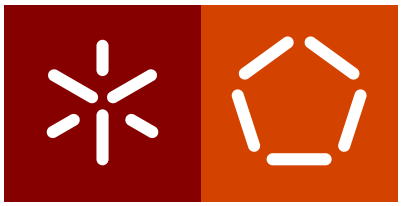
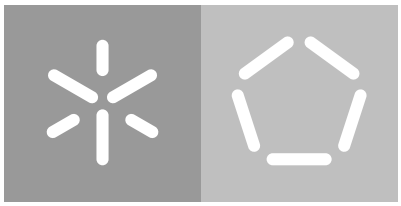


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Jorge Caldas

Analysis and Visualization of Dynamic Social Networks

September 2017



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Jorge Caldas

Analysis and Visualization of Dynamic Social Networks

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
Pedro Rangel Henriques
Alda Lopes Gançarski

September 2017

AGRADECIMENTOS

Quero acima de tudo agradecer à minha família, porque sem eles nada disto seria possível. Quero também agradecer aos orientadores Professor Pedro Henriques e Professora Alda Gancarski pelo apoio, pela curiosidade e pela dedicação com que sempre encararam este projeto.

ABSTRACT

*"**You can think of networks as vast fabrics of humanity, and we all occupy particular spots within the network.**" Nicholas Christakis*

This document represents the study developed under the master's thesis Analysis of Visualization of Social Networks, that overlaps two main scientific fields, sociology (more concisely social networks) and computer science, aiming at the design and implementation of a system for social network analysis.

First we present and identify the problem and challenge of building a system for social network analysis, we also mention the motivation, research hypothesis and goals. From here we start to search the scientific foundations for social networks, starting from the very basic theoretical concepts in Chapter two, traveling to the present to introduce Online Social Networks as the most well known application of this science. In Chapter 3 we go deep into details about Online Social Networks starting from exposing them with a more general discussion of this topic around the world and in the context of the Portuguese society, and exploring deeper seven of the most popular networks available on the Web at present. In Chapter 4 we provide the theoretical background regarding the scientific analysis of Social Networks, having always in mind with the tool we want to build; being Social Network Analysis a very broad field we decided to focus the research on this field on the master's thesis objectives, presenting only the concepts and tools that, in a certain, way built the path for Chapter 5, where we propose the system architecture.

Chapters 6 and 7 are more technical, we discuss and prioritize the requirements for our system and we present our technological choices based on a proof of concept built previously to the system implementation only for architectural and technological study of viability. In Chapter 8, using a case study, we present the developed tool, the final results and respective analysis. At the end we discuss the main contributions of this work, and where could we go from here; to close Chapter 9, we present different possible directions for future research work.

RESUMO

"Podemos ver as redes sociais como vastas fábricas de humanidade, onde cada um de nós ocupa um lugar específico." *Nicholas Christakis*

O presente documento representa o estudo desenvolvido no âmbito dissertação de mestrado sobre Análise e Visualização de Redes Sociais Dinâmicas conducente a esta dissertação, trabalho esse que resulta sobretudo da intersecção de dois ramos científicos, a sociologia e as ciências da computação, com o objectivo de propor o desenho e implementação de um sistema de análise de redes sociais.

Começamos por apresentar e identificar o problema e desafio de construír um sistema para análise de redes sociais, mencionando os nossos objetivos e motivação para o projeto. No segundo capítulo começamos a explorar as fundações científicas sobre redes sociais, apontando os conceitos mais básicos. No terceiro capítulo fazemos uma passagem pelas Redes Sociais Online, começando por fazer um levantamento mais generalista acerca do tema à volta do mundo e em particular no âmbito da sociedade portuguesa; após esta introdução, fazemos um estudo mais detalhado sobre sete redes sociais, atualmente acessíveis na Web, que selecionamos por julgarmos serem as mais usadas e expressivas deste gigantesco fenômeno social. No capítulo 4 apresentamos a base teórica da Análise de Redes Sociais, uma área bastante basta e complexa. Por ser uma área de grande dimensão e para a qual existem já disponíveis diversas aplicações, limitamos o seu estudo ao propósito deste trabalho de mestrado, sendo que o critério para exploração de alguns conceitos vê como base o facto de nos ser útil a sua compreensão para a implementação do sistema, cuja arquitetura apresentamos no capítulo 5.

Os próximos capítulos 6 e 7 são de um teor mais técnico; neles discutem-se e organizam-se por prioridades os requisitos do nosso sistema e decidem-se as tecnologias a usar, tendo a escolha sido feita com base na construção de uma pequena prova de conceito, também apresentada nestes capítulos, desenvolvida precisamente para validar a viabilidade da arquitectura e comparar abordagens de trabalho e tecnologias. Quase a fechar, no capítulo 8, apresentamos a ferramenta construída e os resultados obtidos, discutindo-se ainda um caso de estudo para ilustrar o processo e a ferramenta Socii. Para terminar faz-se, no capítulo 9, uma discussão acerca das conclusões mais relevantes a retirar deste projecto e seus contributos, bem como se discutem algumas possibilidades de trabalho futuro.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

In this Chapter we present an overview of the work being discussed along this master's dissertation. This Chapter presents the essential introductory topics. First we present the problem and context where this project is framed, then we expose the motivation, followed by the research hypothesis, which concisely describes the possible outcome of this project. Finally we list the goals of this project in a generic and simple way.

1.1 CONTEXT AND PROBLEM

In the mid 1950's sociologists introduced the term *Social Network* (*SN*), that despite being a familiar term for today general public because of the Online Social Network (OSN) platforms such as Facebook, Instagram or Twitter, it is a deeper and more mature concept. It was in the 2000's that much of the OSNs we know today started emerging, so it took at least ten years for people to adopt the concept and the new way of living, so today billions of people use these online platforms as channels for socializing, connect with each other and share their daily lives.

From the user's point of view we may consider that all the platforms offer a microscopic perspective from within the network, people have a public profile, and they can visualize their friend's profile (this is a typical scenario that we observe today in the majority of the OSNs), and normally have access to a timeline that displays friends activity. The point is that, to the users of these online platforms, it is not provided a mean to visualize and analyze their network structure in a more abstract and generalized sense, where users are given the opportunity to observe their social network from a macroscopic perspective, and, with that, all the metrics for measuring nodes and relationships within the network.

The problem that is being built in this section resides on general social structure observation and analysis. This dissertation aims to fill the gap or struggle that online social networks users have in understanding their network, how their relationships evolve along the time, what role they play within the network and how can they analyze and visualize their networks based on social properties such as mutual relationships, geographical position, personal tastes and preferences or hobbies.

Within the big data challenges, social network data analysis might be one of the biggest demands that we face today, because besides of dealing with tremendous amounts of data, we are dealing with unstructured data. The unstructured data derives from the diversity of these platforms known as OSNs, and unstructured data adds complexity to the challenge of analyzing social networks data. The major challenges related with big data and unstructured data comes after the data extraction.

The steps for data analysis and visualization

Next we present the steps through data extraction to data visualization, that generally represent the structure and flow of data analysis and visualization systems.

- Data extraction through social media *Application Program Interfaces (APIs)* or through web crawlers (also known as web scrappers);
- Data archiving, that requires, first of all, a careful selection of the relevant data to store; in order to have an efficient system that provides a good structure for data analysis, one needs to select data carefully;
- Data exploration, that requires the definition of what to do with the data, what are the applications that can be foreseen for the stored data, and how can the system digest and transform data in order to make it useful or interesting for the end users;
- Data visualization, that implies the choice of how to present/show the transformed data. Despite the science of visualization represents only a small part of the data scientist work, it has a huge impact on the end user, mainly when targeting a general audience.

1.2 MOTIVATION

As we have seen in the previous section, social media data analysis represents a major challenge for data scientists in every aspect, since the extraction all the way to the visualization. Despite representing a major technological challenge, social media data analysis has an additional motivation, that is the massive daily usage in every country across the planet making OSNs an universal tool for communication, such as radio or television but with the technological flavor of the 21st century.

OSNs, as we will see along this dissertation, are today a *"digital mineral"* in terms of exploration potential, we do not only pretend to have a generalist perspective of the analysis of data that flows within these platforms, we will try when appropriate, to demonstrate the concrete applications derived by analyzing social networks, these applications may go from

health analysis within social structures, to strategic marketing planning supported by the analysis of the already mentioned unstructured data.

1.3 RESEARCH HYPOTHESIS

With this master's project, we aim to prove that a software tool may be designed and implemented in order to actually improve the analysis of social phenomena, allowing not only sociologists but also the public in general to explore with greater detail the connections of individuals within a network, being OSNs the base of analysis for such a tool.

1.4 GOALS

The main goal of this project is to build a useful software tool in the context of social network analysis. Along the process of building and investigating, the following are some of the goals that are also very important to achieve:

- Understand the theory of *Social Networks (SNs)* in sociology;
- Understand how OSNs came to such a massive use nowadays;
- Perceive the roles of Online Social Networks in society and their potential applications in various fields;
- Study and analyze the most used Online Social Networks, learn how to interact with those systems and how to learn and profit from them;
- Design a system of analysis and visualization that matches the desired goals and requirements;
- Explore new technologies and choose the appropriate tools to build the specified system;
- Implement the system.

1.5 DOCUMENT STRUCTURE

In this section we will present how this document is structured, and concisely explain what to expect in each of the following Chapters.

We start by exploring some theoretical background on SNs in the perspective of sociology. In Chapter 2 we present a summary of the history behind SNs, and we review some of the fundamental concepts of SNs.

In Chapter 3 we explore Online Social Network (OSN), this is the *realization* of the SN concept of the 21^{st}. First we present a top level overview of OSNs, discussing the main concepts and common characteristics and defining some metrics to compare them (such as number of active and registered users); then we provide again some historical background followed by the detailed analysis of some selected OSNs considered the most used nowadays an so the more illustrative. We also talk about OSNs usage among Portuguese people and analyze the impact that OSNs have from the recent past till the present.

In Chapter 4 we discuss a very broad theme, *Social Network Analysis* (*SNA*), in the scope of our project. We talk about *Social Network Analysiss (SNAs)* basic concepts and metrics that are useful for network analysis as well as related scientific areas. We do an overview on SNAs software tools and libraries.

In Chapter 5 we present an architectural perspective of the project to develop along this master's thesis.

In Chapter 6 we define the system requirements and organize them according to their importance. We separate the requirements in two large groups, the first group are requirements related to the system back-end where we basically define how the system will extract, store and manipulate data in order to fulfill our end users' needs. The other group of requirements are the front-end requirements that define the applicational behaviors of Socii tool and what features will be truly visible by the end user.

In Chapter 7 we explain in detail how the system was implemented, we dive into some implementation code snippets were we explain some of the main events that occur within our system.

In Chapter 8 we present the final results. Here we look into some real use cases of Socii tool, where we explain all the tool implemented features and how to use them. We also have some specific case studies to simulate real world scenarios where Socii could be used.

Finally in Chapter 9 we present the project conclusion, we mention our main contributions and what work could be done from here on. We also mention some alternative approaches that we could have followed to implement the system and what obstacles we found along the way.

2

SOCIAL NETWORKS IN SOCIOLOGY

Nowadays, it is hard to find something that is not organized as a network, if one tries to understand something about the world around us, then definitely one needs to know something about networks.

Curiously, if we look up the term SN in the (Dictionary, 2002), we may face the following:

> *"a website or computer program that allows people to communicate and share information on the Internet using a computer or mobile phone"*

But, even if today we automatically think in SNs as websites (or web applications), deep down we know that, when talking about SNs, we refer to a much more broader term, that said, we may consider a SN as the following:

> *"A social structure made of nodes that are generally individuals or organizations. A social network represents relationships and flows between people, groups, organizations, animals, computers or other information/knowledge processing entities. The term itself was coined in 1954 by J. A. Barnes."* (Beal, 2016)

One may say that networks work like pipes, and through them things flow, from individual to individual inside the network. Through networks, big institutions can organize themselves, and actually add value to society despite the large number of individuals.

2.1 ORIGINS OF SOCIAL NETWORKS

> *"The network concept is one of the defining paradigms of the modern era."* (Kilduff and Tsai, 2003)

The network concept is broadly used across multiple fields of study, including, physics, biology, linguistic, anthropology, mathematics, computer science and more recently computer networks.

But why is the network approach so adopted in such diversification fields? The answer is because networks allows us to capture the interactions of any individual unit within the larger field of activity to which the unit belongs (Kilduff and Tsai, 2003).

Before reviewing the concept of network (Section 2.2), it is important to talk about it in a sociological perspective.

2.2 SOCIOLOGY PERSPECTIVE

> *"(...) many people attribute the first use of the term "social network" to Barnes (1954). The notion of a network of relations linking social entities, or of webs or ties among social units emanating through society, has found wide expression throughout the social sciences. (...)"* (Wasserman and Faust, 1994)

The SN concept has been around for many years now, maybe not in the exact same format than nowadays, we are familiarized with the "*web way*", in a manner of speaking, but in a more abstract sense, applied in real life within real connections. The term *"social network"* has first came into discussion in 1954, introduced by Barnes, J.A (Wasserman and Faust, 1994).

> *"Social relations in Bremnes, Norway, fall into three categories: relatively stable formal organizations serving many different purposes, unstable associations engaged in fishing, and interpersonal links that combine to form a social network and on which perceptions of class are based. In fishing situations, orders are given and obeyed; in the other social settings, consensus decisions are reached obliquely and tentatively."* (Barnes, 1954)

In the above citation, John Arundel Barnes, does a very well succeeded reflection about the relationships of the people from Bremnes (Norway).

The author points out that relations can form organizations for serving a specific purpose, and today we clearly see that the chosen path of SNs and also OSNs, was to narrow down SNs to very specific purposes, such as professional networks. So one may say that John Arundel Barnes not only coined the term SN, but also was one of the first who described **interest-based social networks**.

2.3 FUNDAMENTAL CONCEPTS

The concepts listed below are of key importance and are the basis of comprehension of SNs (Wasserman and Faust, 1994).

- *Actor* - It is important to understand the linkages among social entities and the implications of these linkages, these social entities are described as actors. Actors are discrete individual, corporate, or collective social units.
- *Relational Tie* - Actors are linked to one another through *social ties*. The type of ties may be extensive, and describe the nature of the connection. Some example of ties:
  - **Evaluation** of one person by another;
  - **Transference** of resources (business transactions);
  - **Association** (to social event or cause);
  - **Behavioural** interactions (communicating);
  - **Moving** between places or statuses (migration, social or physical mobility);
  - Others may be: physical connection (roads, rivers), formal relations (authority), biological relationship.
- *Dyad* - The most basic relationship that can be established is a dyad, a connection between two actors.
- *Triad* - A relation established between three actors. Many studies included breaking SNs down to small groups (triads), this allowed a more clear conclusion about the transitivity of the connections.
- *Subgroup* - It defines any subset of actors in a SN (conceptually, subgroups come after dyads and triads). We may define a subgroup of actors as any subset of actors and respective ties among them.
- *Group* - A finite set of actors who for conceptual, theoretical or empirical reasons are treated as a finite set of individuals in which network measurements are made. The specificity of a group, and what differentiates it from subgroups is that groups, consist in collections of actors on which ties are to be measured. One must be able to argue empirical, theoretical or conceptual criteria that actors within a group belong together in a less or more bounded set.
- *Relation* - A collection of ties of a specific kind among members of a group is called a **relation** (e.g. a connection in *LinkedIn* is a relation while evaluating our connections of sending them messages are ties).
- *SN* - At last, with the definitions of actor, group and relation, a SN consists of a finite set or sets of actors and the relation or relations defined on them. The presence of relation information is critical and a defining feature of a SN.

Next, we present two more advanced and abstract concepts but still fundamental concerning SNs in the context of this project.

Homophily

In a New York Times Magazine article (Retica, 2006) it is mentioned that the term *"homophily"*, was coined in the 1950s by sociologists and in a more literal sense it means *"love the same"*. This term emerges from the natural tendency we have to link to other individuals that are similar to us.

Quoting the sociologists (McPherson et al., 2001), ***"Similarity breeds connection"***, basically similarity is considered a generator of connections among individuals, being the result of this phenomena homogeneous SNs.

The term *homophily* has been cited in the perspective of many different themes, from teenagers choosing friends who drink and smoke similar amounts, or in explaining how homophily influences the matches of partners in online social dating, this proving that one likes, most of the time, someone like oneself, on or offline (Fiore and Donath, 2005).

From another point of view, this trend could be seen as a threat to diversity and globalization. It is said that diversity can be a synonym of power, when bringing different cultures and different ways of thinking together we could achieve great things, but homophily is already a cemented concept/pattern that sociologists observe among SNs, and maybe we could find ways to battle in favor of diversity, or maybe homophily is a fundamental property in order to structure society.

Heterophily

In order to complete the previous presented concept (*homophily*), we now present the opposite that is *heterophily*, that translates in literally the opposite idea, being *heterophily* the trend of individuals belonging to diverse groups thus connecting with different people.

2.4 Abstraction and Generalization

In a more abstract sense networks are merely abstractions that are originated by the generalization both of individuals, and relationships.

> *"When we study social organization of a simple society, we aim at comprehending all the various ways in which the members os the society systematically interact with one another. For purposes of analysis we treat the political system, the pattern of village life, the system of kinship and affinity, and other similar areas of interaction as parts of the same universe of discourse, as tough they were of equal analytical status, and we strive to show how the same external factors, principals of organization and common values influence these different divisions of social life. "* (Barnes, 1954)

In the above citation, the author describes a generalist approach on analyzing social networks. The two main characteristics of this approach are **generalization** and **abstraction**. First generalization because we are trying to simplify reality by minifying different kinds of connections (political, affinity etc.), this will allow us to treat networks as part of a world where they can fit in the domain of the exact sciences, being mathematical the way networks express themselves in order to measure metrics and behavior analysis.

Abstraction comes naturally in the way as the process of generalization takes laces, we could see abstraction and generalization as synonym in this specific case, but it also may be seen as a tool to see through the generalization process. Also fitting (at least try) networks and their analysis within the domain of exact sciences, requires the abstraction of the generalization that toke place before. In Chapter 4 we will cover with much more detail the field known as SNAs, that is responsible of deriving conclusions from analyzing social structures.

3

ONLINE SOCIAL NETWORKS

People need to be connected to other people, and the urge for connection brings to us what today are known as OSNs. These web sites allow us to define a profile as an individual, and to share and visualize content with other individuals in the network, therefore connecting.

> *"We define Online Social Networks as web-based services that allow individuals to construct a public or semi-public profile within a bounded system, articulate a list of other users with whom they share a connection, and view and traverse their list of connections and those made by others within the system. The nature and nomenclature of these connections may vary from site to site.* (Ellison et al., 2007)

OSNs have been around for more than a decade now, but these systems have gain world wide popularity since the global adoption of platforms such as Facebook, Youtube or Twitter, which are platforms that are today massively used across all cultures and age groups, and represents a paradigm shift on social interaction that we do not yet fully understand.

The earlier referenced OSNs, belong to the top of the most visited web sites in the world, that's because these systems not only represent a new way to keep in touch with friends, but also represent for many, a new way of living, basically we live in network.

In this Chapter we are going to explore OSNs, their history, how are these systems are being adopted among Internet users, and for some OSNs, a more detailed and deep study will be conducted for they are important objects of study of this master's thesis.

But first, with intent of obtaining a macroscopic perspective of the different OSNs in the Internet, what they offer that makes them different from one to another causing many of the users using multiple OSNs at the same time, we present next a table featuring some of the most used OSNs.

Name	Year of launch	Registered Users	Active Users	Description/Purpose
Facebook	2004	>1 712 000 000	1 712 000 000	**General**. Photos, videos, blogs, apps.
Google+	2011	1 600 000 000	300 000 000	**General**. Google+ is an interest-based social network that is owned and operated by Google.
Youtube	2005	>1 000 000 000	1 000 000 000	Allows billions of people to discover, watch and share originally-created videos. Provides a forum for people to connect, inform, and inspire others.
Qzone	2005	>652 000 000	652 000 000	**General**. It allows users to write blogs, keep diaries, send photos, listen to music, and watch videos. It's only available in Chinese.
Twitter	2006	645 750 000	313 000 000	**General**. Micro-blogging, RSS, updates.
Tumblr	2007	>555 000 000	555 000 000	Microblogging platform and social networking website.
Instagram	2010	>500 000 000	500 000 000	A photo and video sharing site.
LinkedIn	2003	>450 000 000	106 000 000	Business and professional networking.
Sina Weibo	2009	300 000 000	282 000 000	Social microblogging site in mainland China.
VK	2006	249 409 900	100 000 000	**General**, including music upload, listening and search. Popular in Russia and former Soviet republics.
Reddit	2005	234 000 000	120 000 000	Social media, social news aggregation, web content rating, and discussion website.
Vine	**2013**	200 000 000	100 000 000	Short-form video sharing service where users can share six-second-long looping video clips.
Pinterest	2010	176 000 000	100 000 000	The world's catalog of ideas. Find and save recipes, parenting hacks, style inspiration and other ideas to try.
Flickr	2007	112 000 000	92 000 000	Helping people make their photos available to the people who matter to them. Enable new ways of organizing photos and video.
Meetup	**2002**	27 590 000	-	World's largest network of local groups. Meetup makes it easy for anyone to organize a local group or find one of the thousands already meeting up face-to-face.
Couchsurfing	2004	12 000 000	-	Couchsurfing connects travelers with a global network of people willing to share in profound and meaningful ways, making travel a truly social experience. Is commonly used by travelers to find free hosts across the globe.
ResearchGate	2008	>11 000 000	-	Built by scientists, for scientists. Connect the world of science and make research open to all.

Table 1: Table describing most used OSNs.

Table 1 lists the most used and popular OSNs, **ordered by the estimated number of registered users** [1] [2]. Also notice that, for those OSN where the number of registered users is unknown, we will assume that it is a larger value than the monthly active users represented by the column *Active Users*.

The first obvious comment on the listed OSNs is that general purpose OSNs have more users (social networks with the word *General* in bold), being Youtube an exception, since it is not a general purpose OSNs, neither is focused on individuals, it is build around **social objects**, the videos.

The grey scale in the first column of Table 1 divides OSNs in three groups: the first and smallest, the 1 billion or more users OSNs; the second the OSNs with less than 1 billion users and more then 100 million; finally, the third group, OSNs with less then 100 million users. At this point, we begin to observe that **the narrower purpose OSNs** such as ResearchGate (mainly for researchers) or Couchsurfing (mainly for open minded travelers), **have a smaller number of registered users**, which is expected since the target audience is also smaller.

Other OSNs not listed in Table 1, but still worth mentioning include **Classmates** (helps users finding classmates from kindergarten, primary school, high school, etc.) known for being one of the first OSNs, since it was launched in 1995, and **Ask.fm** (allows users to interact with other users asking and answering questions (revealing identity is optional)).

An important note on the listed OSNs in Table 1 is that only Qzone, Vine, Couchsurfing and ResearchGate don't provide any web APIs to fetch data or publish content, while all the others offer a wide variety of web services for developers to consume and use as they please, of course within the terms and policies of use of each OSN.

3.1 History of Online Social Networks

Although the first platform possessing some of the main characteristics that define OSNs (Ellison et al., 2007), as we can see in Figure 1, the first recognizable OSN launched in 1997 as we can observe in the Figure 1 (Ellison et al., 2007). *SixDegrees.com* allowed users to create personal profiles, connect with friends and consult friends of friends lists. The profile feature came from the online dating sites and online communities, while the surfing through register users in the network and consulting friends was an existing feature in Classmates.com. *SixDegrees.com* was the first to combine these features.

SixDegrees promoted itself as a tool to help people to connect, but in 2000, it became an unsustainable business and the service closed. At the time the creators conclude that *SixDegrees* was a service that was very ahead of its time.

