeling

The data model we are going to use in this project will divide the data in six different classes:

- User: This class represents users stored in the database. Users will be the authors of the content of the social media data, and will be related with other users.
- Tweet: This class represents the tweets stored in the database. Tweets will be de social media data created by users, and they will be related between them.
- Topic: This class represents the different topics that users and tweets belong to. Topics group users and tweets by their theme or origin, and allows to separate the metrics and content from different contexts.
- User metrics: This class represents the metrics of a user stored in the database.
- Tweet metrics: This class represents the metrics of a tweet stored in the database.

- EmotionSet: This class represents the structure of the emotion metrics of a tweet stored in the database. It contains the following attributes.

This classes will be interconnected by the next set of relationships and a diagram is shown in Figure 7.1.

- Follows: This relationship links a user (follower) with the user he follows.
- Retweet: This relationship links a tweet with a retweet.
- Replied: This relationship links a tweet with a replied tweet.
- Created by: This relationship links a tweet with the user who posted it.
- Retweeted by: This relationship links a tweet with the user who retweeted it.
- Replied by: This relationship links a tweet with the user who replied to it.
- Last metrics: This relationship links a user or a tweet with its most recent set of metrics.
- has Emotion Set: This relationship links a tweet with its most recent set of emotion metrics.
- Belongs to topic: This relationship links a user or a tweet with the topic it belongs to.

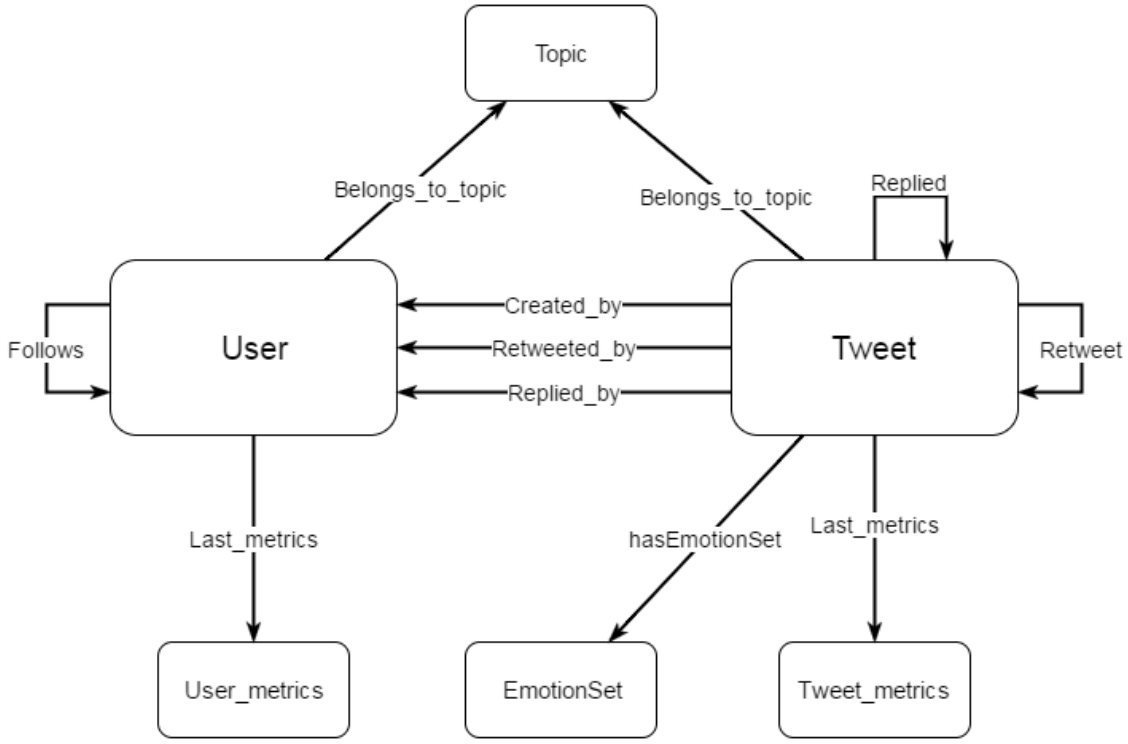

Figure 7.1: DB model

7.4 Web Server and API

This section describes the implementation of the server and the API described in sections 6.4 and 6.5 and will explain the technologies used and the different advantages that brings us.