1 https://www.statista.com/
2 http://expandedramblings.com/

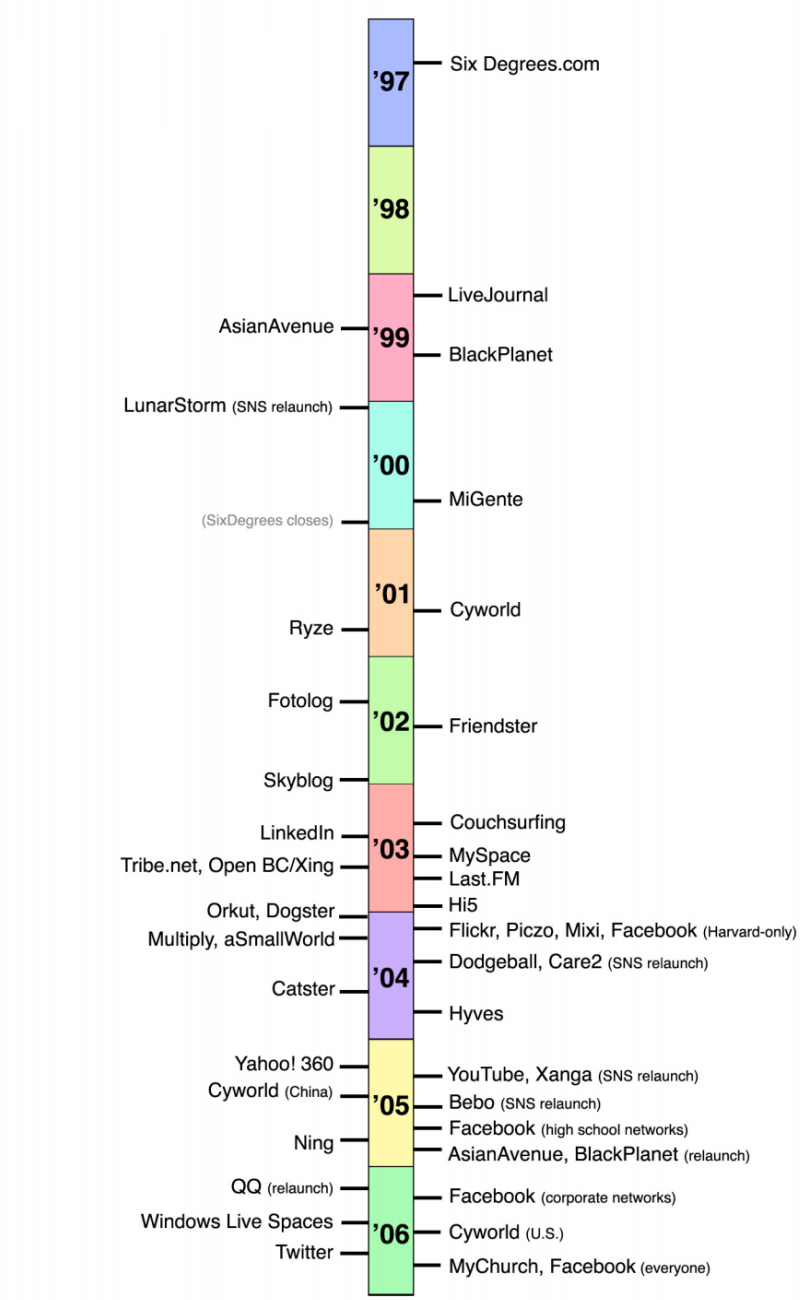


Figure 1: Launch dates of major OSNs.

Until 2002 many OSNs have emerged, but still incapable of projecting themselves at a global scale. As we can observe in the timeline of Figure 1 (Ellison et al., 2007) from 2002 and 2005 the *big players* came to existence, in these periods, OSN such as Friendster, LinkedIn, MySpace, Hi5, Facebook and Youtube were born, shaping the business, cultural and research landscape.

3.2 PORTUGUESE PEOPLE AND ONLINE SOCIAL NETWORKS

From Table 1, we get a good overview on OSNs usage among modern society. In this section we do a deep exploration of the most adopted OSNs by portuguese citizens, and compare then with the more global scenario presented in Table 1, also, other interesting facts will be revealed where appropriate.

A recent study (Marktest, 2016), reveals portuguese relationship with OSNs. This study, has been made by *Marktest Consulting* since 2011, with the goal of know the notoriety, utilization, opinion and habits of portuguese concerning social networks. The study information was collected through online interviews. The sample was built from 819 interviews from individuals with age between 15 and 64 years, living in Portugal and using OSNs in a daily basis.

Some of the most interesting facts revealed in this study, relative to the participants are:

- 94% has a Facebook account and 43% a Youtube account;
- 21% has abandoned a social network in the past year;
- 27% considers that their dedicated time to social media has increased;
- 67% follows celebrities and 62% follows brands;
- 87% is used to watch videos in social networks.

These are indeed interesting conclusions, but what about the top used OSNs, **the most used are (by order): Facebook, Youtube, Google+, LinkedIn, Instagram and Twitter**.

Relatively to past studies, Facebook has maintain the top position, maintaining a grow tendency that has been standing out in the past years.

Going back to Table 1, we may now comment the usage of OSNs by portuguese people comparing it to the global scenario. As one may notice Facebook still rules users preferences within portuguese people.

Concerning to global time related usage statistics, **portuguese spend 91 minutes a day with social networks** (Marktest, 2016), 68% considers that this is the ideal time to spent with social media, despite 1 in each 4 saying that in the past year has dedicated even more time to them. Even if people spent more than one hour and an half in these platforms, the study concluded that **67% of the users that visit OSNs several times a day only 41% does daily publications**.

The prime time for using OSNs is between 8pm and 10pm, being the smartphone the most used device in this time. Also in this short period the featured OSN is Facebook, the majority of the interviewees say that is the most credible site, the one that provides better and more useful information, the most interesting and addictive.

3.3 EXPLORING SPECIFIC ONLINE SOCIAL NETWORKS

In this section we are going to explore in greater detail some of the OSNs presented in Table 1. The selection of the social networks was not aleatory, we are going to study deeply the OSNs that gather some important characteristics, that will be of use in the future when we design the system for analyzing and visualizing social networks. First, the OSN must be accessible, this said, one must be capable of extracting information from the platform in order to analyze it. Second, the OSNs should preferably be the most diversified as possible, so that we can draw different types of conclusions deriving from different kind of analysis, for then give proof of the adaptability of the system to different OSNs. Considering the previous comments, these are the following OSNs that we think that as a group, best represents the intentions previously mentioned, so we will cover them with more detail (with no particular order):

- Facebook;
- Instagram;
- LinkedIn;
- ResearchGate;
- Pinterest;
- Twitter.

3.3.1 *Facebook*

Facebook is an OSN, created by Mark Zuckerberg in 2004, which started out by being an exclusive social network for Harvard students, but came later to spread across the country and the globe, having today more than one billion users.

Before diving into details of Facebook's domain, one must first point out some of its general aspects. Facebook basically allows anyone with a valid email address to create a public and personalized profile, we say personalized in terms of displayed content or information such as profile photo, name, work, homeland, education. The next fundamental step is to connect with other users, by sending friendship requests to other Facebook users (these are bidirectional relations). The base entity of the network is the user, but entities such as brands, companies can also be part of the platform, appearing normally in the form of page, being a page a public place inside the network with marketing or business related purposes (celebrities, public institutions also use pages as form of appearing in Facebook).

The next parts of this section will clarify the roles of these entities and their way of interacting with each other, also other important concepts will be presented.

Domain Model

In this section we explore the domain of Facebook represented in Figure 2 in detail, what are the pieces that conceptually build this platform, and how they relate. The schema in Figure 2 represents a macroscopic perspective among Facebook components and their organization.

There are two entities with bold labels in the schema, these are **User** and **Post**, being *User* the base entity in the network (the node in the network graph basically), and *Post* the most basic unit of content sharing in Facebook.

Facebook is interesting in terms of data gathering, because despite offering users' basic information and to whom users are related (*Friends* box), it has a collection of other interesting data such as the family relationships (*Family* box), geographical locations where the user lives, or visited locations (*Locations* and *VisitedPlaces* boxes respectively), and among other things, user information may contain the personal interests that were explicitly inputed by the user (*Likes* box).

In what concerns to user activity in the platform, the *Timeline* provides all the user Posts chronologically ordered, this is where Facebook dynamism takes place, users are constantly adding content to their timeline, it may be life related events or simply sharing other users posts linking content. The user feed (*Feed* box) represents a global timeline where the user can consult all the posts on his network (this is by default the user's landing page on the platform).

Facebook has, with time, become more then a user profile centralized network, it has invested in expanding its horizons, becoming the place where pages of brands, companies, organizations (media, political, non-profitable etc.), or places (cities, monuments, bars etc.) live (*Page/Local* box). This entities that are now cohabiting with users in the Facebook ecosystem, take advantage of the platform and its range to get their updates to most people as possible. The profile for these pages are in many ways different form the user's profile, it also has a timeline, but the about information and other details represent a smaller part of page's profiles, the most important metric for pages is its number of *likes* (*Likes* box), it represents the number of users in the network that follow the page, it might be users that simply have a certain relation with the entity or simply want to keep in touch by regularly receiving these entities updates in their Facebook walls [3].

Other Facebook entities not yet mentioned, are events (*Event* box). These are events inputed in the platform that allow users to keep updated about relevant events happening mainly in their area. Users can tag the event as *interested in*, showing their friends the will of participating in some event, or they can simply reject the event. Users also can confirm participation on events showing their network that they will be present. Events keep three

3 Facebook wall an area where users can see the posts of their friends and/or liked pages, in a chronological order

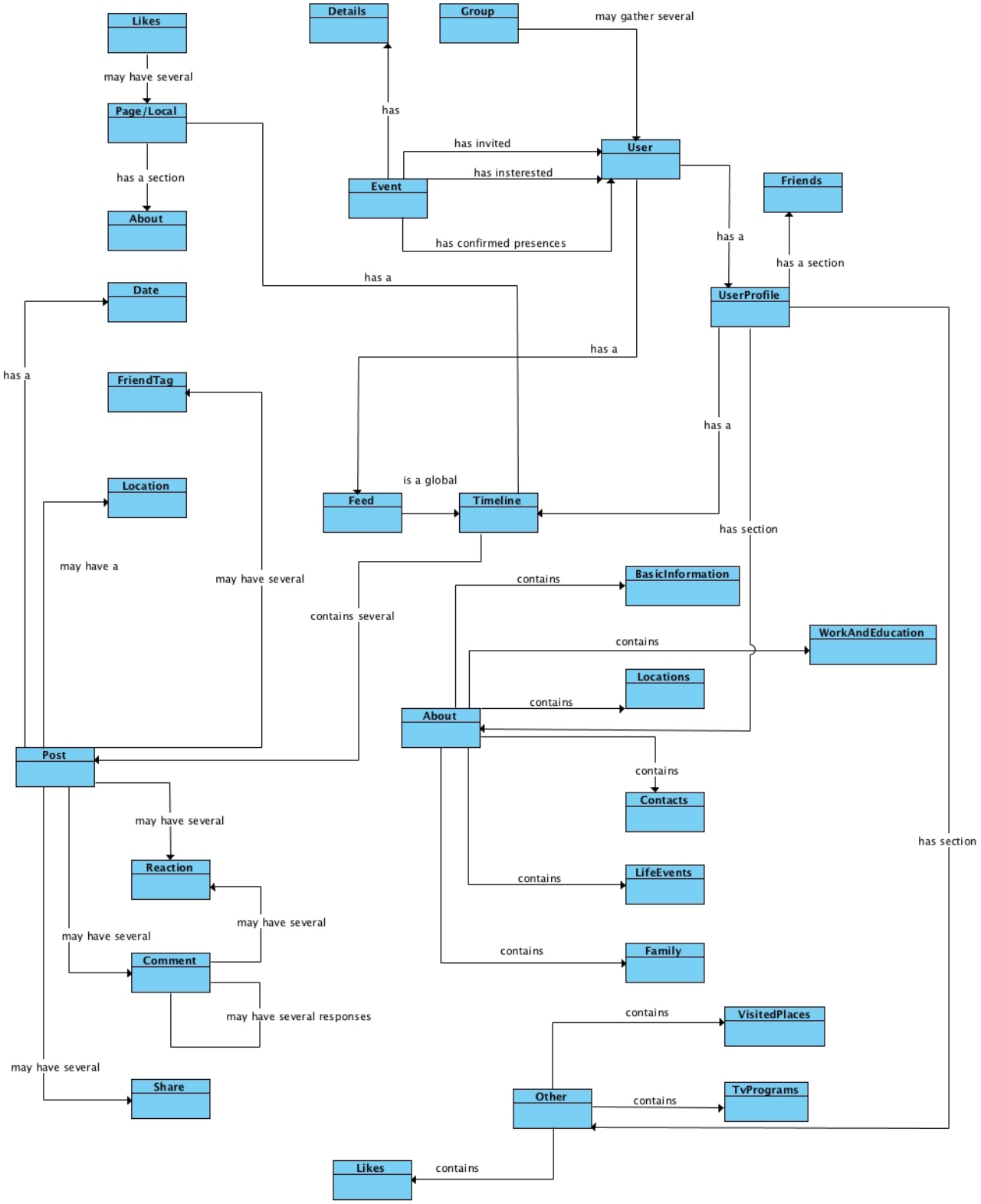


Figure 2: Facebook domain model schema.

separated counters for users, they count the number of invited users, number of interested users and number of confirmed users (these relations are expressed as links between the *Event* box and the *User* box).

In Facebook is also possible to join groups of users, this groups may be public or private, and they generally are focused on a specific matter, or gather users from one same institution or organization (e.g. Facebook group of students of the University of Minho). Having this feature of groups, clustering users by they interests one may say that groups, some way, transform Facebook in a *"multi interest-based OSN"*.

Facebook Graph API

Facebook has today several software *kits* for developers to interact with the platform in the most diversified and imaginable ways. Facebook developers offer a range of variated software products that vary from monetization programs, that focus on how to make users profit from Facebook, Analytics to developers who have their apps embedded in the Facebook platform understand their audience and the performance of their apps, etc. [4].

In this master's thesis context, the relevant software that Facebook has available is the Facebook Graph API. This API basically allows developers to collect information from Facebook such as posts, photos, videos, pages etc. The common scenarios for using the Graph API are the following[5]: determine whether two people are friends on Facebook; publishing new status and updates, uploading content (photos, video etc.); sharing links. But in this project what we seek is to build the most biggest and detailed network as possible, with analysis and visualization purposes in mind.

For building the network, fetching users friends information is crucial, this was possible until Facebook Graph API v2.0 (through the router */me/friends*), where developers could actually retrieve enough friends' information to build a network (social graph) from there. From v2.0 on, to achieve what was explained before, one must request a special permission called **user_friends** from each user. The permission **user_friends** is no longer included by default in every login. This change breaks down the possibility of gather Facebook information via its Graph API, this said, we need in the future to look up alternative paths to extract data from Facebook.

3.3.2 *Instagram*

> *"Since the beginning, Kevin has focused on simplicity and inspiring creativity through solving problems with thoughtful product design. As a result, Instagram has become*

4 https://developers.facebook.com/products/
5 https://developers.facebook.com/docs/graph-api/common-scenarios

> *the home for visual storytelling for everyone from celebrities, newsrooms and brands, to teens, musicians and anyone with a creative passion."* [6]

Similarly to Facebook we are going to explore Instagram in the same way. Instagram was originally developed by Kevin Systrom and Mike Krieger, and launched in 2010, only for iPhone devices. Within a year Instagram was able to gather around 10 million of users. Later, in 2012 Facebook acquired Instagram for approximately 1 billion dollars.

As already mentioned in Table 1, Instagram does not belong to the group of general purpose OSNs, instead, Instagram specially focused on photo and video sharing, building a global community that shares more than 95 million photos every day.

According to Instagram official page [9], since the very beginning Instagram was a very simplistic platform, being this characteristic reflected on its domain model.

Domain Model

Figure 3 represents the domain model of Instagram, and as we can observe, simplicity is the essence of this platform, since this diagram is far more a realistic representation of Instagram than Figure 2 is a representation of Facebook, and this may be why Instagram is so massively adopted by users on the Internet, because it goes directly to the point, focusing mainly on sharing activity, offering a real easy and simple user experience.

Now concerning to the domain model, we can see that a user and its profile (*User* and *UserProfile* boxes) are very simple entities, because a user's profile is only its biography (*Biography* box), relationships (*Followers* and *Following* boxes) and the user's posts, that despite being chronologically ordered, do not intend to form any kind of timeline such as Facebook, instead it represents more the concept of a wall with frames hanged on it.

In Instagram the landing page, represents a timeline (*Timeline* box) with posts from users we follow. Regarding to posts (*Post* box), one can comment posts (*Comment* box), but one cannot react or respond to comments (this preserves simplicity even more, for nested comments represent a complex part of OSN such as Facebook), and react to them by the *like* reaction (*Like* box).

Instagram API Platform

In consequence of its simple domain, Instagram API Platform provides simple and useful end points for programmatic publishing, and for network discovering, as far as concerning to this project, the latter utility is more of interest. Instagram allows to get users, their relationships and also the media shared content (posts).

Similarly when exploring Facebook Graph API, we now found also very intimidating

6 https://www.instagram.com/about/us/

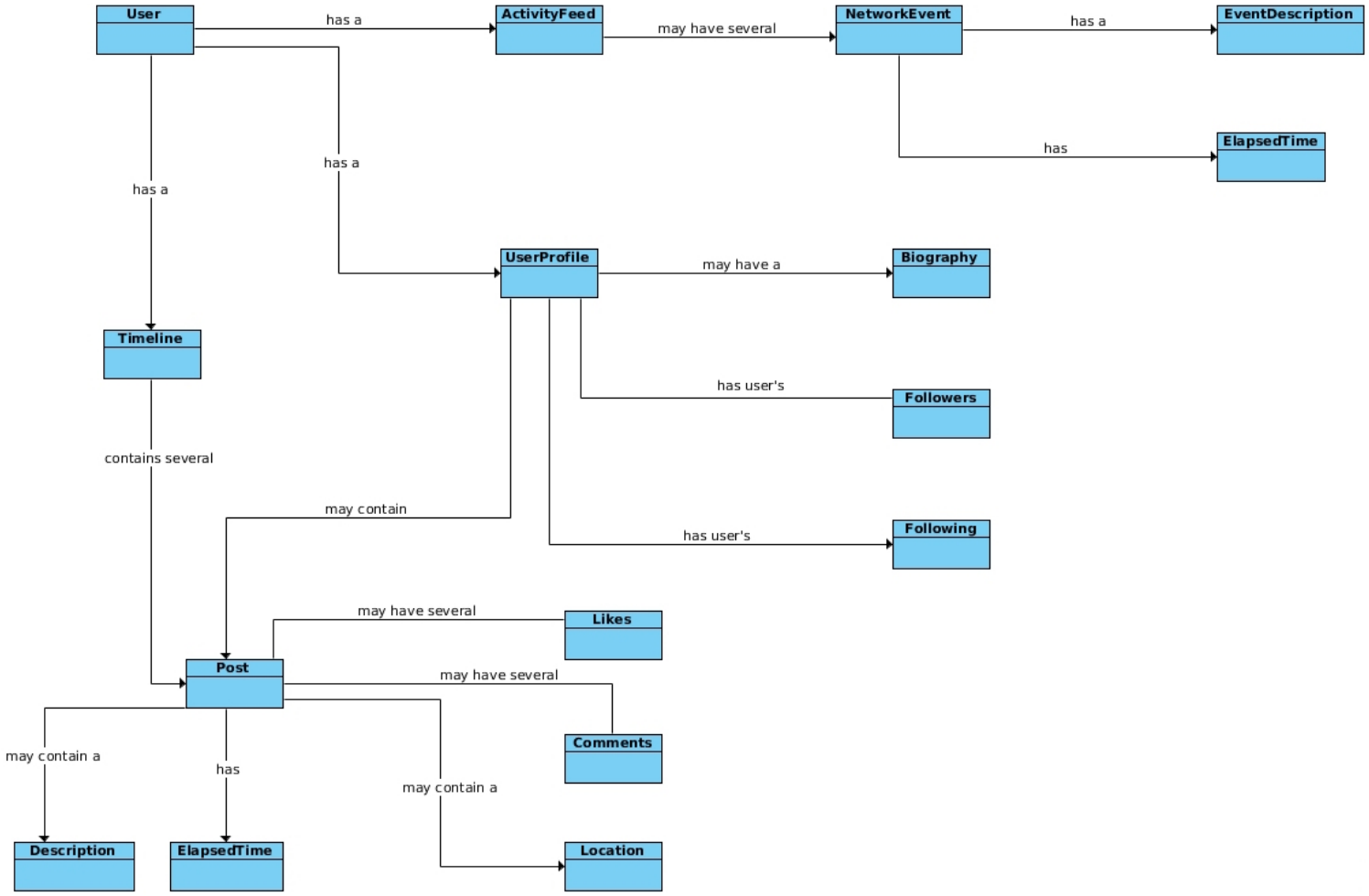


Figure 3: Instagram domain model schema.

restrictions for the purpose of this project, this restrictions include limited rate[7] of 500 API requests per hour, and end point specific limitations that allow only to perform 30 requests per hour to getting users' relationships data.

3.3.3 *LinkedIn*

Moving on to the next OSN we now have LinkedIn. LinkedIn was launched officially on May 5 of 2003 [8], and by the end of that month, the network had already more than 4500 members. In 13 June of 2016 LinkedIn was acquired by Microsoft in an all-cash transaction valued at $26.2 billion (Guardian, 2016).

LinkedIn is an OSN that has a very narrow purpose, which is connecting professionals around the globe to make them more productive and successful.

7 https://www.instagram.com/developer/limits/
8 https://press.linkedin.com/about-linkedin

Domain Model

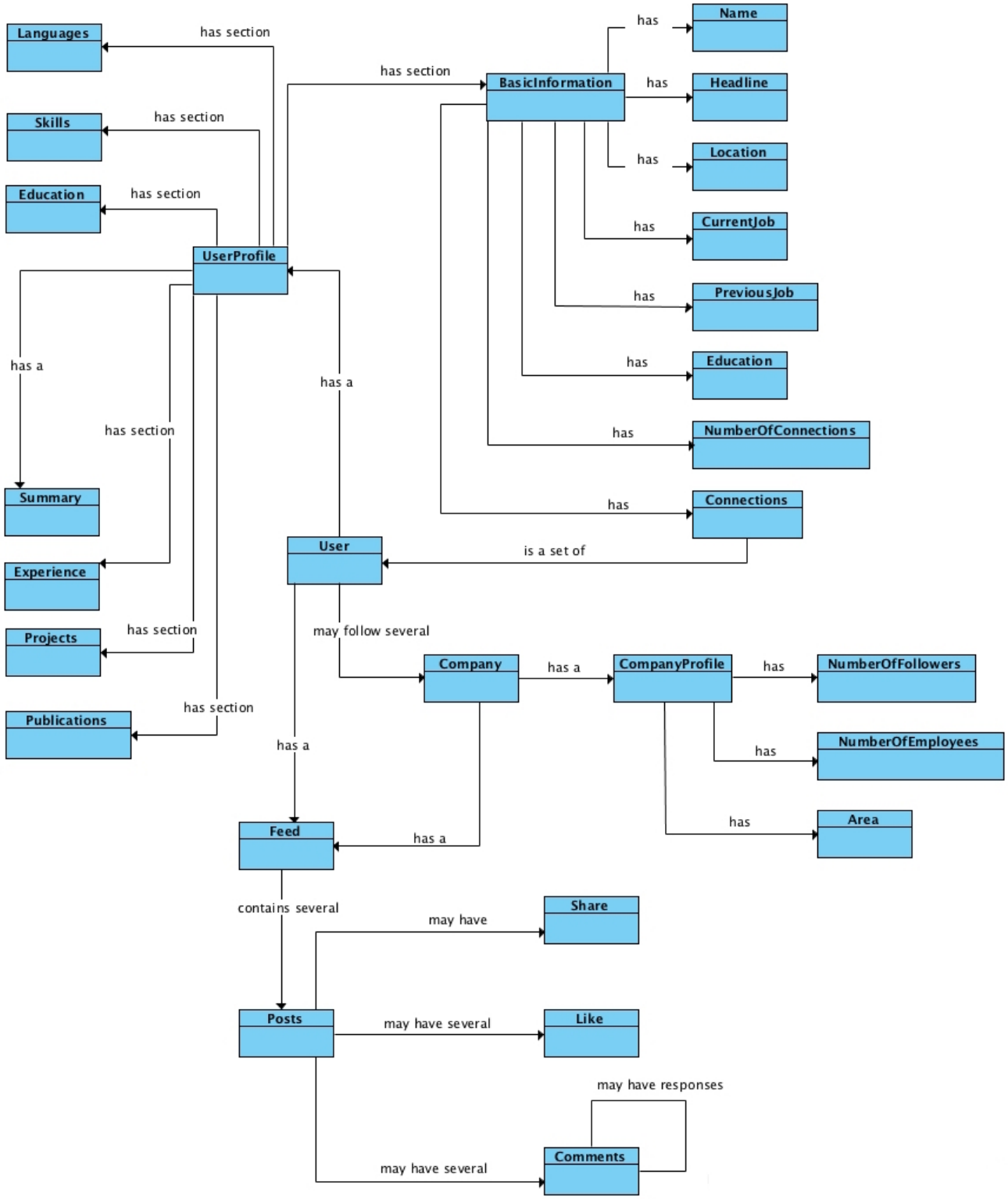


Figure 4: LinkedIn domain model schema.