Our Web Server and the API has been implemented using Swagger, presented in section 5.5. Swagger allows to create the API through a descriptive document in YAML format. YAML is a human-readable data serialization language that takes concepts from programming languages such as C, Perl, and Python, and ideas from XML and the data format of electronic mail. This document allows to define the API calls, the response and call formats and offers a really useful validation system for calls and responses. It is possible to define the structure of the data that will be sent in API calls to validate it before it is processed by the server. It also allows to validate the data of the response from the server, to determine if the call was successful or not. The web form has different fields labelled according to the call definition in the YAML file. When making the call, it will show the result in case the call was successfully answered, or the error code in case the data could not be validated or the call was not successfully answered.

Other advantage that Swagger offers is the creation of a visual interface for the API. This interface allows to view the API structure defined in the YAML file for the calls, visualize examples of the different calls and responses for each method (that can be defined in the YAML file aswell) and to make custom calls to the API with a web form.

These advantages are really helpful, specially during the development phase. It allows the developer to try different API calls really easy and fast and offers access to the error messages instantaneously, making debugging much easier.

7.5 Docker

Docker is probably one of the easiest environments to create a virtualized instance based on a number of flavours of operating systems. Rather that having to install an operating system yourself, you can download one of the many guests templates or "images" available directly from the Docker community [41].

Docker has a scripting language which can be used to create a new instance with a predefined list of commands and properties which will be used to create your new Docker instance. You could, for example, use a docker file to install Apache, configure the firewall and any further configurations we may need to make. The benefits to using a Dockerfile, rather than making all the changes directly and saving the image are that the underlying OS and the additions that you wish to make are completely independent [42]. You can, for example, run a Dockerfile on any OS image.

The Scaner platform consists of several interconnected services (e.g. redis, celery, etc). Each of these services will be deployed as a container, and connected to the rest. Each container has its own configuration file with the script that allows to run the server in exactly the same configuration each time. This allows the system to be easily and quickly deployed, simplifying the development and testing.

Docker Compose simplifies the deployment of multiple interconnected containers. Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a compose file to configure your application's services [43]. Using Compose is basically a three-step process.

1. Define your app's environment with a Dockerfile so it can be reproduced anywhere.

2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.

3. Lastly, run docker-compose up and Compose will start and run your entire app.

Compose also allows to stablish the initialization order of the services and the connections between them. With this Docker Compose file, using a single command, we can create and start all the services from our configuration.

7.6 Task Manager

The task manager is implemented using Celery, introduced in section 5.6. Celery allows to have a several task queues, with multiple workers to allow multitasking and process queuing. In case that a heavy process is blocking the access to the database, the web server could answer the users communicating the state of the server.

Celery also offers a really useful feature: task status consulting and result retrieval. This feature allows to ask celery the status of a task using the task id. This is useful when a user of the system has asked for a time consuming task that can be dispatched immediately. This user could see the status of the task and retrieve the result once the task has finished processing.

This task manager and the modules of the system work independently, so it is necessary a communication system between them. In our system, the modules and the task manager are communicated through a messaging system using a message broker. The messaging system is the mechanism that allows the different modules to exchange messages and initiate and queue tasks.

It is implemented using Redis as a message broker, explained in section 5.7. Redis offers a message queue system, with reliable communications and fast performance. The advantages of Redis are:

- High speed due to in memory datastore.
- Can double up as both key-value datastore and job queue.

It also allows to store the messages in memory, in case they are not delivered. This messages can be sent later until they are received, so no messages are lost in case that one of the components is down. This means that the task already queued will not be lost even if the task manager is down, and it will be run once the module is up and running again.

On the other side, the in memory datastore implies some drawbacks. One of them is that, in case that the whole server is shut down, the queue is lost, due to be stored in

memory and not in the hard drive. The other problem is that, if our hardware has limited memory storage, it may not have enough space to work properly with a high amount of task.

In our case, the pros from Redis outweigh the cons. Worst case scenario, losing our task queue in case of a total shut down is not specially problematic because most of our task are periodical, and will be executed again when the system recovers.