Being a more purpose-oriented OSN and focused on the professional world, makes the LinkedIn platform more complex, even with a simplified representation of the domain model, as we can observe in Figure 4 it is schema [9] far more complex that Instagram, having more or a similar complexity comparing to Facebook.

In LinkedIn the user profile (*UserProfile* box) is very rich in terms of what is important for building an individual professional image (profile), starting by one individual's basic information (*BasicInformation* box) that has information like name, location and current and/or previous jobs. Then the user profile has several sections with very specific purposes such as professional experience (*Experience* box), languages (*Languages* box) or education (*Education* box), all this summed up give a very precise perspective of an individual's "professional appearence". At the bottom of the profile we have along with the professional recommendations and connections, the skills or expertise section (*Skills* box), this is one of the most attractive features in the LinkedIn platform. Skills in LinkedIn are a tagging system that allow user's to expose their expertise through their public profile and then receive feedback on them according to their ability on that specific skill, this is a very important and promising feature for matching user's profiles with job positions requirements.

LinkedIn's main entities are not only users, the industry is massively represented in this network too. Companies may have a company profile (*Company* and *CompanyProfile* boxes) where they present the company, containing basic information such as number of people following the company number of employees (giving the idea of the company dimension) and the area where the company fits (pharmaceuticals, technology etc.) (*NumberOfFollowers*, *NumberOfEmployees* and *Area* boxes respectively).

Other important concept of LinkedIn is the user feed where the user can chronologically consult a series of posts produced by their connections or by companies that their follow.

LinkedIn API

LinkedIn provides a REST API [10], but still similarly to the OSNs we have been studying is very limited. In what concerns to data retrieval, LinkedIn only allows the access to basic profile data, this is the data retried from the LinkedIn interactive REST console:


```
{
"firstName": "Daniel",
"headline": "Graduate Front-end Developer at Blip.pt",
"id": "k_yk8W37WH",
```


9 In the schema presented on Figure 4, much of the platform complexity was simplified in order to produce a simple domain, and to narrow down this analysis to the core components and concepts of LinkedIn.

10 https://developer.linkedin.com/docs/rest-api

```
"lastName": "Caldas",
"siteStandardProfileRequest":  {
  "url": "https://www.linkedin.com/profile/..."
}
```


As we can see from the above data sample, we only could fetch some data properties, that would not bring value in terms of network analysis.

3.3.4 *ResearchGate*

> *"Founded in 2008 by physicians Dr. Ijad Madisch and Dr. Sören Hofmayer, and computer scientist Horst Fickenscher, ResearchGate today has more than 11+ million members. We strive to help them make progress happen faster."* [11]

ResearchGate is an OSN built specifically for scientists, with the goal of easing the task of collaborative research around the globe. ResearchGate strikes to connect the world of science and make research open to all.

11 https://www.researchgate.net/about

Domain Model

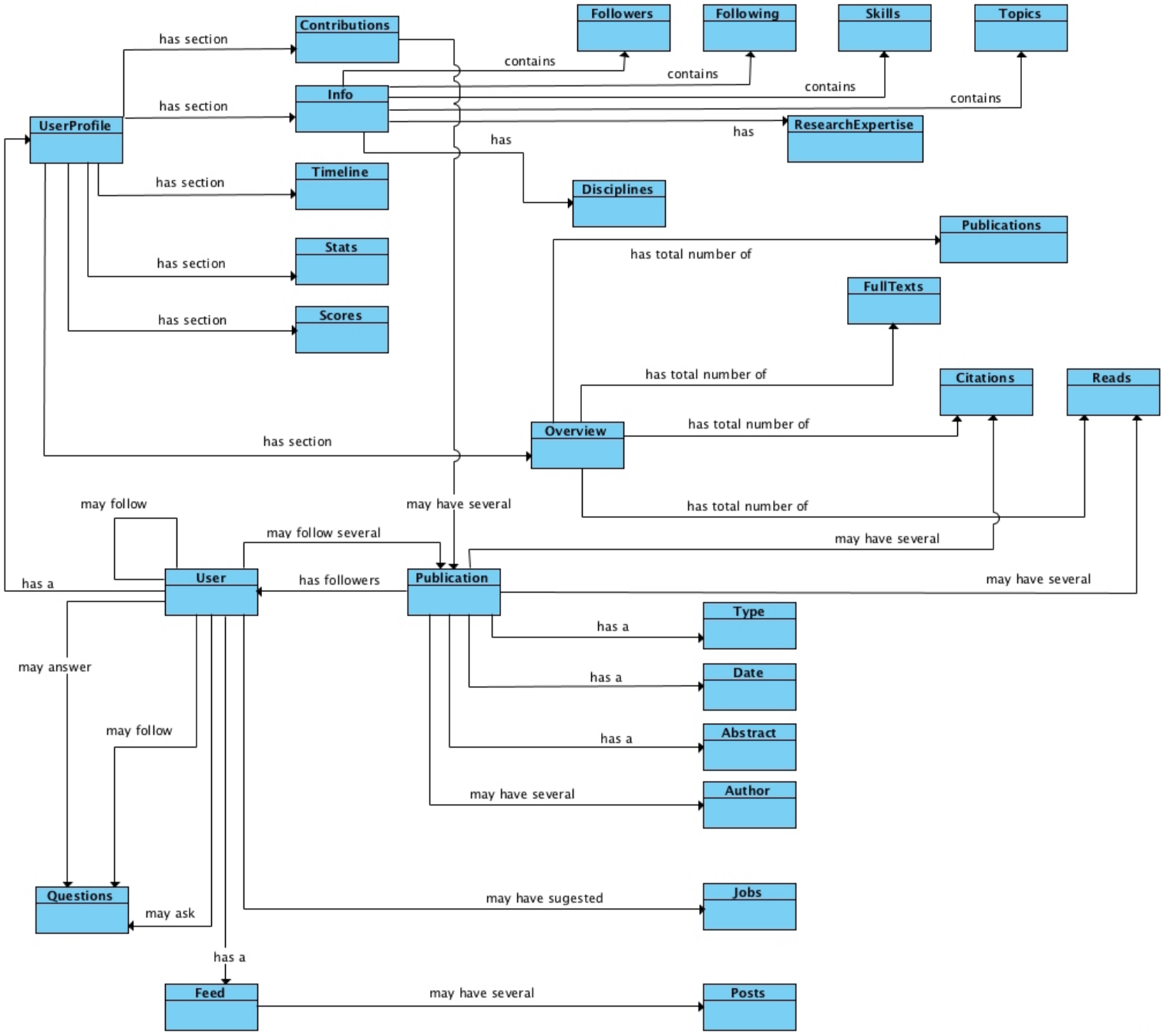


Figure 5: ResearchGate domain model schema.

Data Dictionary

Some terms on the schema presented in Figure 5 may be quite ambiguous due to the the specificity that they represent. In order to make the schema fully legible and before diving

into the domain model analysis, we present first, a small data dictionary detailing the terms one may found more ambiguous:

- **Scores** - This term represents a collection of metrics that evaluate the performance of a user based on his contributions and research experience. The user has also associated a global score;
- **Topics** - Topics represent the user's scientific areas of interest, ResearchGate uses topics to provide personalized suggestions;
- **Disciplines** - Represent more broad areas of the user education, expertise and interest;
- **Type** (*Type* box connected with the *Publication* box in Figure 5) - A type classifies a publication, this said, a publication may be an article, a book, a thesis, a conference paper etc. .

Domain Model Analysis

ResearchGate is a peculiar OSN that despite having connections between individuals, it has alongside connections between individuals and scientific publications, making the publication (*Publication* box) a social object, playing the same role that videos have in Youtube for example.

Like LinkedIn the user profile (*UserProfile* box), is very detailed and builds up a very clear image of the researches work, positions and areas of interest. The relations among users are bidirectional, following the followers/following (*Followers* and *Following* box) strategy like other OSNs such as Instagram or Twitter. Very simillarlly to LinkedIn, a user's profile has a skills (*Skills* box) section, where skills are expressed in the form of tags, the tag description is far more specific than LinkedIn tags, that may some times acquire very abstract or high level descriptions (e.g. Information Technology). In ResearchGate tags are very specific and are normally related with the user topics (*Topics* box) or disciplines.

Publications play along with the user a main role in ResearchGate. Normally publications have associated a type (already explained in the data dictionary section), a date, an abstract and may have one or more authors. The main metrics for Publications rating are the number of reads (*Reads* box) and the number of citations (*Citations* box) of that publication. The publications may also be followed by users that may have interest on particular publications.

Other concept of ResearchGate that raises the collaborative spirit among users, living up to the values that originated the platform, is the questioning system (*Question* box). Users may ask each other specific questions and have them answered by an expert on a specific scientific area, this opens up the possibility of having the best experts on a specific matter giving their opinion, thus the possibility of obtaining the *"best possible answer in the globe"*.

ResearchGate users' receive open jobs suggestions based on their profile, also user's have a post where they receive activity notifications of the people or publications that they are following.

API

Today ResearchGate does not provide any API for accessing its data or for any kind of interaction with the platform.

3.3.5 *Pinterest*

Pinterest **is the world's catalog of ideas** (Pinterest, 2016). Created by Ben Silbermann, Paul Sciarra and Evan Sharp and launched in 2010, Pinterest is a simple but yet very original OSN, instead of aiming for connecting people like Facebook or LinkedIn, it aims for inspire people through new ideas.

Domain Model

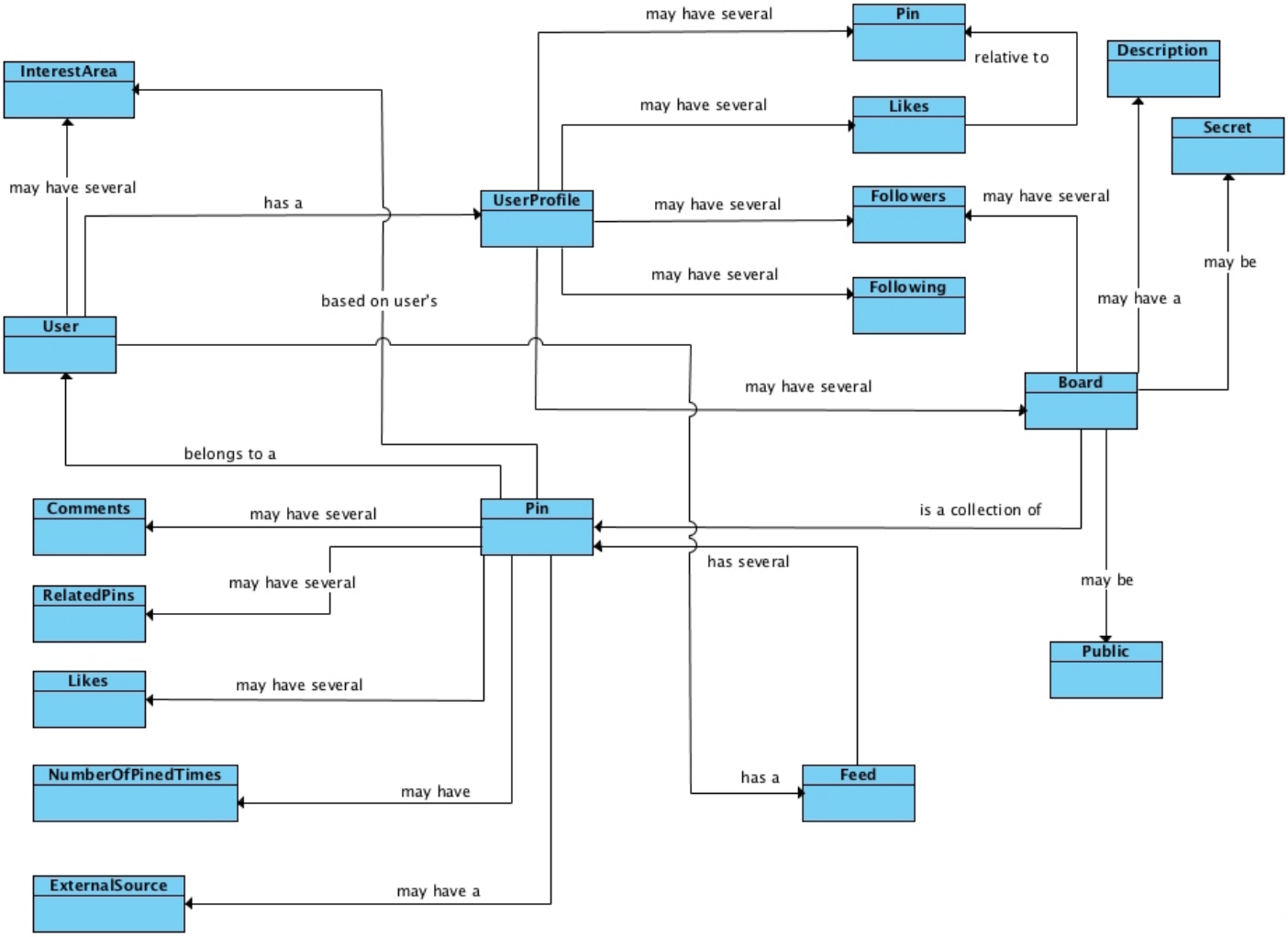


Figure 6: Pinterest domain model schema.

Data Dictionary

As one may notice from Figure 6, Pinterest introduces very particular concepts that may lack explanation, that is why we present first a small data dictionay before going through the analysis, as we did with ResearchGate on a previous section:

- **Pin** - A Pin is the basic unit of Pinterest, it represents an idea of some user, presented in some context (the board context), and it is presented to us with a picture;
- **Board** - As the name suggests, a board is a collection of pins. Boards are created from users to other users, and normally present pins within some context (e.g. travels, technology, food etc.). In Pinterest boards may be followed by other users;

- **NumberOfPinedTimes** - This entity is not entirely a Pinterest entity, instead it represents a relevant metric introduced to measure pins popularity, and it refers to the act of saving pins. Pins that are presented to the users may be saved (or *"pinned"*), and the number of times that users have saved a particular pin is expressed in Figure 6 by the box *NumberOfPinnedTimes*;

Domain Model Analysis

Pinterest introduces new concepts forming a very original OSN, because it's very different from others that we analyzed previously. Just as we seen in ResearchGate, where the domain model is build around a social object (the scientific publication), with Pinterest we have a similar scenario, where the concept of the platform is built around a different social object the Pin (*Pin* box), which also as a grouped perspective introduced by a group or collection of pins that are the boards (*Board* pin). Pinterest is basically a set of pins aggregated in boards that are explored in the platform accordingly to the user's interests.

Simmilarly to other networks (e.g. Instagram) Pinterest also has direct unidirectional connections between users that adopt the concept of *"follow/following"* (*Followers* and *Following* boxes). As user's can follow publications in ResearchGate, Pinterest users may follow boards, being then notified if some pin is added to that specific board.

In what concerns to Pins, they may be commented by users (*Comments* box), they also may be targeted by likes as posts in Facebook (*Likes* box). A particular point concerning to Pins is that they can have an explicit external reference, for instance, if some image is extracted by some other web site or from other OSN they can be explicitly referenced, and that same reference appears at the top of the pin along with its title (*ExternalSource* box).

Pinterest was the traditional concept of feed, but in this case, the feed represents a completely different concept compared to other OSN. First the content of the feed (pins) is not related with users we follow on the network, is instead related is our personal interests (*InterestArea* box) and second, they are not presented according to a chronological order, and visually they do not follow the standards of typical timeline/feed design, instead the different pins displayed on some user's feed, form some kind of board or catalog, like the ones people use to hang in walls and pin post-its on it.

Pinterest API

Pinterest provides a REST API (Developers, 2016) for interactions with the platform. The data restrictions follow Facebook politics, where the application that integrates Pinterest API can only fetch data for authenticated users. Pinterest provides endpoints to interact with users, boards and pins. Concerning the requests limitation, Pinterest offers a 60 minute sliding window where 1000 requests can be made by unique user token.

3.3.6 *Twitter*

One OSN that frequently is brought to discussion for being more of a *"news content generator"* is Twitter. Twitter is one of the most used OSN listed in Table 1, is basically a social networking microblogging service that allows their users to broadcast short posts (short because they're maximum size cannot exceed the 140 characters) called tweets. Twitter was created in March 2006 by Jack Dorsey, Noah Glass, Biz Stone, and Evan Williams, gaining fast worldwide popularity, Twitter has today more then 300 Million users according to Table 1.

Unlike many other social networks that have private or semi-public profiles with restrict policies concerning to external access to information within the network (e.g. LinkedIn, Facebook), Twitter default settings are public, making tweets spread more effectively across all social media, this particularity makes Twitter one of the most *"barrier-free*" OSNs. Of course that despite unregistered people may read tweets they cannot interact with them as Twitter users by linking, comment or *"retweet"*[12] them.

12 The act of retweet consists of sharing some existent tweet originated by another user.

Domain Model

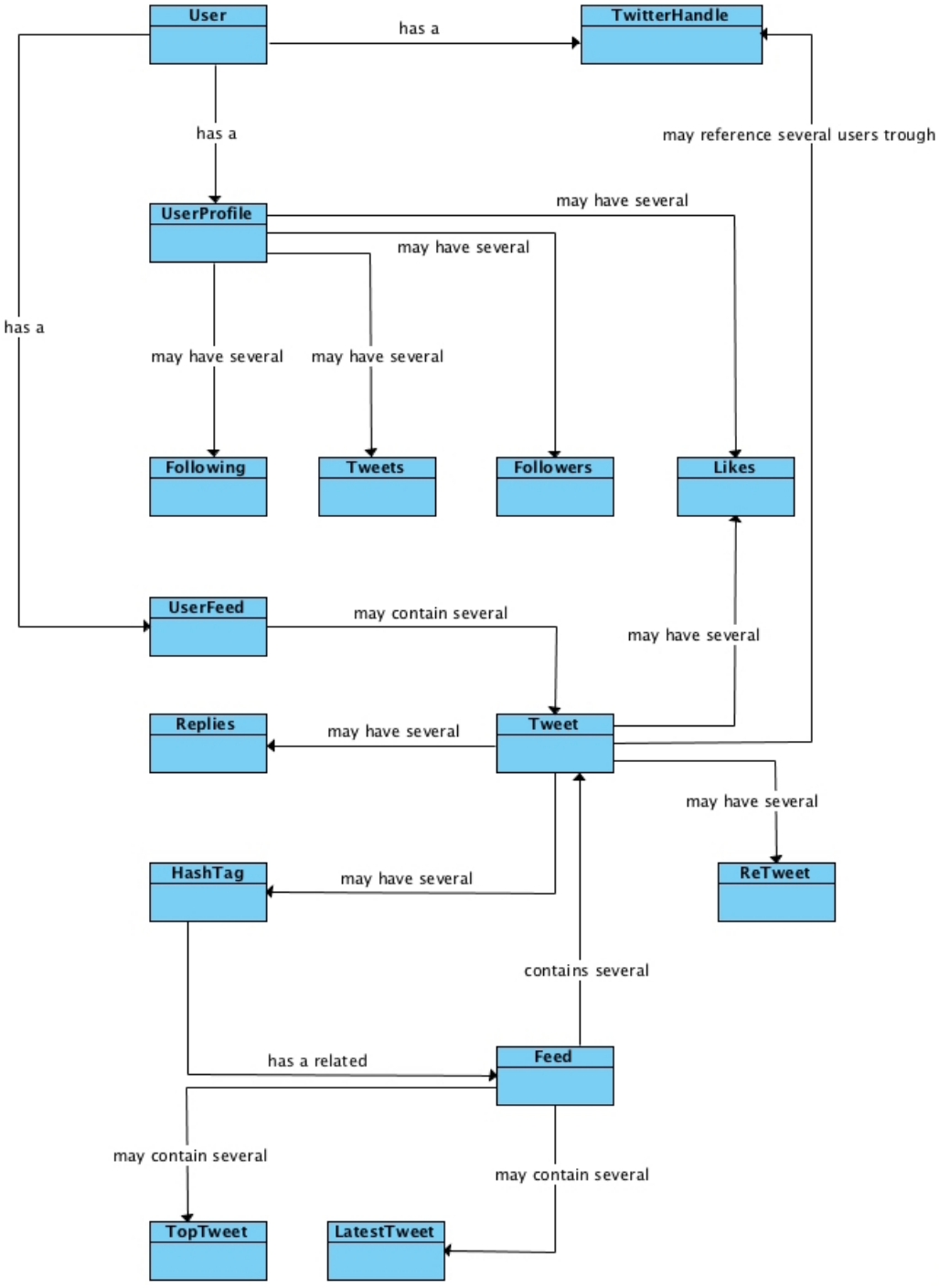


Figure 7: Twitter domain model schema.

Domain Model Analysis

In Figure 7 one may observe a very concise representation of the Twitter domain. Despite being a very minimalist OSN concerning to data properties and relationship complexity, Twitter has some very semantically strong features that brand the platform, being those features used among other well known OSNs such as Facebook. We are referring specifically the hashtag ((*HashTag* box) and the (*TwitterHandle* box)), but let us first introduce some of the more basic features.

As usual in this kind of platforms, Twitter's users have a user profile (*UserProfile* box) that has attached to it some properties and user metrics such as number of followers, number of tweets, number of other users the user is following, and number of likes the user has obtained across all his or her tweets (*Followers*, *Tweets*, *Following*, *Likes* boxes respectively).

The tweet is the basic unit of Twitter, is through tweets that information flows in Twitter. Also tweets may have several comments, and they may be *retweeted*. Now back to hashtags and twitter handles. Hashtags is in some way how these chaos of unstructured tweets gain some semantic value, in order to a group tweets according to a specific matter. Hashtags may in many cases be misleading, because users start to adopt hashtag to express sentiments or simply describing a tweet pushing back to the already mentioned unstructured chaos. Twitter handles are the same as tags in Facebook they serve as a mean to mark a specific twitter user, a twitter handle may be used in a comment or in a tweet leading directly to the respective user's twitter profile.

Twitter API

Similarly to other OSNs, twitter provides a REST API to fetch users data such as profile data, tweets or users' followers. Restrictions are felt again, this API is very limited providing a 15 minutes window for making HTTP requests.

Other data sources

As more and more of OSN appear to be closing doors to data availability even when for research purposes one may want to search for alternative data sources to feed the system that we intend to build along this project. Projects as KONECT (Kunegis, 2013) provide large data sets of networks collected online so that researchers may perform all types of operations and experiments on that data. This kind of alternatives are very valuable in terms of network analysis even for real time data analysis systems that may in a more immature phase of the project benefit from these data sets.

> *"KONECT (the Koblenz Network Collection) is a project to collect large network datasets of all types in order to perform research in network science and related fields, collected by the Institute of Web Science and Technologies at the University of Koblenz–Landau."* (Kunegis, 2013)

3.3.7 *Summary*

In this section we have explored with some detail six of the OSNs listed in the Table 1. In this analysis we followed a similar approach for analyzing each OSN, adding only an additional step for the more domain specific OSNs, that were ResearchGate and Pinterest, which was building a small data dictionary in order to ease the interpretation of the domain model schema.

From the analysis we may draw some generic conclusions concerning the domain of each OSN. Despite the differences and specifics of each platform, all them sum up to the basic primitive concepts of social networks, that are **actors** and **relational ties** between them, which form **subgroups** originating **groups** that build the network. This being the high level conclusion for our analysis, there are other patterns that emerge when analyzing different OSNs like the **user profile** that is a key element characteristic of these platforms and **feeds** (or **timelines**) that represent a standardized way of communicating events within a OSN.

3.4 How Online Social Networks Have Changed The World

Social media have clearly shifted the way we communicate and we perceived the world, simply putting it, nowadays with social media one can say that social media is responsible for *"everyone talking to everyone about everything all of the time"*.

In fact, 62% of the entire adult population in on social media (Duggan, 2015). As an example of events that were clearly influenced by social media, we have the presidential campaign of Barack Obama in the United States, started in 2007 and ended in 2009, Barack Obama had as his campaign technological adviser Chris Hughes, co-funder of Facebook, who played a crucial role in the campaign through online social media. The outcome of the election of 2009 could have been very different without the online social media.

Very interesting reflections are made on how social media impacts the world (Farida Vis, 2016), and the six major drawn conclusions are the following: **across industries, social media is going from a *"nice to have"* to an essential component of any business strategy**; **social media platforms may be the banks of the future**, as an example we have the bank customer profiling through social media in order to get a loan; **social media is shaking up healthcare and public health**, because information is spreaded *at the speed of light* through social media, this means less struggle to achieve public health and well-being awareness;

social media is changing how we govern and are governed, with OSNs public participation has grown and everyone can participate in their opinion making people voices louder, bringing more credibility to the democratically system implemented by many governments across the planet; **social media is helping us better respond to disasters**, as the health public awareness improved through social media information propagation speed, so did improved the response of governments and institutions to disasters such as natural disasters, in countries that may have not the services or infrastructure to respond to some catastrophes, making social media and crucial component to raise awareness across the globe, that have impact in help mobility, or fund raising for supports the damages made by certain disaster; **social media is helping us tackle some of the world's biggest challenges, from human rights violations to climate change**.

If we look particularly to the most globally used OSN there are in *"seven ways Facebook has changed the world"* (Elgot, 2015), we are going to point and comment out some of the most relevant. **Facebook has changed the definition of friend**, if back there having a dozen of friends was already a very large number of relationships, with Facebook the new limit was raised up to the hundreds or thousands of friends, the concept was given a completely new meaning, since we don't need to know a person face to face so that one becomes friend with the other, one simply needs to click the *"add friend"* button, and it does not matter if it is one's neighbor or some other person on the another side of the planet; **We care less about privacy**, *"if you are not paying for it, you are the product"*, means that we are not paying for using Facebook or any other OSNs, this said we must retain that these online platform profits from our information and from our interactions, but even being the majority of the users aware of this situation, that doesn't seems to bother anyone; **Facebook has created millions of jobs – but not in its own offices**, for example the marketing industry suffer a revolution since the raise of the social media, there are jobs for people to manage business and brands profiles on OSNs it's also a new way to approach customers, as we have seen previously with banks; **Facebook has been the tool to organize revolutions**, protests and awareness campaigns are raised inside facebook, this is related to the political influence and awareness capacity that we previously have pointed out in this same section.

Now switching to the negative aspects of not only Facebook but OSNs and social media in general. Very strong campaigns were raised against social media, for instance, *"The Anti-Social Network"* a short film depicting a life of an adult which as become obsessed with social networking at the point he starts to break boundaries between his real life and his virtual one. Strategically or ironically these campaigns use social media to spread the word.

We have seen that social media had a great deal of impact in society, what about our bodies? There are numeral studies on this matter, focusing on finding the true negative impacts of OSNs on our personal health. Scans to brains of people how excessively use social

media, point out that there is a clear degradation of white matter similar to people who are addicted to substances such as drugs or alcohol (Lin et al., 2012), in the regions that control emotional processing, attention and decision making, because social media immediate reward (instant feedback) with very small effort, this causes the brain rewire itself make us to desire these stimulations (Berridge and Robinson, 1998). Another common situation among OSNs users is the idea of multitasking, the felling that one is able to being productive in some task while browsing on social media. Users who heavily use social media are more susceptible to interference from irrelevant environmental stimuli (Ophir et al., 2009), leading these users to perform worse on a test of task-switching ability, because they were not able to filter out interferences.

4

SOCIAL NETWORK ANALYSIS

Social Network Analysis (SNA) is the study of how people are connected to each other, basically it studies a set of relations among a set of entities, these entities may be individuals, organizations, or even countries.

The common analysis procedure consists in mapping the network and then computing metrics to characterize the network. Then one tries to figure what is the structure of the network and why does it have that structure. SNAs is also about looking at the individuals inside the network and where are those individuals located.

4.1 GRAPH THEORY

Graphs are typically the base of representation of social structures. This mathematical approach maps with extreme convenience social networks. Nodes are individuals, and edges are relationships. Despite looking a quite simple approach, there is a very strong theoretical background that is of basilar importance for interpreting social networks. In the next sections we will explore how graph theory and network analysis coexist in order to provide more formal metrics for analyzing network structures and provide information about each node within the network.

4.2 NETWORK ANALYSIS OVERVIEW

In this section we intent to explore the scientific concepts behind network analysis, always trying to map them to reality, so only the core and applicable concepts will be explored in this section, namely:

- ***Power Laws*** - Power laws or power law distribution, represent in general a dependency relationship between two quantities. In SNs, the power law distribution describes a particular trend in the evolution of the number of relationships of individuals within a network;

- ***Centrality Measures*** - Centrality measures aim to answer the following question ***Which vertices are important?***. In a SN actor centrality measures the actor's interactions with other individuals;

- ***Link Analysis*** - Link analysis is a well known term form web search engines, popularized by the Page Rank algorithm. In SNs, link analysis measures individuals connections, such as identifying strongly connected nodes, absorbing nodes or even cycles inside networks;

- ***Community Detection*** - Community detection is related to clustering in social networks. Normally when analyzing SNs we aim for detecting communities (groups) that express similar ideas in matters such as politics, music or philosophy. Community detection is a far more abstract concept than geographical clustering, despite we often found it in OSNs such as Facebook, that the two concepts are tightly coupled;

- ***Spread of Information*** - Spread of information consists in a set of metrics that classify the propagation of the information within a network. Considering a Facebook post by a newspaper, it would come in hand to know, where was the starting point of that post, how many individuals it reaches, in which sub-networks the information was propagated, what were the entry points for that sub-networks, how fast the information got to the individuals, these are some of the concerns relating to spread of information;

- ***Social Learning*** - Social learning consists in the change of behavior or beliefs based on direct observation of other individuals. Considering again a Facebook post by some random individual A, and consider an individual B that shares (*'re-posts'*) the individual's A post. If one detects a pattern in this kind of interaction, one may say that individual B is learning from individual A (imitating, mirroring).

Some of the previous listed concepts represent metrics for analyzing networks, thus requiring a more detailed explanation. In the next sections we will focus one the most fundamental metrics that will be relevant for further reference in this document. [1]

4.3 RELEVANT METRICS FOR NETWORK ANALYSIS

These are crucial metrics that will be referenced within integral components of our system (that we will propose in the Chapter 5). We will use these metrics to add value to analysis features that we will provide to the end user. For that we must first address this concepts

1 At this point, and being network analysis basic concepts being covered it is normal that we interchangeably use the terms actor, node or vertices for denoting the same things

with a smaller granularity in terms of what they represent and also in terms of what can they offer us.

4.3.1 *Centrality*

Centrality if often mixed with node degree. Despite node degree being in fact used for centrality calculations, these metric have some variations that are worth to take a close look, in order to understand the different perspectives from where we can observe a particular node in a particular network.

Degree Centrality

The **degree** of a node is equal to the his number of adjacent nodes (or simply the number of first degree connections). So basically what do we get from this metric? When normalized the node degree value tells us the level of direct interaction of an actor with other actors within a network.

Closeness Centrality

Closeness centrality tells us how close an actor is to all the other actors in the network (not only with his first degree connections).

This metric is considered a sophisticated measure of centrality in network theory. It is defined as the mean geodesic distance (i.e., the shortest path) between a certain vertex v and all other vertices reachable from it. This concept is normally associated to geographic distances, being actors closeness mapped to reality. Still there are abstractions that compute this value not considering nodes as geodesic markers [2].

Betweenness Centrality

This measure reflects the number number of shortest paths going through a particular actor. **Nodes that occur in many shortest paths** between other nodes in the network have a higher betweenness centrality, basically takes into account the connectivity of the nodes' neighbors, giving a **higher value for nodes which bridge clusters** [2].

Eigenvector Centrality

This measure is based on the following statement:

> ***"Importance of a node depends on the importance of its neighbors."***

2 Social Network Analysis - Theory and Applications: https://www.politaktiv.org/documents/10157/29141/SocNet_TheoryApp.pdf

Eigenvector centrality [2] measures importance of a node within a network. This measure assigns relative scores to all nodes, then if a node is connected to a *high scored* node it has a bigger increment to its score then when connected to a *low scored* node. One of the most famous variants of eigenvector centrality is the Google's PageRank algorithm (Brin and Page, 1998).

Page Rank

PageRank algorithm (Brin and Page, 1998) was thought as a way to rank online content in order to discover what sites are important and really worth to consult. A simple justification is that a page may have a high PageRank when there are many other pages that point to it, or if some pages point to it and simultaneously have a high PageRank.

4.3.2 *Clustering and Community Detection*

Represents the value of tendency for certain nodes to form a cluster. Normally actors within a network tend to aggregate when having some simple characteristic in common such as living in the same city, working in the same place or event frequenting the same gymnasium.

A common approach for detecting communities is through graph clicks (subset of vertices of an undirected graph where all vertices are connected between each other), being the normalized clustering coefficient a high value when the network consists in a set of disjoint clicks.

4.3.3 *Node Dominance*

Dominance may be related with betweeness centrality but it focus particularly on node reachability. One may say that a node $v1$ dominates a node $v2$ if $v2$ needs to go through $v1$ to reach a certain node $v3$.

4.4 SMALL WORLD PROBLEM

This principle of *small-world phenomenon* is based on the idea that all human beings are connected by **short chains of acquaintances**. The pioneers of this work were Stanley Milgram and Jeffrey Travers (Travers and Milgram, 1967).

Six Degrees of Separation

The concept of six degrees of separation is an extension of the small world problem. In the sequence of what we stated before, the six degrees of separation materialize the previous concept in six interconnections for some individual to reach any other one. Six degrees of separation gained a particularly strong relevance, when a play was written in the 90's portraying the concept.

4.5 NETWORK VISUALIZATION

Network visualization may be considered as a science by itself. In the context of this project we will not look further into network visualization, we will instead in further Chapters (more technical Chapters) reference advanced visualization technologies that will help us on the tool implementation serving as a fundamental complement to social network analysis.

4.6 SOCIAL NETWORK ANALYSIS SOFTWARE

> *"(...) more sophisticated graphics capabilities should make exploratory studies using visual displays of networks more fruitful. One should be able to display actor attributes and nodal or subgroup properties (such as expansiveness, centrality, or clique membership) along with the graph. (...)"* (Wasserman and Faust, 1994)

4.6.1 *Software Tools*

Next we present some relevant software tools on SNAs.

4.6.2 *Structure*

The program Structure (Pritchard Lab, 2000) is a free software package mainly used to investigate population structure. Its uses include inferring the presence of distinct populations, assigning individuals to populations, studying hybrid zones, identifying migrants and admixed individuals, and estimating population frequencies in situations where many individuals are migrants or admixed.

4.6.3 *Gephi*

Gephi (Bastian et al., 2009) is a tool for keen data analysts and scientists who want to explore and understand graphs. Like Photoshop™ but for graph data, the user interacts with the representation, manipulates the structures, shapes and colors to reveal hidden patterns. The goal is to help data analysts to make hypothesis, intuitively discover patterns, isolate structure singularities or faults during data sourcing. It is a complementary tool to traditional statistics, as visual thinking with interactive interfaces is now recognized to facilitate reasoning. This is a software for Exploratory Data Analysis, a paradigm that appeared in the Visual Analytics field of research.

4.6.4 *UCINET*

UCINET 6 (Lin Freeman, 2002) for Windows is a software package for the analysis of social network data. It was developed by Lin Freeman, Martin Everett and Steve Borgatti. It comes with the **NetDraw** (Borgatti, 2002) network visualization tool.

4.6.5 *SocNetV*

Social Network Visualizer (Kalamaras, 2004) is a cross-platform, user-friendly application for the analysis and visualization of Social Networks in the form of mathematical graphs, where vertices depict actors/agents and edges represent their relations.

With SocNetV you can construct social networks with a few clicks on a virtual canvas or load field data from various social network file formats such as GraphML, GraphViz, Adjacency, Pajek, UCINET, etc.

Furthermore, you can create random networks using various random models.

4.6.6 *networkx*

networkx (Hagberg et al., 2013) is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. networkx relevant features are listed below:

- Python language data structures for graphs, digraphs, and multigraphs;
- Many standard graph algorithms;
- Network structure and analysis measures;
- Generators for classic graphs, random graphs, and synthetic networks;

- Nodes can be "anything" (e.g. text, images, XML records);
- Edges can hold arbitrary data (e.g. weights, time-series).

4.6.7 *Vizster*

Vizster (Heer and Boyd, 2005) is a tool for Visualizing online social networks, a visualization system for playful end-user exploration and navigation of large-scale online social networks.

4.6.8 *Project Palantir (Facebook)*

Project Palantir (Facebook, 2008) [3] is an impressive tool that displays the rate of interactions on Facebook across the globe. This was a result of an intern annual event that happens in Facebook where all the employees are invited to participate and to build creative prototypes in the context of the company.

3 This project is not at the level of SNAs specificity of the previous tools, still we consider worth mention it.

5

SYSTEM ARCHITECTURE PROPOSAL

Before diving into the architectural details of the system, we first presented a state of the art summary, that concisely describes what is the positioning of this project in the perspective of the previous explored SNAs tools that we presented in Chapter 4, and also considering the OSNs that we studied in Chapter 3.

Specifically regarding the SNAs tools, we will comment some of them, some of their useful features and overall comments to what may lack on these tools that this project may target, in order to differentiate and not only *"reinvent the wheel"*.

5.1 SIMPLICITY

Aside of **Vizster** (Heer and Boyd, 2005), the majority of the previously presented tools such as Gephi (Bastian et al., 2009) or Social Network Visualizer (Kalamaras, 2004), are very complex tools with very heavy interfaces, that have a big learning curve and are meant for users that have particular advanced knowledge in SNs and SNAs. The tool to be developed could also serve for less expert users, providing a set of core basic functionalities (e.g only allow users to load and visualize their networks), and then, allow the user to build complexity from there enabling and disabling other features.

5.2 ACCESSIBILITY

All the software that we presented above exists in the form of desktop applications. These applications need to be downloaded, and installed in a compatible machines (sometimes with dependencies on other software that is not yet installed). Nowadays almost every application is web-based, this allows users to access them every where through a browser, making web apps a solution that is Operating System and device agnostic. This said, building a web-based social networks analysis tool could be a way of tackle the accessibility of such tools.

A web-based application, is good for sake of accessibility but in another hand it is a

culprit when it comes to performance. This is a decision to take into account, but always having in mind that tackling performance it's not the main goal this master's thesis, also, the mentioned tools are mature projects that are highly performant and are capable of rendering huge networks.

5.3 Online Social Network (OSN) integration

Social Network Visualizer (Kalamaras, 2004), allows to *scrap* web sites to build networks, but for this feature relies only on links to build the network (it blindly scraps recursively some url to build the network). By allowing the user to analyze networks that are directly reporting their social network status would be a differentiation factor from the other tools, and would certainly be a more meaningful and valuable analysis for the end user.

5.4 drawing accurate conclusions

As we stated before when talking about simplicity, the mentioned SNAs tools provide generic metrics on networks such as network density or actor centrality. The values outputted from these tools are the result of running generic formulas and algorithms against some networks, so its very common for current SNAs researchers to be worried about the size of the network, being their focus on **quantitive analysis**.

In a hypothetical analysis scenario where some researcher has a network with a few thousand nodes, **what is the meaning of his assumptions when analyzing the network**? Since this is a pure quantitive analysis the numbers will seem reasonable for the given network, but this will not allow him to extract contextual conclusions, because in this case analyzing data from Facebook or analyzing data from LinkedIn will sound just like the same, it would all come down to the network. A better approach for drawing conclusions would be to have a mixture between **quantitive analysis** and **qualitative analysis**, the tool could do some content and context analysis to help the end user on achieving more meaningful conclusions, rather than just some numerical metrics.

5.5 system positioning and tools comparison

In this section we will make a high level comparison between the software tools presented in Chapter 4. In Table 2 we can observe the tools classification based on some predefined metrics that show the positioning of the proposed system, these metrics are:

- **Availability (Desktop or Web)** - Whether the tool available through a desktop application or a web application;

- **Complexity (Low, Moderate, High)** - Whether the tool has very complex features that require expertise to be used, also we may consider the learning curve for using the tool with efficiency;
- **Performance (Low, Medium , High)** - Whether the tool is performant, if it computes metrics with velocity and if it renders dense graphs without struggle;
- **Network Edition (Yes, No)** - Whether the tool allows network editing, such feature allows adding nodes and edges to existing network or event creating new networks from scratch;
- **OSNs Integration (Yes, No)** - Whether the tool is able to integrate data analysis of OSNs.
- **Contextual Analysis (Yes, No)** - By contextual analysis we do not mean that the user will not be aware of the network context, our contextual analysis has a strong meaning, it represents the capacity that the system demonstrates (or not) to be aware of the context of the network and providing metrics with a specific meaning.

Tool	Availability	Complexity	Performance	Network Edition	OSNs Integration	Contextual Analysis
Structure	Desktop	Moderate	Medium	Yes	No	No
Gephi	Desktop	Moderate	Good	Yes	No	No
UCINET	Desktop	High	Very Good	Yes	No	No
SocNetV	Desktop	Low	Medium	Yes	No[1]	No
Our system projections	Web	Low	Low	No	Yes	Yes

Table 2: Software tools comparison and our system positioning.

As we can observe in Table 2 our system has essentially three differential factors, that are: web availability; OSNs integration; contextual analysis; being the trade off for such gains the system performance. When describing our system compared to the other tools we want to be able to have a web tool that has a complexity level similar to SocNetV[1].

5.6 System Architecture

Now, after building up our aiming for this project, we now present a more concrete image of the overall system. In Figure 8 we present an abstract system architecture.

1 SocNetV has a feature that allows the user to launch a *web spider* that navigates through web sites building a network representative of the links between the sites, but the *spider* is not content aware, it blindly builds a network without context.

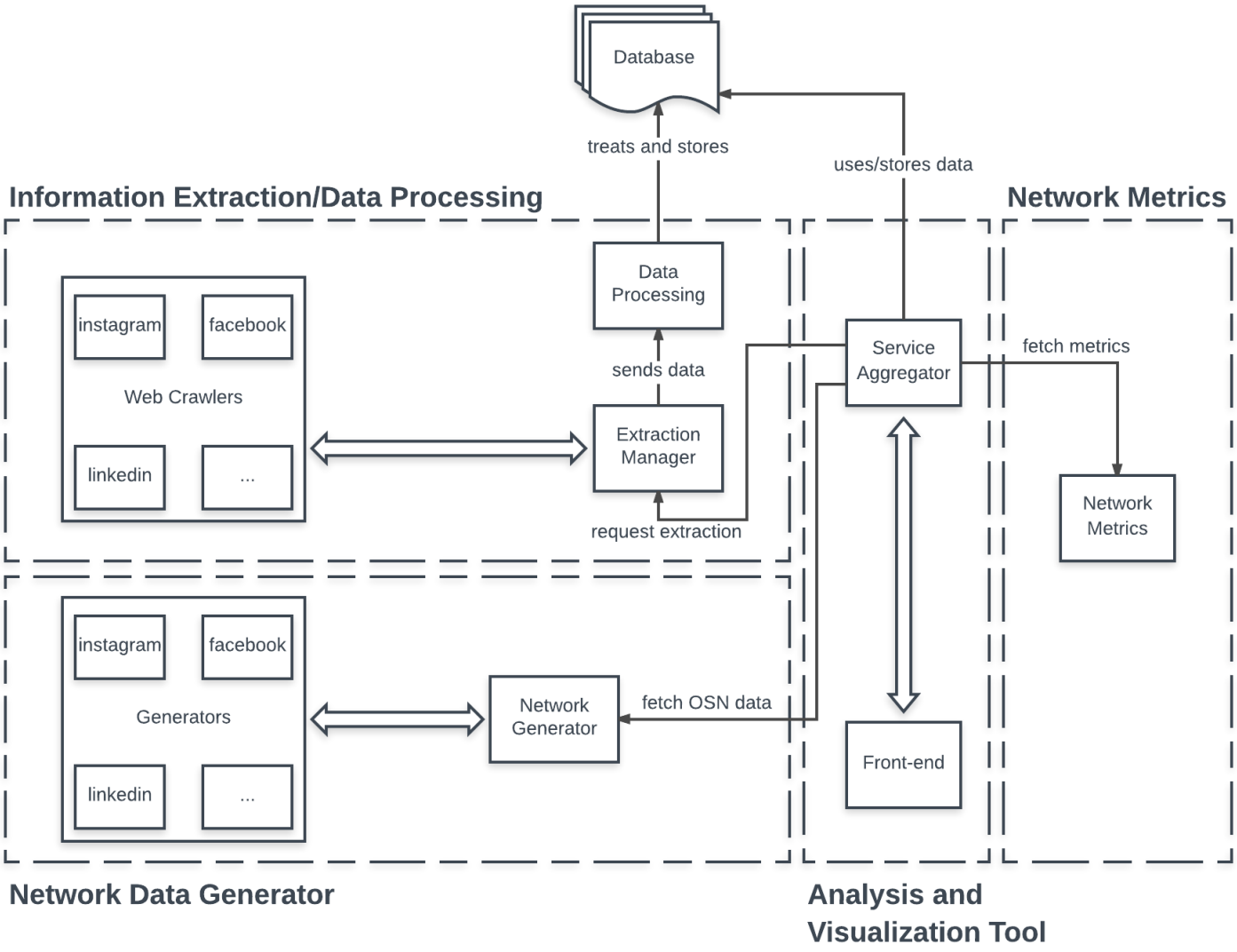


Figure 8: System architecture proposal.

5.6.1 *General overview*

As the interaction of the software components may be clear from the diagram, the role of each module is not clear by simple diagram observation, an underlying explanation of each component is needed in order to understand the system.

We will follow a *top down* approach for explaining the system architecture. First let us be clear about the two main and distinct parts of the system:

- ***Information extraction and data processing*** - All the components are built for extracting information from existent databases, or from OSNs (through the **Web Crawler**) and store information being information properly treated before stored;

- ***Network Data Generator*** - For sake of a ease of development process, and also to assure a fallback strategy upon information extraction failure, we create a network generator module that basically generates data models confined to the data schemas that we previously presented for Facebook and LinkedIn;

- ***Network metrics*** - This module acts as a isolated component that is dedicated to perform calculations and algorithms on stored networks. It will feed metrics as requests by other components;

- ***Analysis and Application/Visualization*** - The tool that directly interacts with the end user is composed by a **Service Aggregator** that fetches data from a database, requests extractions to the application back-end and runs calculations and algorithms on top of stored networks as the user requests by interacting with a **Front-end** that provides the visualization and interaction features.