One of the alternative to Redis is RabbitMQ. RabbitMQ uses hard drive storage, which makes it more slow but more reliable. It also has a more complex precess of integration. Because of this, we decided to use Redis.

CHAPTER 8

Conclusions and future work

This chapter will present the conclusions obtained from this project and the lines of future work and improvement.

8.1 Conclusions

In this section we will present the conclusions extracted from this project. We are going to analyse if the project goals have been achieved and we will also present the different challenges we have faced during this project and how we have overcome them.

8.1.1 Goals

Now we will analyse if the original goals of the project has been successfully achieved.

Defining and characterizing the concept of social context

The goal of defining and characterizing the concept of social context has been achieved. We have been able to offer a definition, and we have analysed the most important components and metrics. We have studied the effect and importance of the different metrics and we have studied the different approaches to obtain the most complex ones.

Creation of the platform for social context analysis

Our platform has successfully been able to extract and process social media information data from Twitter, allowing us to analyse and process different metrics from tweets and users.

The platform includes the model of users and tweets presented in this work. It creates the appropriate relationships between users and their related tweets, it calculates their influences and makes this information available through a REST API.

A modular architecture enables the addition of new modules that extract more information or use the same data for different purposes. The independent task system and API further contribute to decouple the modules in the platform.

Thanks to the technologies that we have used for the project, we have been able to of retrieving, storing and processing large amounts of information. OrientDB has allowed us to store all the amount of data that we needed with no problem, the crawler has been able to extract the necessary information from Twitter, being only limited by the Twitter API rate limit. And finally, we have been able to process all of this information thanks to the task manager. The task manager allows us to task time consuming processes and retrieve the results later, so having a lot of data simply made us wait more for the tasks but the system worked in the meantime and could accept more task.

We have been able to create an interface to interact with the users. Using Swagger has made the interface development really easy, as explained above. Using it, we have created a REST API that allow user to access the information stored in the database, the running tasks statuses and the results of finished tasks.

8.1.2 Challenges

In this section we will describe some of the challenges that we have encountered in the development process of this project.

One of the main challenges we have faced was to determine which metrics from users and tweets were relevant, how to obtain them and what we could be their utility. We had to research how different companies and researchers were studying Twitter and what metrics they were using. After that, we needed to synthesize all this information in one list. This led to the problem of the "influence" metrics. There is not a standard method to calculate it, so we had to compare several algorithms until we found the one that we liked most.

Another challenge we had to face was the deployment of the system. To solve the problem we decided to use Docker. Docker has proven to be a really useful tool. Using Docker has simplified the start up of the system during the development process and has ensured that the software ran with the same configuration each time. Docker compose has been specially useful, because it allowed to deploy all the services that compose SCANER with a single command and the same configuration everytime and following a specific initialization order.

During the development of the interface, we identified the need for an API definition that could be used both in the server side (platform) and client side. In other words, we needed a language agnostic interface that would allow us to use the API from different modules and software both in server and client side. We decided to define the API using Swagger. Swagger has greatly simplified the development process. Defining the API for the platform was really simple once we have understood the basics of Swagger. The interface provided makes the process of creating calls really easy and shows the errors in case that a problem occur. The field validation system helped to ensure that our system always received correct data and detected errors in the inputs. Besides that, the online API editor that Swagger offers aids to develop and correct your configuration files. It also has a preview mode that allows you to see the changes each time you process the file without deploying your system. All of this combined made the development process much faster and agile.

The migration of the files that contained users and tweets to the OrientDB database

was also a challenge. We tried to use the ETL (Extract, transform and Load) system that OrientDB provides, but it gave us a lot of errors due to the unstructured nature of the data. This ETL system is better suited to migrate databases from relational databases to OrientDB. At the end, we solved using scripts to load the data until the actual platform mechanisms for loading data were developed.

Another challenge in the coding process was to implement the aforementioned influence calculation algorithm. It required to process all the information related to a topic at the same time. This led us to memory shortage and we have to optimize and change the code in order to be able to work with reduced memory without loosing too much performance.