5.6.2 *Detailed Components Description*

The components presented in Figure 8 more detailed explanation, next we look more carefully into each on of the components.

- ***Online Social Network (OSN)*** - This are the object of study, the source of information that the systems will process and analyze;

- ***Web Crawler*** - The **Web Crawler** consists in a set of modules for crawling each one of the OSNs (*fb-extraction* and other modules);

- ***Extraction Manager*** - This module consists in a wrapper for extracting information from social networks, and allows extraction orchestration spreading extraction processes along multiple hosts, so that we can mitigate the slowness of web crawlers and extraction process in general;

- ***Data Processing*** - The data processing assures that we store a well defined data schema that describes in the more simplified way the state of the networks;

- ***Database*** - The database is where we store our data. It is not represented by the *classical cilindro* because it resembles relational databases, and the possibility of using non relational databases such as document databases, grows strongly within the project, and the reason is the unstructured data that we will be storing into our database. We also plan on feeding some data through already existing databases, instead of crawling data from OSN. This databases may be provided from projects that we already mentioned in this document (Section 3.3.6), such as (Kunegis, 2013). This data

would be accessed through the **Data Miner**, or a new module could be constructed exclusively to feed this data to our database;

- ***Generator*** - a generator is a simple module that creates contextualized sets of data in order to fed our front end with the expected data that would come from the extraction module;
- ***Network metrics*** - These module fetches data directly from the database in order to perform network operations that may be heavy. Isolating this component will allow logic separation from the service aggregator and will allow a separated infrastructure deploy, so that we may have dedicated computer resources on network metrics calculations;
- ***Service Aggregator*** - Ideally this component application will read the already normalized information from the database, run SNAs calculations and algorithms against the stored networks, and request data to the back-end (the Information extraction and data processing component). The Service aggregator is also responsible for communicating with network measures component in order to fetch metrics about a given network as the user requests to access it;
- ***Front-end*** - The front-end will render the networks to the user, and will allow the user to interact with the network; these interactions will be defined in the requirements specifications.

6

SYSTEM REQUIREMENTS

In this Chapter we will specify in detail the system requirements and particular features to be implemented. The requirements will be divided in two major sections.

First we will describe what tasks the Back-end of the system should perform in order to provide all the data and tools for supporting the system Front-end. Then we will define the tool requirements from the user point of view. For the aggregator no requirements will be specified since this component will only bridge requests from the Front-end and the Back-end or will eventually fetch data directly from the database.

6.1 SOCIAL NETWORKS PRIORITIZATION

Before diving into the requirements we first will review our OSNs preferences regarding information extraction and the interest we have in analyzing these specific networks.

First we want to analyze **Facebook** because it is the most general purpose network, the most popular and the most used thus allowing us to derive more interesting conclusions since the resultant graphs will be more realistic having a more concrete social structure representation. Second we want to analyze **LinkedIn** because it is also widely used and the only that specifically focus on professional worldwide networking, generating different kinds of graphs and understand how companies and professionals are interacting online. Analyzing LinkedIn may also introduce an interesting analysis that is merging information from Facebook and analyzing friendship networks within professional networks.

Having two networks embedded in the system proves that we can analyze social networks in general since we have more than one and with different purposes, but since the system is designed to simply accommodate new networks simply adding a new extraction module should the major part of the work to integrate a new OSNs, this said we could eventually also implement some extra modules to the remaining OSNs listed in Chapter 3.

6.2 BACK-END

As seen is Figure 8, our Back-end is essentially composed by two parts: **web crawlers/extraction modules**; **extraction manager** and the **data miner**. We will write the requirements for each one of the components. We will not prioritize these requirements (as we will do in the next section for the Front-end requirements) because **all listed requirements are essential for the overall system usefulness**.

6.2.1 *Web crawlers*

Each web crawler (or extraction module) must fulfill common requirements that are listed below [1]:

1. Web crawlers should be able to login with a user account (an *entry point*);
2. Web crawlers should be able to navigate through the pages of a given OSN;
3. Web crawlers must be capable of performing "human" interactions such as click and scroll;
4. Web crawlers should be able to output a predefined (agreed and formally defined in the next section) data schema, covering eventual exceptions due to privacy limitations;
5. Web crawlers must be able to perform user extraction with second order depth, from the user entry point perspective (this means that we want to extract user's friends and friends of friends information);
6. Extraction modules should provide a global extraction method where extraction parameters can be passed from the outside reducing or amplifying the scope of extraction as specified (e.g. under given circumstances we may only need to extract the friends' list or the basic information like name, city and birth date);
7. Extraction modules must be available to the data miner through a web API in order to allow remote and distributed extraction. The web API must wrap all the different supported OSNs being each one accessible through a different path within the same web API. The extraction web API required specifications are presented next:
   - **GET /api/v1/extraction/{osn}** - should return a confirmation message signalizing that API is up and ready for receiving requests;
   - **GET /api/v1/extraction/{osn}/{user_id}** - should perform full extraction of the user with the *user_id* in the *osn* ;

1 These requirements are agnostic to the OSNs context

- **POST /api/v1/extraction/{osn}/{user_id}** - should receive a set of options, that parameterize the extraction and reduce the scope of the extraction for a given *user_id* within some *osn*.
- **POST /api/v1/extraction/{osn}/** - same as the previous but instead of performing extraction for a given *user_id*, performs it to a set of *user_ids* performing multiple extractions;
- In API version 1 **osn** must be one of the following: **facebook**, **linkedin**;
- **user_id** is a string that uniquely represents the user within a specific OSN.

6.2.2 *Extraction Manager*

Below are the extraction manager requirements [2]:

1. Orchestration of extraction processes scattered across various hosts: one should be able to define a list of hosts and the number of extraction processes that each host should handle;
2. Chunk an entry point (that is a set of user identifiers within the OSNs) in order to delegate different users to different hosts;
3. Call the extraction endpoints according to the OSNs from where we need to extract data.

Extraction pipeline

Being listed above the requirements for each component we will now draw the specification of what is the expected workflow for data extraction, in Figure 9 we design a pipeline that tries to reflect, with maximum detail, the listed requirements. The diagram does not cover the data processing that is responsible for normalizing data and store it. This diagram is exclusively focused on how we pretend that data extraction is achieved in order to mitigate the slowness of web crawlers.

As we can see from Figure 9 we aim to follow a very straight forward process in order to extract information. First we provide an entry point for a given OSNs (the user the web crawlers will use to log in into the social platform), and a hosts file that describes the resources available for extractions, this is intended to be simply a list of hosts (IP addresses) that have the extraction web API running and awaiting for extraction requests.

Next each extraction API instance is responsible for handling a session of some web crawler instance and waits for it to return data so the extraction API instance can give it back to the extraction manager.

2 Again, these requirements are agnostic to the OSNs context

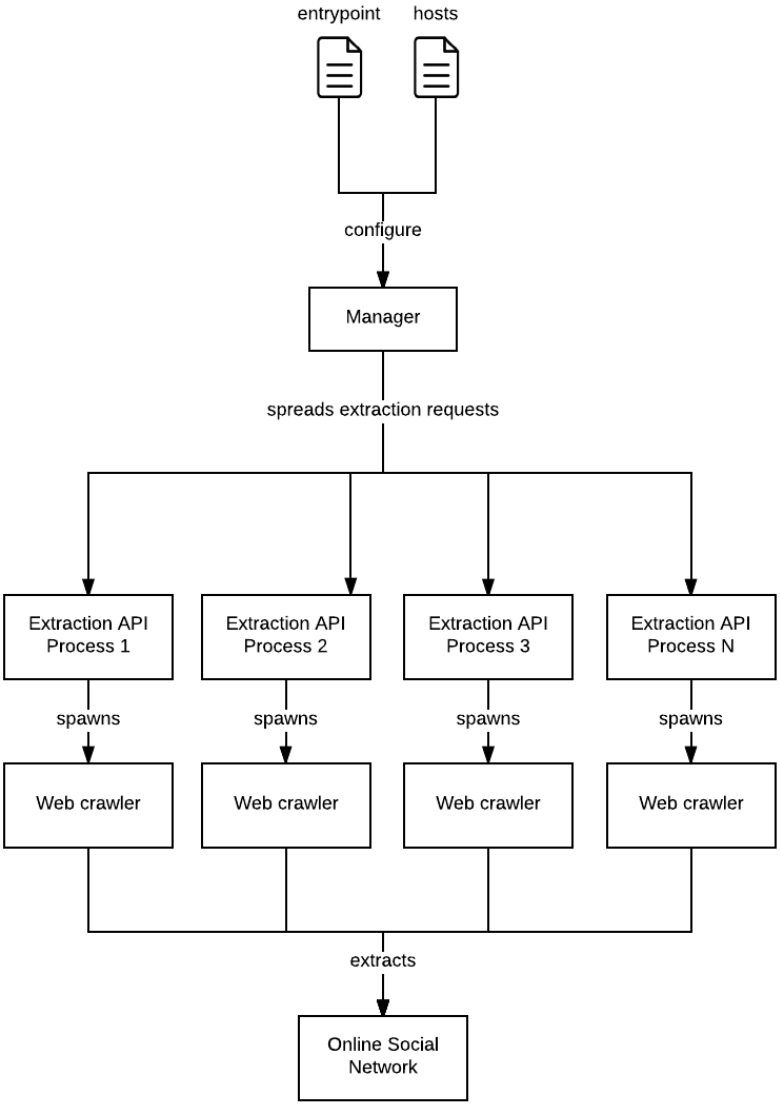


Figure 9: Extraction pipeline diagram.

6.2.3 *Data miner*

The data miner simply assures some data treatment before storing it on the database, that said there is a very narrow requirements list for this component:

1. Receive extraction data and normalize the fields that may need some treatment giving as result a normalized data structure;
2. Store normalized data in the database;
3. Assure that the data schemas (these are presented in the next section) are well defined.

Data schemas

Defining data schemas in earlier stages of system specifications will allow us to develop the Front-end and the Back-end simultaneously, we must for that consider that the only source of true when it comes to data structures is a well agreed contract between both parts. The data miner will assure that the next presented schemas are stored in the database. For convenience reasons we will describe the data structures with a JSON like notation.

Facebook data structure

```
{
    "uid": {string},
    "livesIn": {
        "id": {string},
        "description": {string}
    },
    "life_events": {
        {string}: [{string}]
    },
    "birthDate": {string},
    "likes": {
        {string}: {string}
    },
    "friends": [{string}],
    "relationships": {
        "civil_status": {
            "id": {string},
            "description": {string}
        },
```

```
        "family_members": [
            { "id": {string}, "relationship": {string} }
        ]
    },
    "from": {
        "id": {string},
        "description": {string}
    },
    "name": {string},,
    "gender": {string},
    "age": {number},
    "posts": [
        {
            "timestamp": {string},
            "description": {string},
            "author": {string},
            "reactions": {
                "likes": {number},
                "love": {number},
                "laugh": {number},
                "sad": {number},
                "angry": {number},
                "surprise": {number}
            }
        }
    ]
}
```

LinkedIn data structure

```
{
    "uid": {string},
    "name": {string},
    "headline": {string},
    "from": {string},
    "summary": {string},
    "experience": [
        {
            "company": {string},
```

```
        "position": {string},
        "duration": {
            "count": {number},
            "unit": {string},
            "from": {string},
            "to": {string}"
        }
    }
],
"education": [
    {
        "institution": {string},
        "course": {string},
        "startYear": {number},
        "endYear": {number}
    }
],
"skills": {
    {string}: {number}
},
"languages": {
    {string}: {string}
},
"projects": [
    {
        "name": {string},
        "date": {string},
        "description": {string}
    }
],
"groups": [
    {string}
],
"following": [
    {string}
],
"connections": [
    {string}
```

```
    ]
}
```

6.2.4 *Network metrics*

In this section we will list the requirements for the module that is responsible for calculating metrics upon our stored networks. This component must provide a web API in order to access all the algorithms and metrics calculations that the service offers.

1. The API must be able to calculate strongly and weakly connected components;
2. The API must be able to calculate the clustering coefficient for a given network;
3. The API must be able to calculate the average neighbor degree;
4. The API must be able to calculate centrality measures, including:
    a) Degree centrality;
    b) Closeness centrality;
    c) Betweenness centrality;
    d) Eigenvector centrality;
5. The API must be able to compute node importance through the page rank algorithm.

6.3 FRONT-END

The Front-end is actually where the majority of the requirements work is, since we need to go into detail of how the user will interact with the tool, we must decide how those interactions will be drawn so that the tool can actually be what it was meant to, also bear in mind that these represent the tool requirements, what the user actually will able to see.

6.3.1 *Requirements Prioritization*

For simplifying the prioritization process we will use the **MoSCoW** method (Clegg and Barker, 1994) that is a simple method to define what requirements are more important for the system overall functionality, allowing us to focus on the very essential requirements for getting a functional product [3].

Next we present the MoSCoW method as it is defined in requirements engineering.

3 In this section we will tend to use some terms often find in requirements engineering that may seem a bit off topic, still we find that this is the more objective way for describing our prioritization method

- ***M**ust* have requirements are critical requirements that are part of the identity of the product, they must by all means be implemented.
- ***S**hould* have requirements are definitely important, but they are not critic to the product definition, and they are not time critic as well having the possibility of being included in later stages of the implementation. Some times these requirements may have another ways of satisfying the customer.
- ***C**ould* have requirements are indeed the *nice to have* requirements, being often left outside of the first deliver, but seen as very valuable to the future of the product in later stages of the product time line.
- ***W**on't* have requirements that are agreed to not be included in the first deliver of a project, this does not exclude the possibility of including them in later stages of the project. *Won't* requirements may be seen as future work.

The requirements will be listed by groups (sections) that aggregate common requirements or features. Each requirement will have a classification according the MoSCoW method.

6.3.2 *Network configuration and construction*

In this group of requirements we present a set of requirements that represent the operations that allow the users to get their network built.

1. **[MUST]** The user must be able to generate a network for available OSNs;
2. **[MUST]** The user must be able to choose the number of nodes for a certain network and which metrics he wants to compute for that network;
3. **[MUST]** The user must be able to register available OSNs accounts in the system;
4. **[MUST]** The user must be able to order the build of its network for a given OSN;
5. **[MUST]** The system must give feedback on the extraction status;
6. **[COULD]** The user must be able to *blacklist* from the network nodes with a minimum or maximum number of connections;
7. **[COULD]** The system must clearly warn the user about the impacts that extracting some kind of data (e.g. extracting complete list of user's likes on Facebook) could have on extraction time and consequently on render network time (these could be expressed via label warnings in the user's interface);

8. **[COULD]** The user must be able to *blacklist* nodes specific from being extracted and consequently rendered on the user's graph;
9. **[WON'T]** After the first extraction all the extracted nodes must be marked as extracted, being the user able to extract the missing properties for some given nodes.

6.3.3 *General interactions and display*

These requirements express general behavior of the tool, and some display features.

1. **[MUST]** The system must be able to render a graph using the information provided by the aggregation service;
2. **[MUST]** The system should be able to automatically identify communities by painting nodes belonging to the same community by the same color and providing information about the community such as *"People that studied at School/University X"* or *"People that live in city Y"*;
3. **[MUST]** The user must be able to drag and drop the graph to any place on the graph render area;
4. **[MUST]** The user must be able to zoom in and zoom out the network so he is able to explore specific parts with more detail;
5. **[MUST]** The user must be able to select two nodes at the same time in order to compare them all values should be displayed side by side in order to provide a practical way to compare two individuals at any level;
6. **[SHOULD]** The user should be able to choose activate animations despite these have been deactivated by the system for sake of graph interactions performance ;
7. **[SHOULD]** The system should be able to automatically deactivate heavy graph animations if a large graph is being rendered;
8. **[COULD]** The user should be able to enable and disable *fisheye distortion* alike effect;
9. **[WON'T]** The user must be able to perform a *hive plot* [4] of his network;
10. **[WON'T]** Double clicking on a empty zone should perform a smooth zooming effect on that area.

4 hive plots define a linear layout for nodes, grouping nodes by type and arranging them along radial axes based on some property of data

6.3.4 *Node interactions*

Here we describe interactions at the node level.

1. **[MUST]** Along side the node a label with the node name or id should be displayed;
2. **[MUST]** The user must be able to activate highlight functionality for more interactive node consulting. This functionality will highlight the node and his first degree connections, clarifying relations within very dense clusters;
3. **[MUST]** When the user clicks a node a side panel must be opened, this panel should display the following:
   a) Should contain all node user's available information;
   b) Should allow the user to perform calculations on that specific node;
   c) Should allow user to request extraction of more information on that node (e.g. if the list of user's likes wasn't extracted this option should be available);
   d) Should offer the user all the metrics already mentioned the previous network metrics section 6.2.4.
4. **[MUST]** The user must be able to drag and drop the node to some place else in the screen and the node should be fixed in that place (being the rest of the graph automatically rearranged);
5. **[COULD]** The user can pick color and size of his nodes within the network;
6. **[COULD]** When the user mouseover a specific node relevant information should be displayed when possible, such as: name, age, address, number of connections;
7. **[COULD]** The user can pick color and size of specific pre selected nodes within the network;
8. **[WON'T]** Right clicking on some node should open a context menu that provides options to the user such as:
   - Opening the users' profile in the current OSNs;
   - Change the node symbol (e.g. if it is a circle the user might want to make the node a triangle instead).
9. **[WON'T]** Double click on some node should make the node grow and stand out comparing remaining nodes.

6.3.5 *Link interaction*

Links are not only visual node connectors, these also possess characteristics and metrics that can be consulted.

1. **[MUST]** User may choose to render the graph links with semantic thickness, if the user activates this option, the link thickness should be proportional to the number of common connections between two given nodes, indicating strongly connected individuals;
2. **[COULD]** When the user performs a mouseover on some link, the link itself should be highlighted as well as the intervenient nodes;
3. **[COULD]** When the user performs a mouseover on some link, relevant information about the link should be displayed such as number of interactions between the two nodes, or number of common connections.

6.3.6 *Bulk operations*

The user may select a set of nodes with a selection box, allowing him to perform bulk operations on nodes, such as:

1. **[COULD]** The user must be able to collapse dense clusters in one single node (all nodes would be replaced by a bigger node, not necessarily representing a community);
2. **[COULD]** The user must be able to group nodes in communities based on specific OSNs property (e.g. such as page likes on Facebook or skills on LinkedIn);
3. **[WON'T]** Check what are the connections that the selected nodes have in common;
4. **[WON'T]** All the metrics that can be consulted in node interaction must also be available in bulk interactions so that the user may compare metrics among a set of nodes;
5. **[WON'T]** The user must be able to paint all selected nodes with the same color;
6. **[WON'T]** Check what are the preferences (in Facebook it would be the *likes*, in LinkedIn would be the companies they follow) that the selected nodes have in common.

6.3.7 *Statistic analysis*

The system should also provide some statistics on the user's network.

1. **[SHOULD]** The user must be able to visualize geographical network distribution;
2. **[SHOULD]** The user must be able to rank nodes by various metrics such as node centrality;
3. **[SHOULD]** The user must be able consult node rankings (what are the most popular or active users) in the context of a given OSN (e.g. on Facebook we may have a rank by number of reactions to user's posts while in LinkedIn we can have a rank of must recommended user's on particular skills).

6.3.8 *Other operations*

These are other operations that differentiate from the other groups of requirements and do not fit any particular requirements bucket.

1. **[MUST]** The user must be able to download his network in the standard graph format **GraphML** (Brandes et al., 2001) so that it could be imported to other SNAs tools such as Gephi (see Chapter 4 section 4.6);
2. **[WON'T]** The user should be able to enter an edition mode where he appends new nodes to the social structure.

6.3.9 *Specific OSNs requirements*

As we previously mentioned in Chapter 5, one of the main value propositions of building an OSN analysis and visualization tool is to offer contextual analysis, specific inferences driven by system awareness regarding the OSNs that we are analyzing.

Facebook specific requirements

Below we list requirements that are Facebook specific:

1. **[COULD] Sentiment analysis** - The user must be able to see a metric on each node that describes sentiments such as happiness or sadness, this will be simply the result of the mapping and extraction of reactions to user's posts giving us an overall idea of the user sentiments without involving any natural language processing or other complex processes. Our approach should consist in the analysis of the Facebook posts reactions (presented in section 6.2.3):

```
...
"reactions": {
    "likes": {number},
    "love": {number},
    "laugh": {number},
    "sad": {number},
    "angry": {number},
    "surprise": {number}
}
...
```

2. **[COULD] User activity** - By analyzing timestamps on user's posts we will provide a metric that describes user activity;

3. **[WON'T] Link Analysis for user social interaction** - When clicking on links in the graph the user must be able to tell the degree of interaction between two nodes (this interaction metric should derive from the number of mentions or posts in user's posts).

LinkedIn specific requirements

Below we list requirements that are LinkedIn specific:

1. **[COULD] Human resources discovery** - As companies struggle to find people with particular skills, it might in some cases be a matter of how to reach certain nodes in the network. The user must be able to find individuals with particular skills on the network but also the *shortest path* to that individual, as well as the point of contact a third individual that is a first degree connection with the target and that could serve as proxy to reach that person;

2. **[WON'T] Carrear history** - It could be useful to see a particular career path for a specific user (we could call it a user career diagram);

3. **[WON'T] Carrear development** - Because nowadays people tend to change jobs more frequently, the user could be able to tell from the network general behavior, that users from a certain company tend next to go to some particular companies.

7

SYSTEM IMPLEMENTATION

In this Chapter we will get a closer look to the steps we took towards Socii implementation. First we will present a small proof of concept as mean for validation of our architectural main workflow (get the network rendered with OSN data), and at the same time experimenting with some technologies that we think that best suite our needs, we will then present our technological choices based on this first proof of concept.

In the next parts of the Chapter, we will detail more on each component of our system from the extraction component to the front-end of Socii. We will also present at the end of the Chapter some of the main workflows within Socii and how all the components interact between them in order to produce a certain outcome.

7.1 IMPLEMENTATION FIRST STEPS

In this section we describe our approach towards the implementation of the system, we will describe the process since the requirements definition to the technological choices, some challenges and implementation details.

For gathering requirements we simply defined two groups, the first, the system Back-end has essential base functionalities, we focused only on the essential without scoping or prioritizing, all the collected requirements are in the progress of being implemented, these include web crawling modules, data processing for some data treatment and an extraction manager that allows remote calls of parameterized (granular) extractions. In the system Front-end we followed a different approach by collecting a larger group of requirements that consist mainly in user interactions with the tool, allowing us to narrow down the essential features based on requirements comparison. So at the end we sum up a few *must have* requirements that define the system identity an reflect the principles on which the project was designed upon (accessibility, simplicity, OSNs integration and contextual analysis).

From here we built a simple *proof of concept* that demonstrates the most basic of the workflow, this consists in a few steps that we next list:

- **Back-end** - Extract users from a OSN (for this particular case we used Facebook as source);
- **Service Aggregator** - Aggregate the extracted users in a graph respecting front end data contract;
- **Front-end** - Rendering a graph on the browser, allow simple interaction of node data display on the user mouse click.

Aside note

As one may noticed in the previous list, for sake of objectivity we skipped the implementation of some pieces in the architecture, namely, the network metrics API and the data processing. In fact, these will only be included in the full implementation, because for the current proof of concept we labeled this components as complements (this may be seen as add ons or plugins that added to proof of concept will bring the project to life).

7.1.1 *Proof of concept results*

The previously listed steps prove that the designed architecture produces the expected results, furthermore we also conclude in an empiric way what are the best tools and technologies that better suite the project requirements.

In Figure 10, we can observe a network being rendered, this represents the friendship network of a given user. Since there is an entry point user, if we let him in this network we would obtain a egocentric network that could not depict all the surrounding relations in this small society. What we did was to remove this node in order to obtain more clarity to observe the network. At Figure 10 we also can see the interaction of clicking on a certain node and displaying the node information.