The last challenge we faced was to process large amounts of data. Due to limited hardware, we decided to process it offline using task management and queuing. We use Celery to achieve this. Celery has allowed SCANER to have a task manager integrated with the processing code. It has an easy code decoration system that allowed to create and queue tasks in a simple and fast way. One of the most interesting features has been the ability to queue a task that will take a long time and be able to retrieve the result later, when the task has already finished. This allowed us to work with large amounts of data without powerful hardware.

8.2 Future Work

In this section, we will present some of the future work lines and some ideas to continue expanding the project. The most important of them are the following:

Develop more modules

The basic idea of the architecture is the modularity of the system. With that in mind, we will continue developing processing modules to enhance the processing capabilities of SCANER and add more features.

Develop authentication system

Currently, our system does not support an authentication system that allows to register the activities of the users and limits the access to the user to certain data. With this system our objective is to be able to separate data from different users and isolate it from the information of the rest of the users.

Optimization

SCANER needs to process a large amount of data each time the influence metrics are

computed. This makes optimization vital, in order to reduce the load on the system and reduce the waiting times. Optimize the current software will allow us to include more metrics modules without increasing the waiting time for the results or upgrading the hardware.

Distributed architecture

Continuing with the optimization, we are planning to make the architecture of the platform a distributed architecture. This will allows us to parallelize tasks, distribute the processing load and run the system using less powerful hardware.

Graphical interface

We are planing to develop a graphical interface for the system that will allow the platform to show the data in the system in a user-friendly way, using charts and to interact with the system to load or manipulate data.

APPENDIX A

Docker Installation and Deployment

A.1 Installation of Docker

In order to install Docker, we require a 64-bit version of our compatible Operating System. Docker is currently supported in Linux, Windows and OS X. We will focus this section in Linux installation, Ubuntu 14.04 in particular:

1. Update your apt-sources:
   (a) Log into the machine as a user with **sudo** or **root** privileges.
   (b) Open a terminal window.
   (c) Update package information, ensure that APT works with the https method, and that CA certificates are installed.

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates
```

   (d) Add the new GPG key.

```
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --
    recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

(e) Open the /etc/apt/sources.list.d/docker.list file in your favourite editor. If the file doesn't exist, create it.

(f) Remove any existing entries.

(g) Add an entry for your Ubuntu operating system.

```
deb https://apt.dockerproject.org/repo ubuntu-trusty main
```

(h) Save and close the /etc/apt/sources.list.d/docker.list file.

(i) Update the APT package index.

```
$ sudo apt-get update
```

(j) Purge the old repo if it exist.

```
$ sudo apt-get purge lxc-docker
```

(k) Verify that APT is pulling from the right repository.

```
$ apt-cache policy docker-engine
```

2. Install the linux-image-extra package:

(a) Open a terminal on your Ubuntu host.

(b) Update your package manager.

```
$ sudo apt-get update
```

(c) Install the recommended package.

```
$ sudo apt-get install linux-image-extra-$(uname -r)
```

3. Install Docker:

(a) Log into your Ubuntu installation as a user with sudo privileges.

(b) Update your APT package index.

```
$ sudo apt-get update
```

(c) Install Docker.

```
$ sudo apt-get install docker-engine
```

(d) Start the docker daemon.

```
$ sudo service docker start
```

(e) Verify docker is installed correctly.

```
$ sudo docker run hello-world
```

4. Install Docker Compose.

   (a) Install python-pip (if not already installed).

   ```
   $ sudo apt-get -y install python-pip
   ```

   (b) Install Docker Compose using pip.

   ```
   $ sudo pip install docker-compose
   ```

5. OPTIONAL: Create a Docker group.

   (a) Log into Ubuntu as a user with sudo privileges.

   (b) Create the docker group.

   ```
   $ sudo groupadd docker
   ```

   (c) Add your user to docker group.

   ```
   $ sudo usermod -aG docker ubuntu
   ```

   (d) Log out and log back in.

   (e) Verify your work by running docker without sudo.

   ```
   $ docker run hello-world
   ```

A.2 Deploying SCANER

Once Docker has been installed in the computer, the system is ready to launch SCANER.

1. Open a terminal on your host.
2. Navigate to the directory where SCANER is located.
3. Build the Docker Compose file in the directory.