7.2 CHOICE OF TECHNOLOGIES

Having the requirements been defined and a small proof of concept been developed as we see in the previous section, we are now able to present our technological choices and provide some context on how we came to these conclusions. We will divide the presentation into four main sections where we present technologies specific to an application layer, starting from database technologies until we reach the front end technologies. For each section

Figure 10: A *screenshoot* of our first proof of concept.

we first list the technologies and then we present the motives that lead us to that specific choices.

7.2.1 *Database technologies*

- **MongoDB**.

Relational databases are one of the complex and advanced technologies that we have today. We have been building our applications on top of these technologies with very strict rules that allow our data to remain coherent through applications lifetimes. Databases engines such as MySQL, PostgreSQL and SQL Server are good live examples of the relevance of these technologies. Meanwhile, applications have grown not just in size but also in complexity, the *web era* came, and with it the need for tools that allow us to manage unstructured data. Other alternatives to relational databases have emerged, today known as **non relational databases** (also known as NoSQL databases). These are database engines that allow us to store unstructured data or store data in a non relational way.

We use **MongoDB** (Home page, 2009) (a document oriented database) to store data, this

gives us more flexibility in manipulating complex JSON structures that are persisted in documents. These flexibility and interoperability would be considerably more complex to achieve using relational databases.

7.2.2 *Back-end technologies*

- **Flask**;
- **Python**,
- **networkx**;
- **PhantomJS**;
- **Selenium WebDriver**;
- **XPath**.

The main language that will support our back-end is **Python**. The choice for this language came very naturally since Python is one of the most used programming languages in the data science field along with others such as R or Java. We also choose Python for two other main reasons: first, we will be building data scrapper modules that need to simulate browser interactions. For that we will use **Selenium WebDriver** (Documentation, 2013) for browser automation and interaction (with the complement of **PhantomJS** (Hidayat, 2013), a headless browser with a Javascript API), and Python integrates very naturally with these technologies, along side with **XPath** (Clark et al., 1999) for querying HTML pages and narrow extraction to the essentials; second because **networkx** (Hagberg et al., 2013) that, already presented in Chapter 4, is written in Python and is a Python module. networkx is the most popular and powerful library that offers a large range of metrics and algorithms to run against graphs that come *out of the box*.

For networking, to make our back-end services available through web APIs, we will use **flask** (Ronacher, 2015), a micro-framework for building simple networking applications in Python.

7.2.3 *Middleware technologies*

- **NodeJS**;

Sometimes we just need something very specific to perform some networking middleware operations, for this purpose NodeJS (Home page, 2017) is an emerging technology that has been famous for performing well this kind of tasks. For bridging between our

back-end and front-end we might need some small pieces that act as *glue* between these two larger components, we will use NodeJS for that purpose.

7.2.4 *Front-end technologies*

- **HTML**;
- **Javascript**;
- **CSS**;
- **D3.js**;
- **React**;

Since we are building a web application, we automatically address to three main technologies that need no introduction, these are HTML, Javascript and CSS. In complement, we will use for our specific needs, that consist in building interactive graphs, a web data driven document representation system, **D3.js** (Bostock, 2012). In what concerns to visualization D3.js will be our main third party, that will bring us many features to help us on network representation and graph interaction [1]. In order to improve our application performance and also the development process, we choose a modern web library, React (Facebook, 2017).

7.3 IMPLEMENTATION DETAILS

In this section we will explain with more detail the more important parts of the system, how they were implemented with some technical notes and more importantly how they interact within our architecture. We show some of the main workflows of the Socii tool in order to clarify that.

7.3.1 *Extraction (web crawlers)*

The web crawler is the module that will allow us to extract data from OSNs. Each OSN has its own web site so each web crawler module has its own implementation, still the extract operations are wrapped in a single API as we described in Chapter 6 when explaining the requirements for the web crawlers. The workflow for extracting some user is the following:

- Login with user credentials;

1 in our proof of concept we already used D3.js for rendering the network as we have demonstrated in Figure 10

- Go to a specific page within its social profile;
- Perform extraction on that page.

These listed three steps may be repeated more than one time since we extract information from different pages. In the following explanations we will be referring to Facebook implementation details for clarification only.

Login

This piece of code performs the login:

```
1  """
2  sleep_extra allows multiprocesses not conflit on login in different
3  browsers (avoid logging in at the same time)
4  """
5  def login_facebook(driver, email, passwd, sleep_extra):
6      driver.get(CONST.FACEBOOK_LOGIN_URL)
7      time.sleep(CONST.BEFORE_LOGIN_SLEEP_TIME + sleep_extra)
8      driver.find_element_by_id('email').send_keys(email)
9      driver.find_element_by_id('pass').send_keys(passwd)
10     driver.find_element_by_id('loginbutton').click()
```

code/login.py

Here we see that browser driver is used to navigate between pages and selector functions such as *find_element_by_id* (line 8 to 10) are used to access DOM elements.

Perform extraction of facebook friends list

The following function extracts a list of Facebook friends:

```
1  """
2  get friends list for a given user identified by the passed uid
3  """
4  def get_friends(driver, uid):
5      go_to_all_friends(driver, uid)
6      total_friends = _get_totaln_of_friends(driver)
7      _load_all_friends(driver, (total_friends/CONST.MAX_FRIENDS_DISPLAYED))
8
9      friends = driver.find_elements_by_xpath("//div[@class='uiProfileBlockContent
           ']")
10     friends_list = []
11
12     for friend in friends:
13         tmp = friend.find_element_by_xpath(".//a[not(@class)]").get_attribute('
               href').split(CONST.FACEBOOK_BASE_URL)[1]
14         if tmp:
```

```
15              try:
16                  res = re.search("id=([0-9]+)?", tmp)
17                  friends_list.append(res.group(1))
18              except:
19                  try:
20                      res = tmp.split('?')[0]
21                      friends_list.append(res)
22                  except:
23                      pass
24
25      return friends_list
```

code/facebook_friends_list.py

Here we see some helper functions where the names are sufficiently explicit in order for one to understand the goal of this function without consulting the others. First we navigate to the users' friends list, then we get the total number of friends and call **_load_all_friends** (line 7) that will scroll the friends' list in order to load all friends in the same page (this needs to happen because Facebook uses lazy loading of friends in order to not load them at once in the web page). Next we get the container that wraps the friends list (line 9), loop for each block that holds some friend info (line 12) and for each one of them we extract the **user id (uid)** from the link to the friend's profile, this may be a numeric value (line 16) (code in the *try* block) or it can be a string that comes as first parameters in the URL query (line 20) (*except* block).

7.3.2 *Network generator*

Network generator is a very purpose-oriented piece of software in this architecture. Its only goal is to produce data sets (users) for a given OSN. This component is implemented in *Node JS* and it uses faker (Marak, 2014) in order to generate data for a given data schema. Next we show a simple schema for Facebook data generator.

```
var schema = {
    'uid': 'number',
    'livesIn': 'city',
    'lifeEvents': 'lifeEvents',
    'birthDate': 'restrictedDate',
    'likes': 'facebookLikes',
    'relationships': 'facebookRelationships',
    'from': 'cityCountry',
    'name': 'name',
    'gender': 'gender',
    'age': 'number',
    'posts': 'facebookPost'
```

```
};

var facebookPostSchema = {
    timestamp: 'dateRecent',
    description: 'text',
    reactions: 'facebookReactions',
    comments: 'number',
    shares: 'number'
};
```

code/fb_schema.js

We also add some threshold restriction to some values in order to not produce extremely unrealistic data.

```
var facebookThreshold = {
    MAX_FB_LIKES: 20,
    MAX_FB_POSTS: 10,
    MAX_FB_REACTION_VALUE: 1000,
    MAX_FB_POST_COMMENTS: 1000,
    MAX_FB_POST_SHARED: 1000,
    // There is a 30% chance of no reactions for a given post
    NO_REACTION_PROB: 30,
    // There is a 60% chance that a given post only receives likes (as reaction)
    ONLY_LIKES_REACTION_PROB: 60,
    // There is 40% chance that some post is not commented
    NO_COMMENTS_PROB: 40,
    // There is 50% chance that some post is not shared
    NO_SHARES_POST: 40,
    MAX_FB_LIFE_EVENTS: 10
};
```

code/fb_restrict.js

Given a data model and a set of restrictions we generate sets of contextualized data for a given OSN (in the previous examples, for Facebook).

7.3.3 *Network metrics*

As we mentioned already in this Chapter (Section 7.2) we use networkx (Hagberg et al., 2013) to perform metrics calculations such as centrality measures (betweenness, degree, eigenvector etc.), node rank or clustering against a network that is fed into this API. This micro service is very simple and could practically be seen as a wrapper to networkx methods. In the API we made available an endpoint **/metrics** that is available to receive metrics

requests. Below a sample payload that the **/metrics** endpoint is expecting in order to calculate metrics.

```
{
    directed: {boolean},
    graph: {name: "Graph's Name"}
    links: [{source: 0, target:1}, ...],
    nodes: [{id: 0}, ...],
    metrics: [
        'averageClustering',
        'betweennessCentrality',
        'closenessCentrality',
        'clusteringCoefficient'
        'degree',
        'degreeCentrality',
        'eigenvectorCentrality',
        'pageRank'
    ]
}
```

code/metrics_payload.js

If no metrics array is requested all metrics will be computed, this is a fallback/default API behavior.

The response is a mapping between each node and the respective computed metrics, and a **global** object that holds metrics for the graph.

```
{
    "0": {
        "betweennessCentrality": 0.006629585250616832,
        "closenessCentrality": 0.29365513766170576,
        "degreeCentrality": 0.026819923371647507,
        "eigenvectorCentrality": 4.8893348311943 43e-05
    },
    "1": {...},
    "global": {
        "averageClustering": 0.0434353
    }
}
```

code/metrics_response.js

Next we present the piece of code responsible for receiving such request and bridge it with the networkx software wrapper module (our **nx_interface**). The responsibility for computing the metrics is delegated to the **nx_interface** which calls the correct networkx method for computing a given metric.

```
@app.route(CONST.ROUTE_PREFIX + '/metrics', methods=['POST'])
def calculate_metrics_for_graph():
    try:
        payload = request.get_json()

        if 'multigraph' not in payload:
            payload['multigraph'] = False

        # Check for metrics, if no metrics assume all available as default
        metrics = []
        if 'metrics' not in payload:
            metrics = CONST.NX_AVAILABLE_NODE_METRICS + CONST.
                NX_AVAILABLE_GRAPH_METRICS
        else:
            metrics = payload['metrics']

        if payload['graph']:
            G = json_graph.node_link_graph(payload)
            return jsonify(nx_interface.calculate_metrics(G, metrics)), 200
        else:
            return jsonify({CONST.MSG: CONST.ERR_INVALID_PAYLOAD}), 404
    except:
        return jsonify({CONST.MSG: CONST.ERR_GENERIC}), 500
```

code/metrics_api.py

Then we simply observe what are the requested metrics and build a response that is divided into two metrics groups. **Node** metrics are node specific, meaning that we will have this metric value for each node; other metrics are global, these are metrics that concern to the network as a global entity (e.g. network clustering coefficient). The next presented method is called one time per metric.

```
"""
Compute a given metric for given graph G
"""
def compute_nx_metric(G, metric):
    if metric == CONST.NX_AVERAGE_CLUSTERING:
        return nx.average_clustering(G)
    elif metric == CONST.NX_BETWEENNESS_CENTRALITY:
        return nx.betweenness_centrality(G)
    elif metric == CONST.NX_CLOSENESS_CENTRALITY:
        return nx.closeness_centrality(G)
    elif metric == CONST.NX_CLUSTERING_COEFFICIENT:
        # Clustering coefficient at specified nodes
        return nx.clustering(G)
    elif metric == CONST.NX_DEGREE:
```

```
15          return nx.degree(G)
16      elif metric == CONST.NX_DEGREE_CENTRALITY:
17          return nx.degree_centrality(G)
18      elif metric == CONST.NX_EIGENVECTOR_CENTRALITY:
19          return nx.eigenvector_centrality(G)
20      elif metric == CONST.NX_PAGE_RANK:
21          return nx.pagerank(G, alpha=CONST.NX_PAGE_RANK_DEFAULT_ALPHA)
22      elif metric == CONST.NX_RICH_COEFFICIENT:
23          return nx.rich_club_coefficient(G, normalized=False)
24      elif metric == CONST.NX_NODE_CONNECTIVITY:
25          return nx.node_connectivity(G)
26      elif metric == CONST.NX_TRANSITIVITY:
27          return nx.transitivity(G)
```

code/nx_interface.py

7.3.4 *Service Aggregator*

Anytime our front end needs to interact with some of the previous micro services, it goes through this service aggregator in order to fetch data or to perform some other data operation. This service aggregator also manages Socii users concerning the authentication process. Next we present a simple generic client that is implemented in our aggregator in order to communicate with the metrics micro service.

```
1  export class SociiMetricsClient {
2      constructor(transport) {
3          this.transport = transport;
4      }
5
6      fetchMetrics(directed=false, graphName='graph', nodes, links, metrics=
           DEFAULT_METRICS) {
7          if (links.length && nodes.length && metrics.length) {
8              return this.transport.post('/metrics', {
9                  directed,
10                 graph: {name: graphName},
11                 links,
12                 nodes,
13                 metrics
14             });
15         } else {
16             return Promise.reject({msg: CONST.ERROR.NOT_ENOUGH_DATA_PROVIDED});
17         }
18     }
19 }
```

code/metrics_client.js

In the Section 7.4 we will explain the main workflow that takes place inside the service aggregator, and there more details about this component will be provided.

Middleware optimizations

Some optimizations were put in place considering that this is the main networking component, where more traffic flows. The considered optimizations consist in using an existent middleware third parties in order to mitigate high payload exchange (between front-end and aggregator, and also between aggregator and other micro services). We use the **compression** [2] NodeJs middleware to achieve optimal payload compression levels.

7.3.5 *Front-end*

Our front-end is built using the react library as we have already mentioned, and the main component of the application consists on a visualization dashboard that displays an interactive network that is rendered using an open source component that we have built, **react-d3-graph** (danielcaldas, 2017) that will be explained in the following sections. Other relevant third parties that we use in our front end are the following:

- ***react-redux***[3] - redux architecture official bindings for React library. This eases the process of manage application state.
- ***axios***[4] - an HTTP client library that makes networking operations more straightforward.
- ***material-ui*** [5] - a library that offers visual components with a pre established look and feel and some built in interactions.
- ***react-bootstrap*** [6] - a complement to the previous library, this one offers also some visual components like overlays and tooltips.

Graph render and interaction component with react-d3-graph

> *"React component to build interactive and configurable graphs with d3 effortlessly"*

2 https://github.com/expressjs/compression
3 Official React bindings for Redux https://github.com/reactjs/react-redux
4 Promise based HTTP client for the browser and node.js https://github.com/mzabriskie/axios
5 React Components that Implement Google's Material Design http://www.material-ui.com
6 Bootstrap 3 components built with React https://react-bootstrap.github.io/

From an end user perspective the most relevant and valuable aspects of a SNA tool are powerful network interactions and clean visualization (and by clean we mean perceptible). In order to achieve this in Socii our efforts were directed into a generic, reusable and configurable software: **react-d3-graph** (danielcaldas, 2017). This component allows us to focus only on the visualization features and in how we want to represent and interact with our network (graph). This component is built on top of the react library and creates configurable abstractions upon D3.js. All the detailed documentation [7] of react-d3-graph may be consulted on the web, also a live demo[8] with the all the possible public configurations offered by react-d3-graph is also available.
With this development we will isolate all the visualization concerns leaving Socii with less work on this matter. Also Socii can wrap this component and use the available user actions to interact with the graph, and all the visual aspects such as nodes colors or text size may or may not be editable by a Socii user. Socii has the power to decide the level of granularity of user control upon graph configurations, but of course since we want a simple application we will leave to Socii how the network looks like exposing some basic interactions to the end user such as node click, zooming, mouse overing and drag and drop on all the network or upon a specific node.

7.4 Main Workflows

In this section we describe two of our main workflows within Socii tool. The first and most relevant workflow is the network aggregation and rendering that has within its main intervenient the aggregator. Next sections clarify the role of this aggregator. Then we will present a front-end only interaction that consists in coloring nodes with given properties.

Network Aggregation and Rendering

In the diagram of Figure 11, one may observe what happens since the user requests that a certain OSN network is built. In the next explanation we will be referring to the workflow steps numbers in the figure.

First the user configures the network choosing the number of nodes for a certain OSN and what metrics he wants to calculate against the network and then see in the network visualizer area. Then the user presses a button that orders network build (**1**, **2**), and waits until all the aggregation operations are performed.

First the service aggregator fetches a set of N requested users from the **socii-generator** (**2.1**) microservice, then based on the retrieved information it builds the network nodes and links data structure and performs some other minor operations (**4,5,6**). Once the previous

7 https://danielcaldas.github.io/react-d3-graph/docs/index.html
8 https://danielcaldas.github.io/react-d3-graph/sandbox/index.html

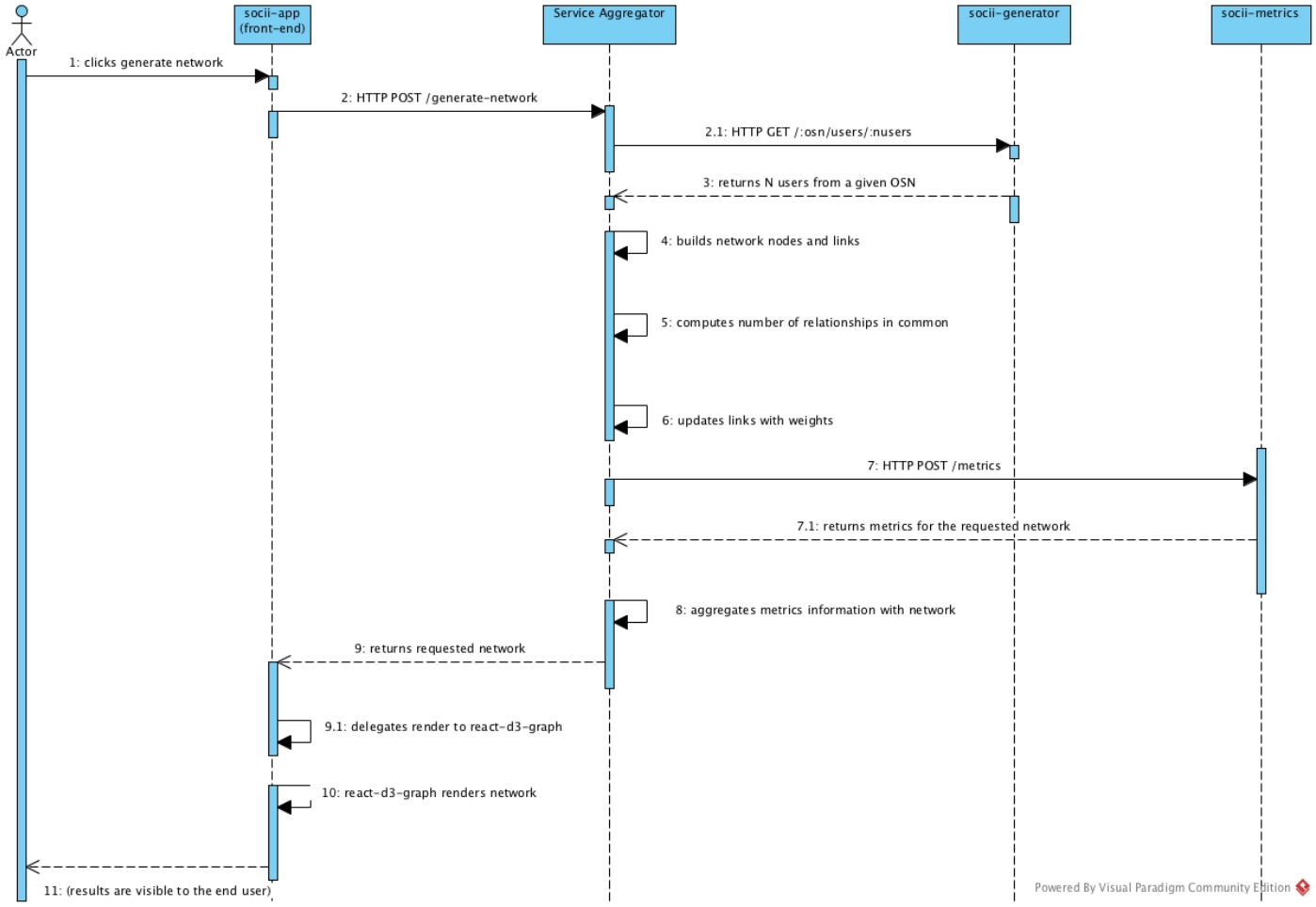


Figure 11: In this figure we may observe Socii sequence diagram for building a network.

operations are completed a request is made to the **socii-metrics** microservice in order to collect the metrics for the previous built network. The returned metric values are aggregated with the users, nodes and links within a network data structure (**8**) and a response is sent to the front-end (**9**). The front-end only needs to take the nodes and links that the aggregator puts together and delegate the network rendering to the react-d3-graph component (**10**), and that's it, the user has now available the full network with all the aggregated information (**11**).

Community detection with node coloring by property

In this case we present a more simple workflow that happens only on the front-end. The community detection is a module that allows the user to paint nodes that have some property in common. For example, lets say we want to color our network according to where the users live, we will have to associate a unique color per city and then map these colors to

the nodes painting each node with the color that corresponds to the corresponding user's city. This algorithm is generic so we can reproduce the same of other properties such as birthday, name or gender.

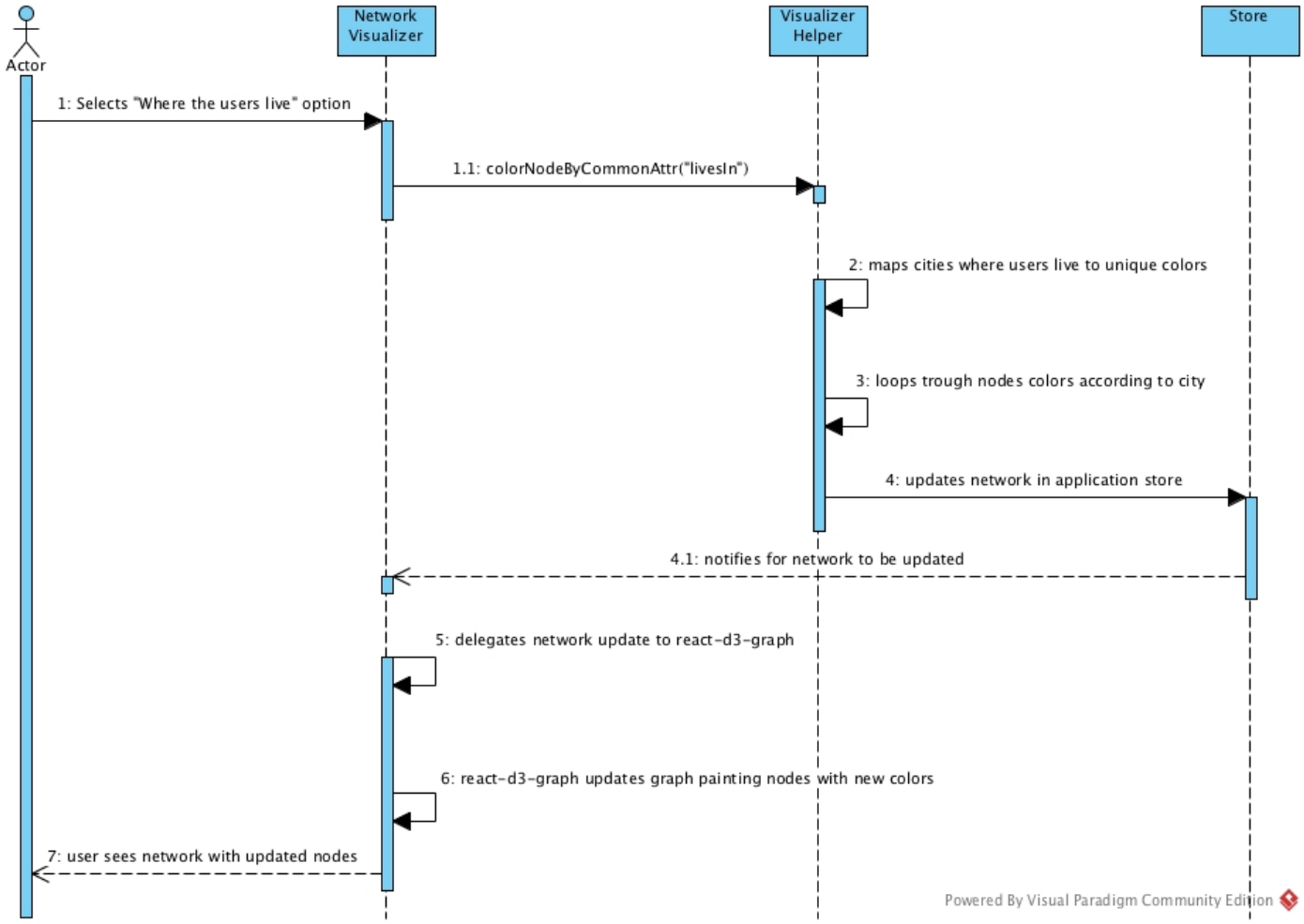


Figure 12: Socii sequence diagram community detection based on node properties.

As we can see in Figure 12 the process is simple, the user chooses an available property for community detection (**1**), then some logic helper front-end component takes the request and maps the different cities to a unique color (**2**). The next step consists in painting user nodes according to the color of their city (**3**) and update the store (the store is the object that holds the network state, this includes nodes and their properties). When the store is updated the network visualizer component is notified and delegates the new node properties to react-d3-graph that handles visual updates (**5 and 6**).

8

FINAL RESULTS

In this Chapter we will present the results of the implementation of the requirements previously specified. As we previously mentioned we chose a set of requirements (among a big list of possibilities) that would allow us to have a tool with the most relevant and core functionalities in order to prove our hypothesis of building a web-based SNAs tool.

Along the Chapter we will not only list the system outputs (considered the project outcomes) but we will also illustrate them; in some of the following sections we present a set of Socii images that depict the implemented functionalities and how to use them. Then, we present some case studies, concerned with the analysis of a real network that enabled us to draw some different conclusions that demonstrate well some of the use cases of Socii defined previously.

As the development of the tool depended on the OSNs integration and real data analysis, we had to somehow manage to fed real data sets into Socii. The way we achieve this was not a totally automatic process, we used the crawler modules and extraction APIs explained in Chapter 5 and 6, to extract real data sets into a local filesystem, and then with a migration script we pointed to the production MongoDB instance in order to store the real data so that Socii aggregator component could get the real data. To improve data feeding interoperability we have two functionalities working side by side, the user may choose to analyze a real network (previously extracted by the mentioned process) or the user may choose to generate a network with data from our generators module.

The following table summarizes the features that were implemented on Socii. All the "**MUST**" requirements were implemented and two additional "**SHOULD**" requirements were also implemented since they were almost cost free once we had the component react-d3-graph in a more advanced stage. To sum up, we will present a concise table that points to each requirement with the proper reference created in Chapter 6.

Requirement	Short Description	Status
6.3.2, 1	Socci login	✓
6.3.2, 2	Order network build	✓
6.3.2, 3	Extraction feedback	✓
6.3.3, 1	Render network	✓
6.3.3, 2	Community detection (visually identifiable)	✓
6.3.3, 3	Drag and Drop all network	✓
6.3.3, 4	Zooming interactions	✓
6.3.3, 5	Interactive node comparison	✓
6.3.3, 6 (should requirement)	Global network interactions (Toolbar)	✓
6.3.3, 7 (should requirement)	Disable heavy animations	✓
6.3.2, 1	Network Generator (Facebook)	✓
6.3.2, 1	Network Generator (LinkedIn)	✓
6.3.2, 2	Network Generator - Configuration	✓
6.3.3, 4	Zoom network	✓
6.3.3, 5	Detect heavy network and disable animations	✓
6.3.4, 1	Render label along side each node	✓
6.3.4, 2	Highlight node and adjacent connections	✓
6.3.4, 3	Node click and show information (network metrics and information in the context of the OSNs)	✓
6.3.4, 4	Drag and Drop nodes	✓
6.3.5, 1	Render network links with *semantic thickness*	✓
6.3.8, 1	Download network in the format GraphML	✓

Table 3: Summarization of Socii features.

In Table 3 we present the actual list of implemented requirements, taking into account the requirements introduced in Chapter 6. All the requirements are **MUST** requirements unless otherwise stated.

8.1 SOCII - FINAL ASPECT AND FUNCTIONALITIES

In this section we do an overview across Socci application, we present the overall functionalities that Socii offers from an end user perspective.

8.1.1 *Network Configuration Area*

Entry state

In Figure 13 we may observe our initial page where we display available OSNs that can be configured in these area and then a network generation or network calculation upon

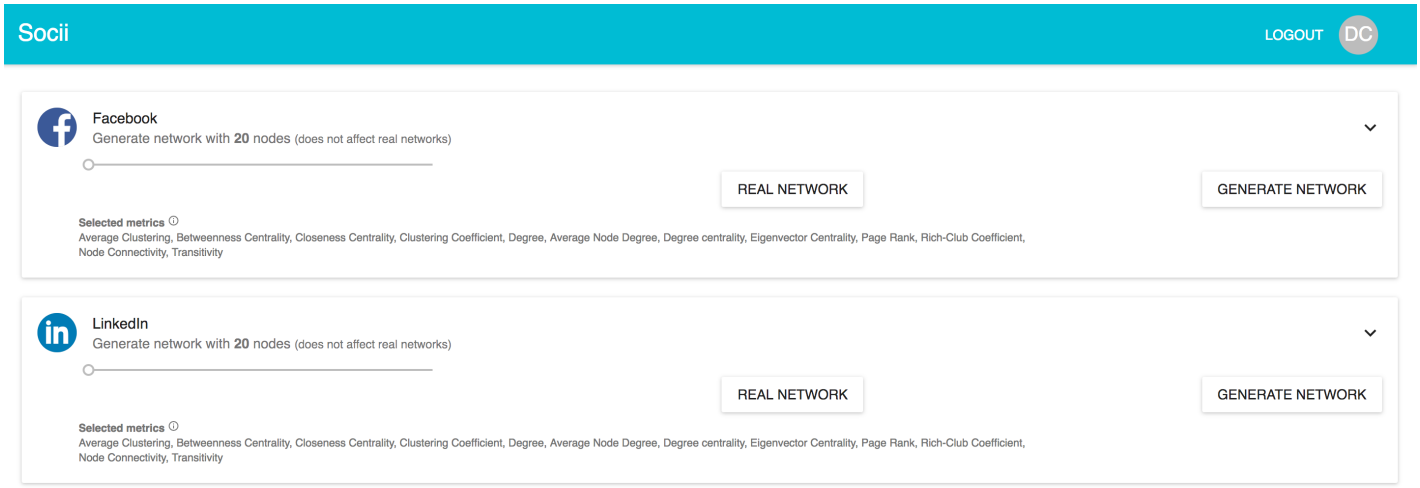


Figure 13: Socii landing page. Network configuration area.

real network may be ordered. Each OSN consists in a expandable card that once expanded contains all the details and information about the OSNs and metrics configuration.

Configuration card detail

In Figure 14 Facebook configuration card is expanded and here we see that we can have a great level of granularity upon the SNAs metrics that we want to calculate against a certain network. On the top of the card we have a brief description of the OSN followed by sections that represent sets of metrics, and for each one of the metrics we also provide a brief explanation for a given metric.

These are the metrics that we will then be able to see for each node in the network visualization area. To reach more flexibility we decide on switches visual elements to turn on/off a certain metric, by doing this we may combine any set or subset of metrics. Some of the metrics are global (these mean that they are calculated regarding the graph/network in general) other are node specific (have a meaning at node level), this is implicit within each metric description.

8.1.2 *Network Visualization Area*

The network visualization area is the main area of Socii tool, here is where the user will actually be able to visualize and analyze a given social structure for a certain OSN. As we may observe in Figure 15, we still have the app header where the user can logout at any time, below we have a large area where the network is rendered. At the bottom of the page

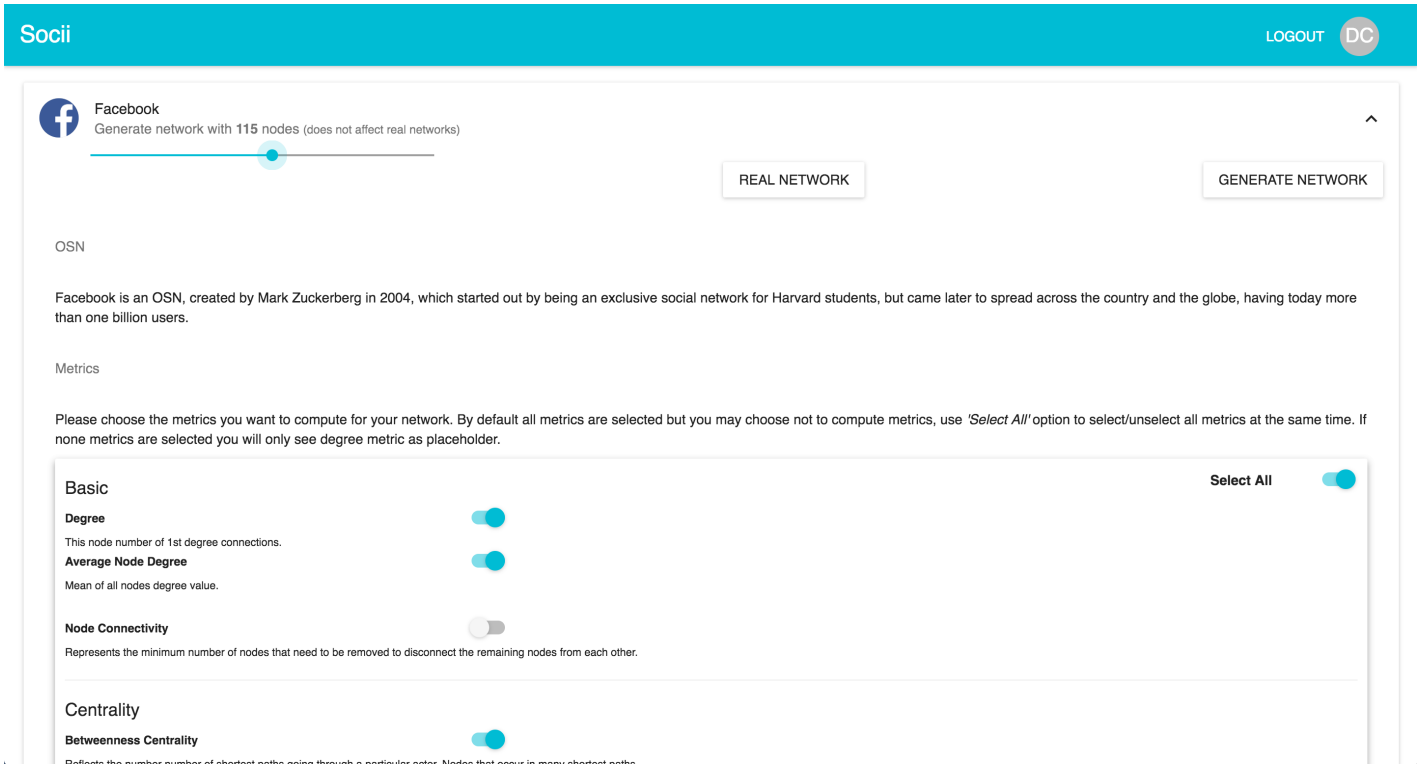


Figure 14: Network configuration area. Facebook configuration expanded.

we have the toolbar that contains among others, that will be explain latter with greater detail, main graph interactions such as pause animations or show/hide nodes labels.

Toolbar

In this particular section we will only focus on the toolbar and on the functionality that this application component offers. As we can see in Figure 16, we may observe an explanation of each icon present on the toolbar, this pop up descriptive dialog appears when the information icon (last icon counting from the left) is pressed, this will allow us to keep the toolbar clean (not having to render descriptions or labels at the side of the symbols) having only the toolbar with the icons and no additional descriptive content.

Next we present the detailed description of each action/icon (icons are enumerated from left to right as displayed in the toolbar).

- **Pause** - the pause icon allows users to stop ongoing animations, this animations happened mainly at the start of the visualization when the network is still being arranged and nodes are being positioned to not overlap (this job is done by D3.js);
- **Refresh/Redo** - clicking this icon will make all the dragged nodes go back to their initial positions (approximately);

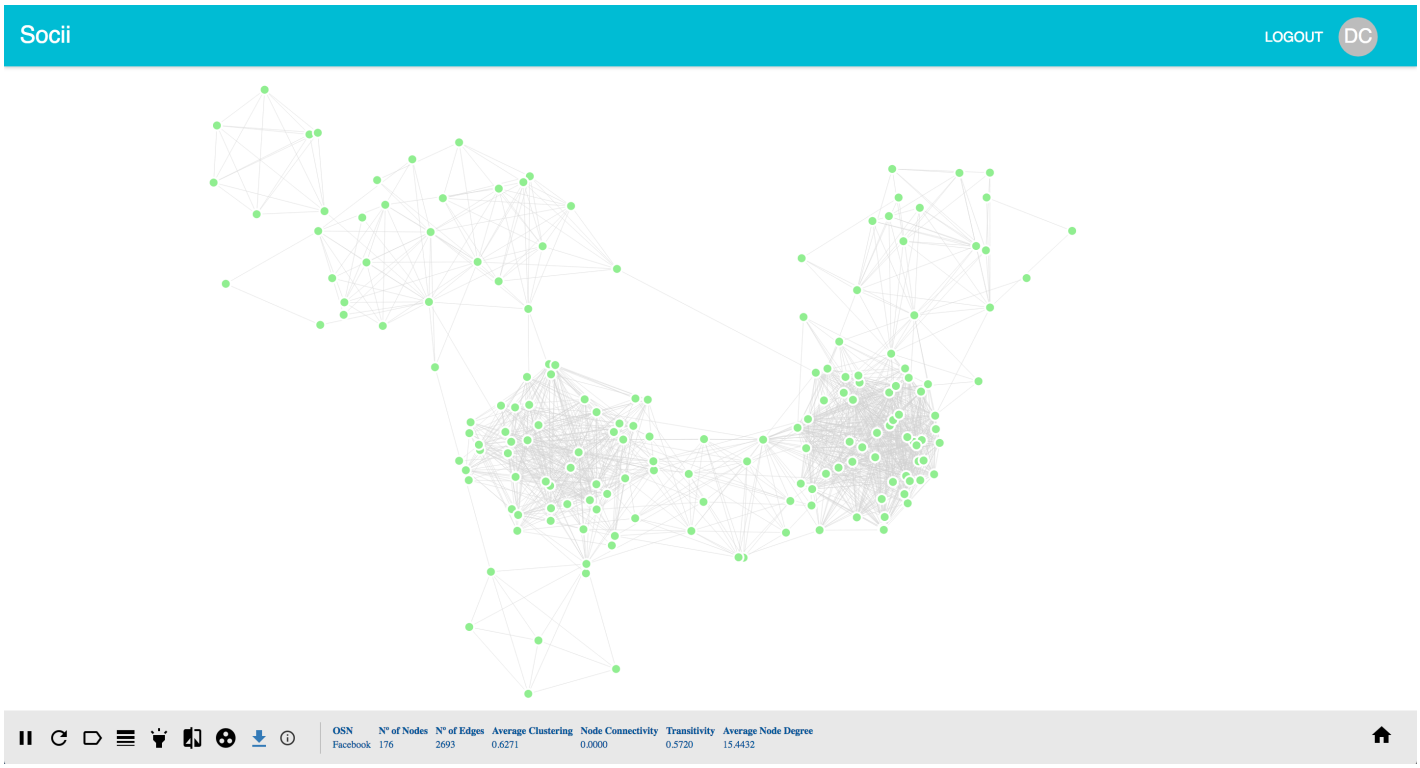


Figure 15: Network visualization area.

- **Label/Etiquette** - clicking in this icon shows and hides the labels of the nodes in the network, the user may use this as he pleases according to his visualization needs;
- **Strokes** - clicking this icon will make nodes connections semantic visually, this means that the link thickness will be propositional to the number of common relations between the two parts;
- **Flashlight** - when this feature is active the user is able to mouse over nodes in the network and to stand out the selected nodes relations (only the mouse hovered node relationships 1st and 2nd degree will be highlighted);
- **Switch/Compare** - activating this feature will make the clicked nodes comparable. This means that when we click on some 1st and then a 2nd node, we will be able to compare this two nodes at the most granular level (this feature is explained in the Node Comparison section);
- **Circle/Community** - clicking here opens a pop up dialog form that allows users to perform some community detection studies on the network, this is explained in the Community Detection section;

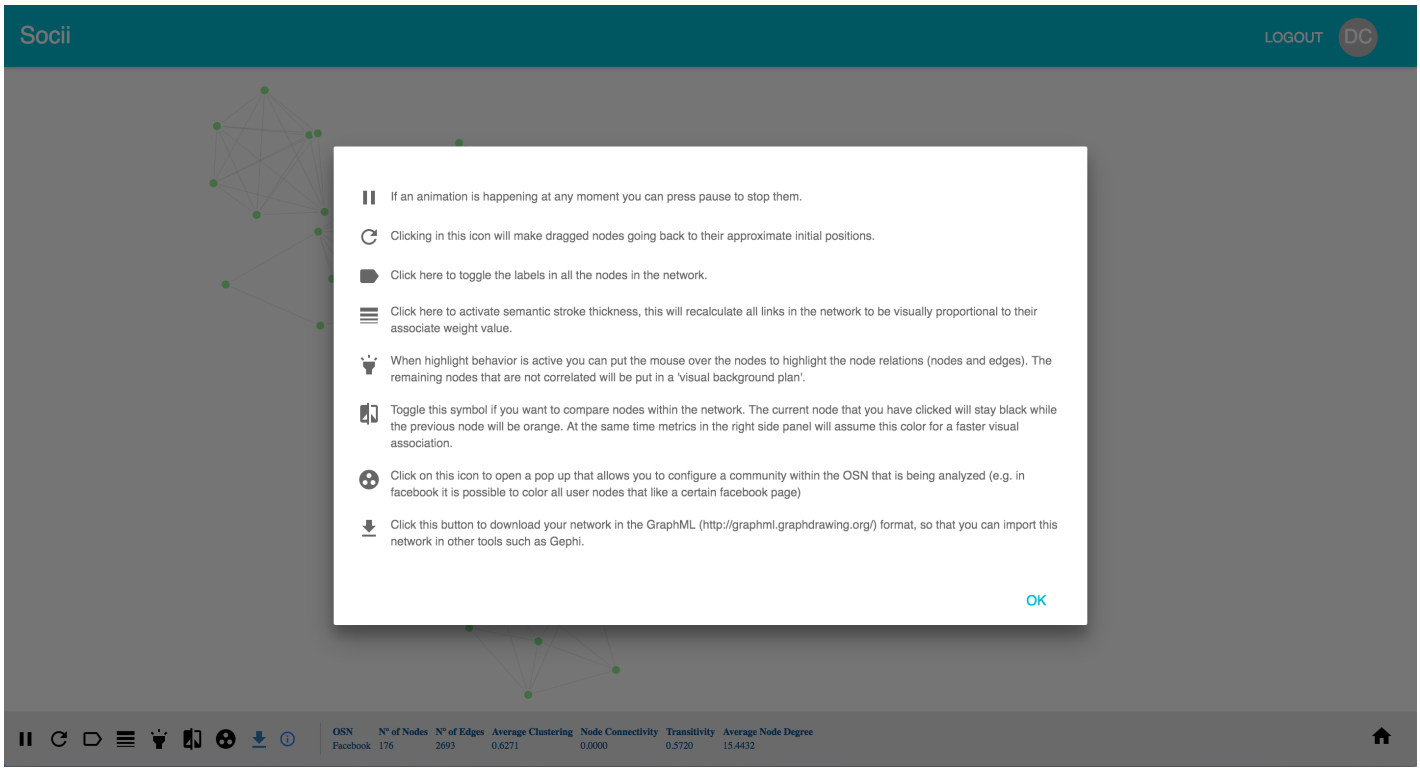


Figure 16: Dialog containing toolbar help information.

- **Arrow/Download** - when the user clicks the downloaded icon a GraphML file is generated and download containing the rendered network basic structure.

After this actions/icons section the toolbar presents some global metadata about the rendered network. This information consists in: the OSN name, the number of rendered nodes and links and some global metrics such as average clustering coefficient and average node degree.

Node Discovery

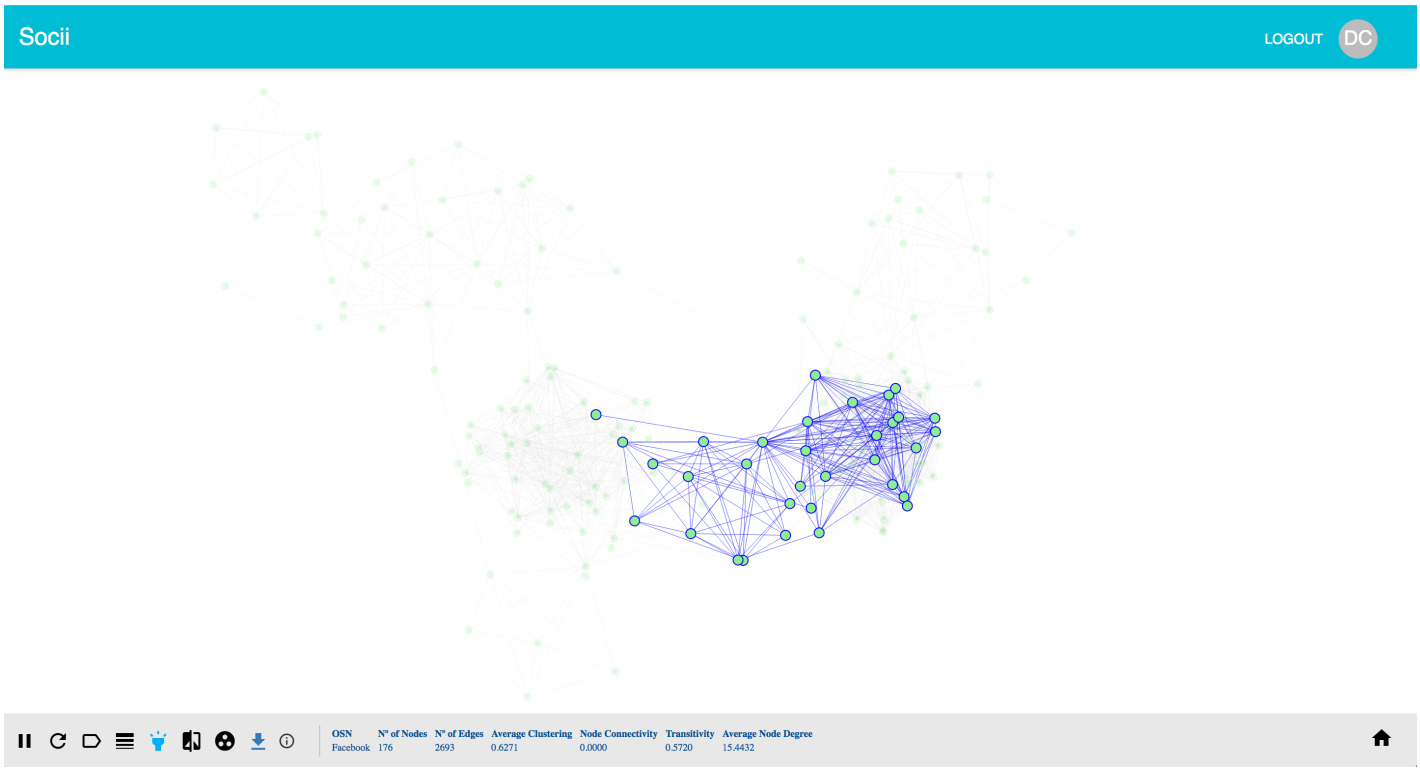


Figure 17: Node discovery feature.