   ```
   $ docker-compose build
   ```

4. Start the system.

   ```
   $ docker-compose up
   ```

Bibliography

[1] S. Moturu, *Quantifying the trustworthiness of user-generated social media content.* Arizona State University, 2009.

[2] X. Hu and H. Liu, "Text analytics in social media," in *Mining text data*, pp. 385–414, Springer, 2012.

[3] E. Otte and R. Rousseau, "Social network analysis: a powerful strategy, also for the information sciences," *Journal of information Science*, vol. 28, no. 6, pp. 441–453, 2002.

[4] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[5] D. Knoke and S. Yang, *Social network analysis*, vol. 154. Sage, 2008.

[6] J. Golbeck, *Analyzing the social web.* Newnes, 2013.

[7] N. M. Tichy, M. L. Tushman, and C. Fombrun, "Social network analysis for organizations," *Academy of management review*, vol. 4, no. 4, pp. 507–519, 1979.

[8] P. Goyal and S. Diwakar, "Data mining and analysis on twitter," tech. rep., 2012.

[9] H.-W. Shen and X.-Q. Cheng, "Spectral methods for the detection of network community structure: a comparative analysis," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no. 10, p. P10020, 2010.

[10] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[11] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "Scan: a structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 824–833, ACM, 2007.

[12] B. A. Huberman, D. M. Romero, and F. Wu, "Social networks that matter: Twitter under the microscope," *Available at SSRN 1313405*, 2008.

[13] B. Whalley, "7 crucial social media metrics and what they mean," aug 2011.

[14] P. Bray, "Social authority: Our measure of twitter influence," 2013.

[15] SoMaMFyC, "Follower wonk."

[16] I. Klout, "Klout."

[17] T. Noro, F. Ru, F. Xiao, and T. Tokuda, "Twitter user rank using keyword search," *Information Modelling and Knowledge Bases XXIV. Frontiers in Artificial Intelligence and Applications*, vol. 251, pp. 31–48, 2013.

[18] T. Noro and T. Tokuda, "Effectiveness of incorporating follow relation into searching for twitter users to follow," in *Web Engineering*, pp. 420–429, Springer, 2014.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web.," 1999.

[20] "Python & java: A side-by-side comparison," may 2009.

[21] M. L. S. C. NYC, "Python programming language advantages & disadvantages - mike levin seo consultant nyc," aug 2011.

[22] "What is docker?," oct 2015.

[23] Dataconomy, "Sql vs. nosql- what you need to know - dataconomy," oct 2015.

[24] TechRepublic, "Relational databases: Defining relationships between database tables - techrepublic," nov 2015.

[25] DB-Engines, "Relational dbms," may 2016.

[26] S. K. Gajendran, "A survey on nosql databases," *University of Illinois*, 2012.

[27] H. Vera, M. H. Wagner Boaventura, V. Guimaraes, and F. Hondo, "Data modeling for nosql document-oriented databases,"

[28] DB-Engines, "Document oriented database," may 2016.

[29] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 1, 2008.

[30] Neo4j, "What is a graph database? a property graph model intro," oct 2015.

[31] DB-Engines, "Graph-oriented databases," may 2016.

[32] DB-engine, "Db-engines ranking," may 2016.

[33] M. K. Ahsan Bilal, "Graph databases and orientdb," *Erasmus Mundus Master's Programme in Information Technologies for Business Intelligence*, 2015.

[34] A. Ronacher, "Opening the flask: How an aprils fools' joke become a framework with good intentions," Pycon, 2011.

[35] A. Ronacher, "Welcome to flask — flask documentation (0.10)," jan 2016.

[36] S. Cohen, "What challenges has pinterest encountered with flask?," mar 2016.

[37] "Swagger – the world's most popular framework for apis," jan 2016.

[38] O. A. group, "Open api initiative," jan 2016.

[39] A. Solem, "Celery: Distributed task queue," dec 2015.

[40] RedisLabs, “Introduction to redis,” dec 2015.

[41] J. Coyle, “Create your first docker container,” nov 2015.

[42] J. Coyle, “Using dockerfiles to build new docker images,” nov 2015.

[43] Docker, “Overview of docker compose,” nov 2015.