Node discover allows us to stand out some nodes upon the rest of the network. When we mouse over some node, the selected node and its 1st and 2nd degree connections are highlighted, as we can see in the example provided in the Figure 17.

Node Details

When clicking on some network node a right side panel will slide and display all that node information, including the computed SNA requested metrics and the OSN data that appears in a more fashionable way.

Node Comparison

When activating node comparison functionality the user is able to compare two distinct nodes anywhere in the network (this said, they do not need to be connected so that we are able to compare them). But in what really does this node comparison consist? It simply allows us to visualize in a comprehensive manner the metrics and OSN data of each one of the nodes simultaneously in order for the user not having to jump between some two nodes details in order to compare them.

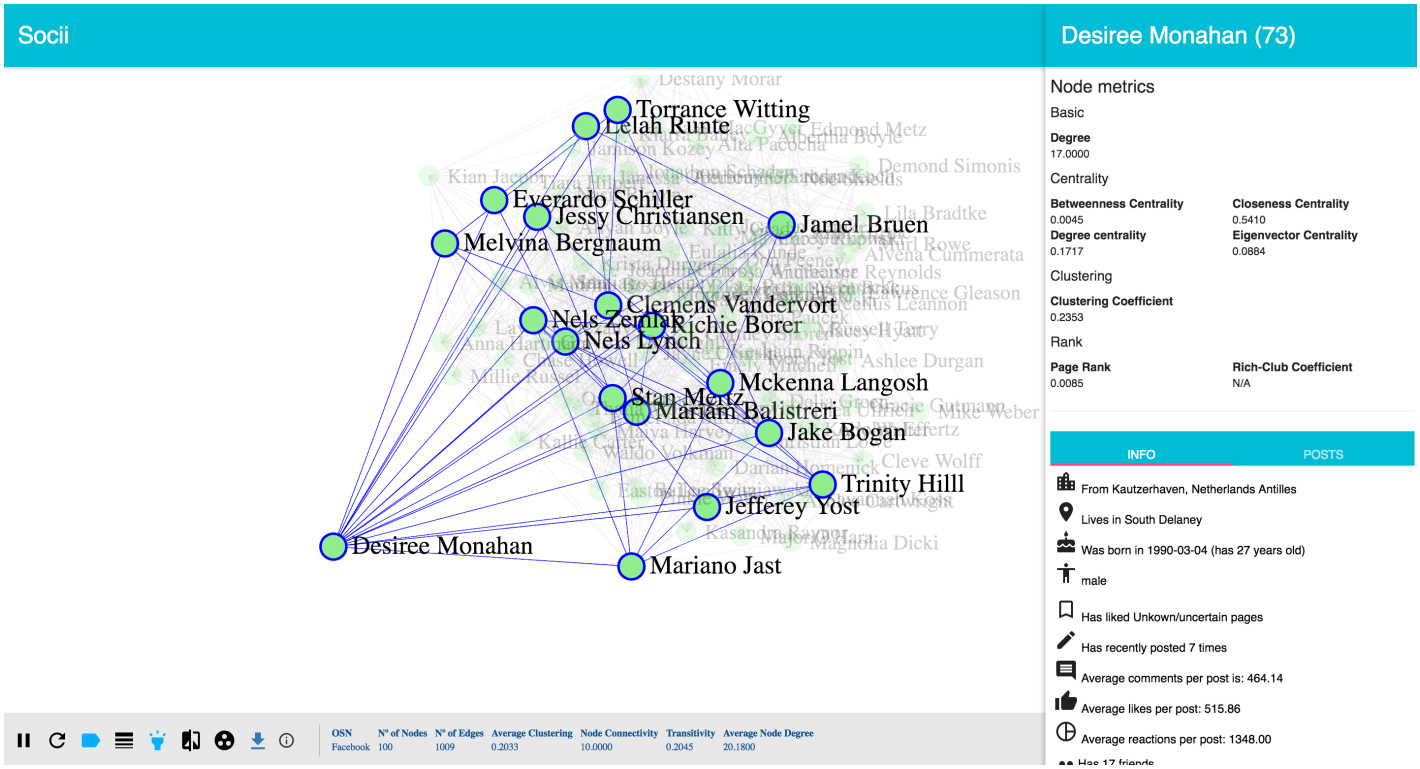


Figure 18: Node details panel opens on the right side of the screen.

As we can observe in Figure 19, we can simultaneously see the metrics of the blue (color that represents the last node that the user clicked), and the the orange node (color that represents the previous selected node).

Community Detection

This feature is one of the most crucial because it allows us to intercept individuals properties/attributes and intercept them at a large scale.

Community detection functionality allows users to color the network according to some individuals OSN specific data property such as *where the users live* or *what the are the users gender*, allowing us to perceive community patterns and identify communities based on a certain assumption. Also we made available a special property that allows users to be colored in terms of what Facebook pages they have liked.

Community detection by user properties

In Figure 20 we may observe how the user may choose to color the network by simply choosing a **"key question"**. The available key questions and the result that they produce are the following:

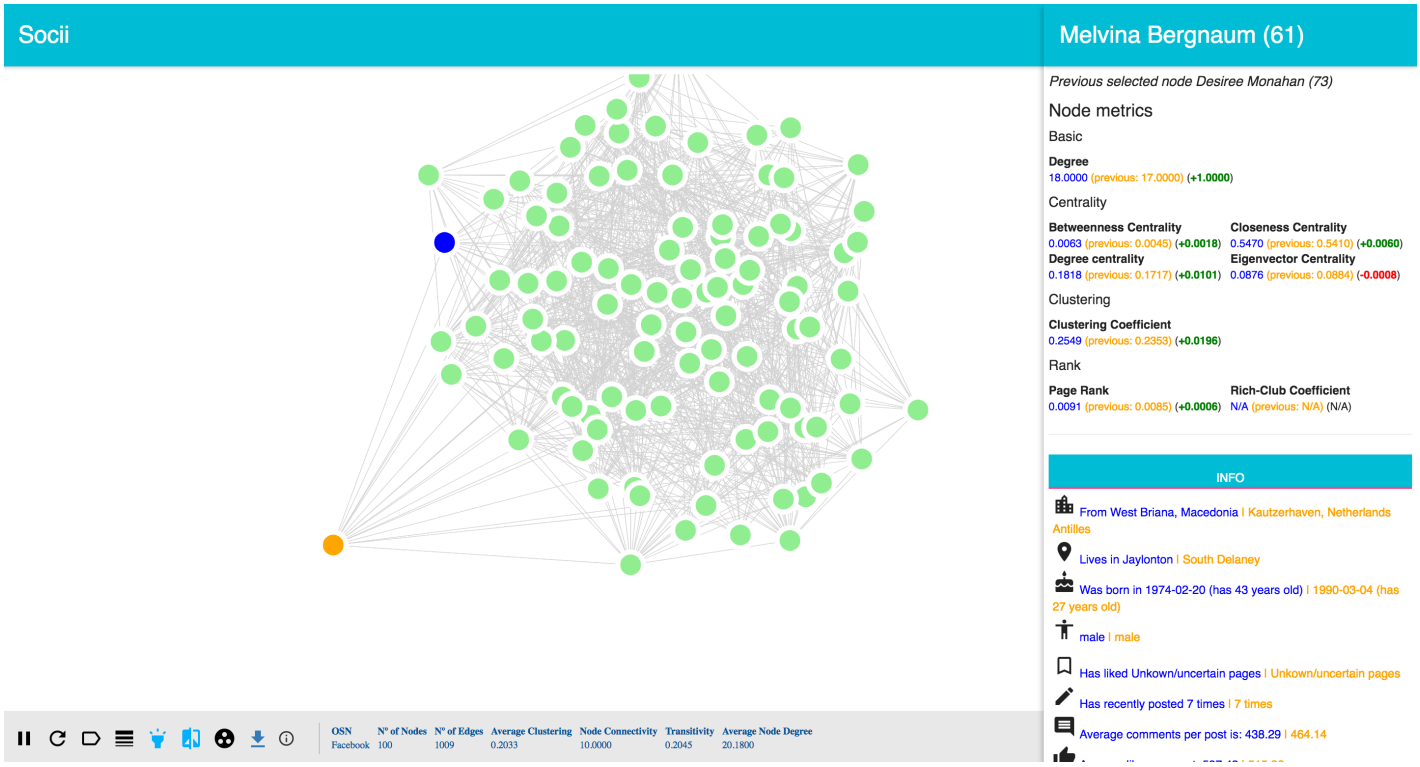


Figure 19: In the right side panel we may observe the detail of the selected nodes.

- **Where do the users live?** - by choosing this option all the users that have a common residence (current address) will be colored with the same color [1];
- **Birth date** - individuals with the same birth date will have the same color;
- **Where are the users from?** - individuals with the same place of birth will have the same color;
- **Gender** - individuals with the same gender will have the same color;
- **Users with the same name** - users with the same name will have the same color.

In Figure 20 we may observe the result of having detected individuals with the same place of birth.

Community by Facebook page likes

By coloring individuals by a specific Facebook page like we can track users preferences within the network, this preferences can be anything (sports, Hollywood stars, food, shoes brand, tv channels, musics, etc.).

1 **Note for future reference**: our color generator mechanism makes sure that no color collision happens, still, some colors may be the same with different tonalities

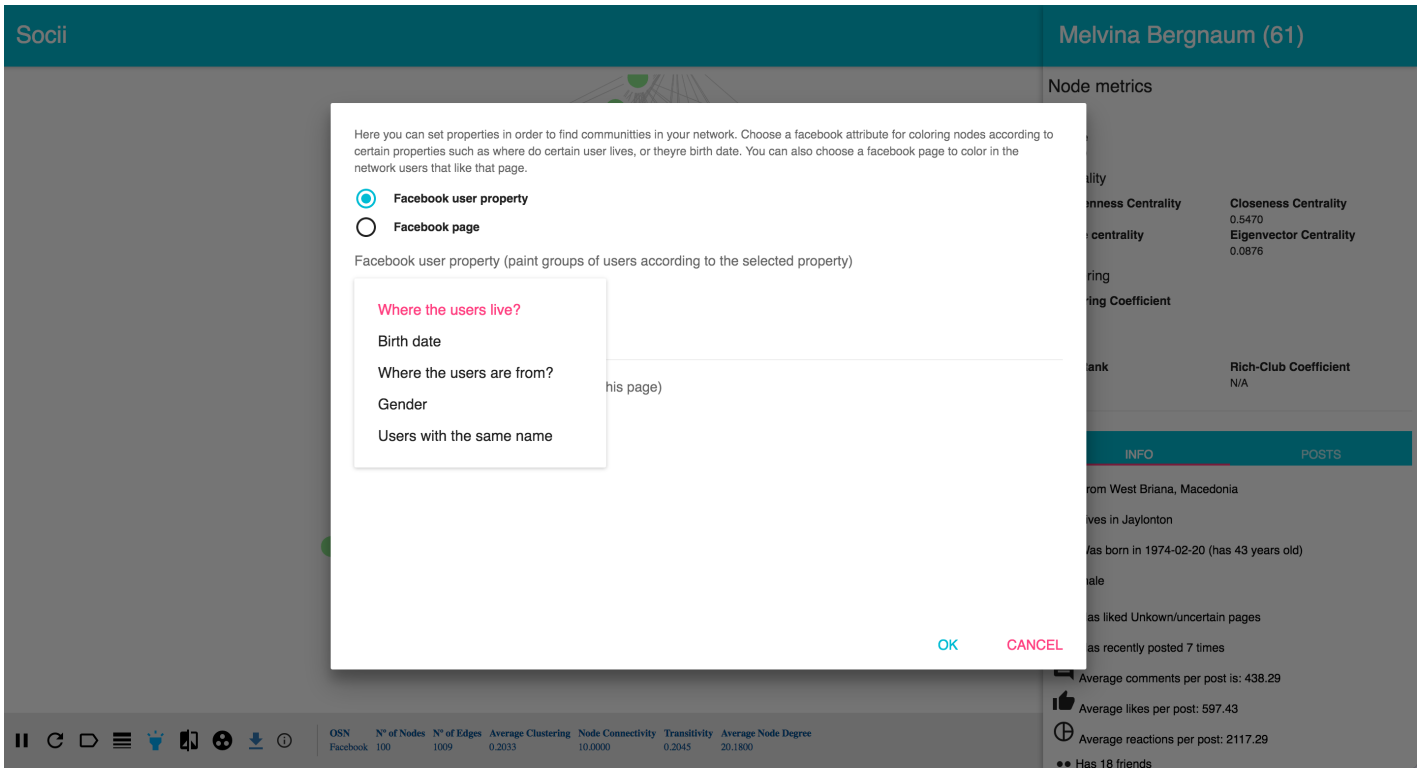


Figure 20: Pop up where user can configure community detection settings.

Figure 21: Community detection for Facebook page likes. *SyfyPT* is a Facebook page.

As we can see in Figure 21 one may select among a variety of Facebook pages, note that after each page id we have the number of Facebook likes within the all network.

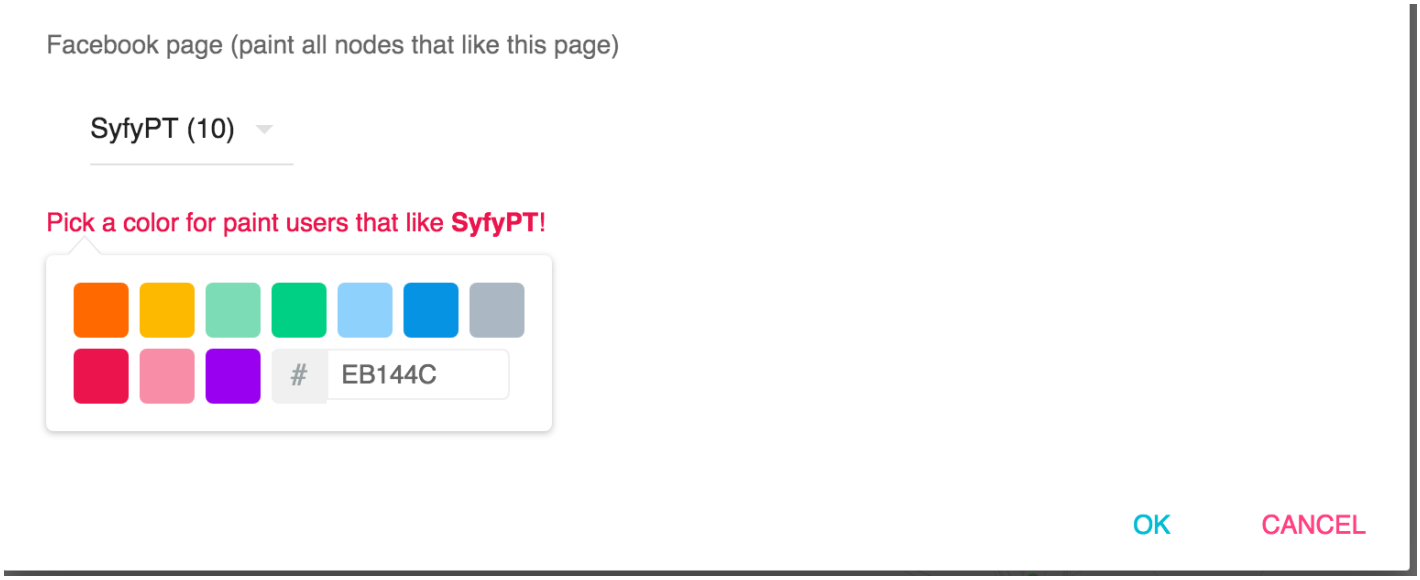


Figure 22: Community detection for Facebook page likes. Picking a color.

After choosing the page from the dropdown in Figure 21 we may see that the user is prompted with a color palette and a color input field. The user may either pick an already available color, or he may choose instead to insert the color code directly in the input field (Figure 22).

Next after the user performs the previous two steps and clicks in the **"OK"** button the nodes in the network are coloring accordingly, a red label (*chip* visual element) appears on the top left corner of the network visualization area so that the user knows what the color represents. In the specific case of Figure 23 all the red nodes represent Facebook users that *like* the **SyfyPT** Facebook page [2].

2 Aside note: if one wants actually to access the Facebook page one just needs to navigate into `https://facebook.com/SyfyPT`, if we have a open Facebook session the page will open in our browser

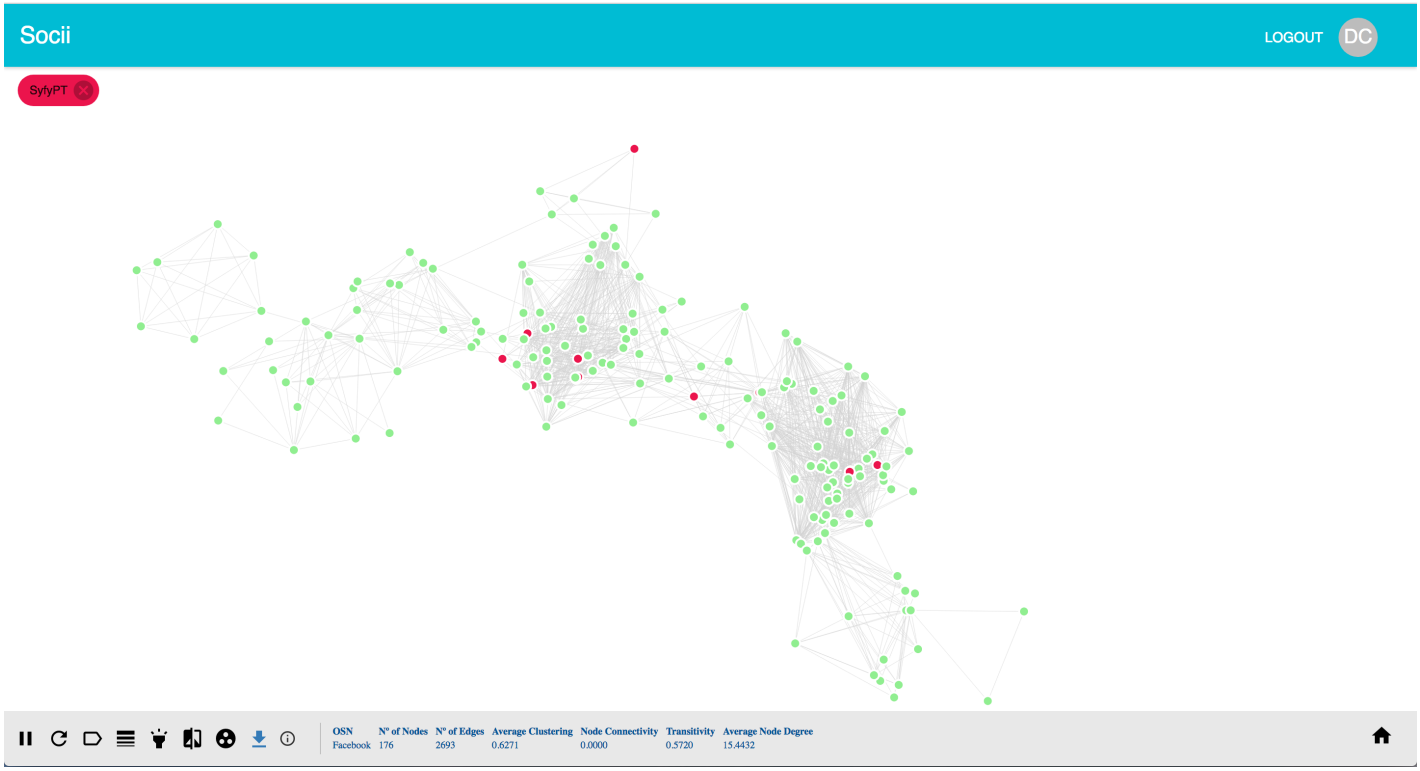


Figure 23: Colored nodes *like* the Facebook Syfy page.

8.2 CASE STUDIES

Now that we exhaustively presented all Socii features, we will present how we can use them and derive some conclusions from network observation and analysis.

As we mentioned Socii uses generators to build a network for a given OSN with a specific required number of nodes. In this section we present some real case studies with real data in order to prove the accuracy of conclusions that Socii provides in a real context. For these case studies we use a real account and extract information from Facebook using the respective web crawler module that was developed initially as a back-end requirement to allow extraction on the fly. However, that not being possible by the mentioned reasons, we use it as a mean to obtain a real data set that we inject in Socii and associate to a whitelisted Socii account that, besides being able to generating OSNs networks as normal users. Also, a real data set is also available as an option to consult a predefined network that is built on the fly with already stored users (the case studies data)[3].

3 Since the data is real some of the images in the following sections may have hidden content in order to maintain anonymity of data. If no data is hidden then all the data in the image is false, it is provided by the generator component.

Detecting active and influent Facebook users

1. Stop the network.
2. Rearrange the network to better suite visualization needs for that one may use features such as network drag and drop, network zooming, and node drag and drop.
3. Turn on node discovery (optionally turn on node labels).
4. Search for central nodes with degree above the average node degree within the network.
5. Turn on node comparison.
6. Compare the previous group of central nodes and choose the one that has the highest eigenvector centrality and betweenness centrality and highest average Facebook reactions per post.
7. If the previously steps led to a specific node we may conclude that we found the most active and influent user in our network. Active because he simply posts with a certain frequency. Influent because of the **interception** of **SNA metrics** that leads us to a specific individual and **OSN Facebook specific data** that tells us that this particular user is socially (online) active and has a significat impact on other users since they react to his posts.

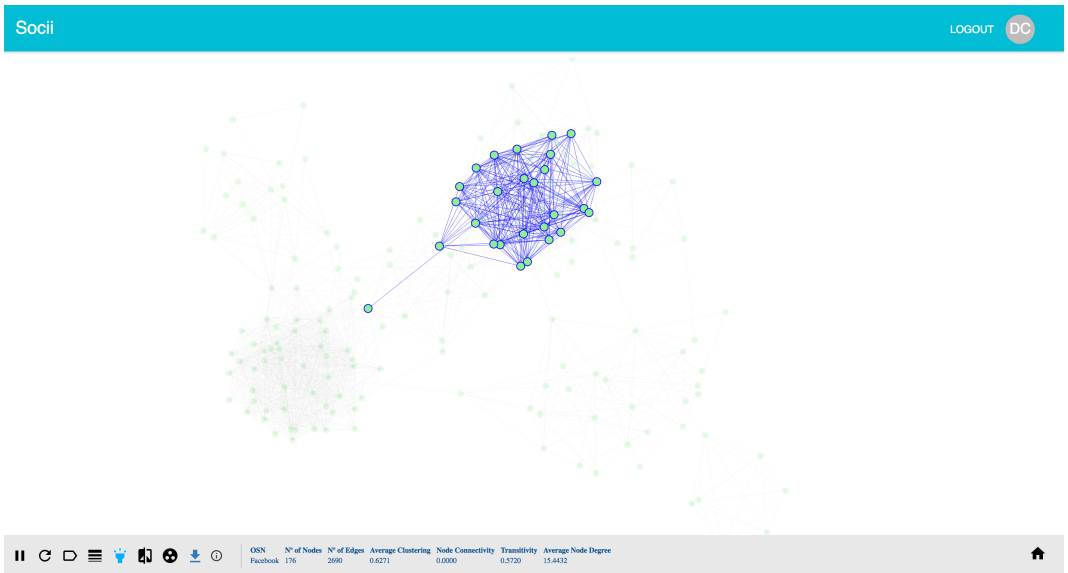


Figure 24: Finding most influent node.

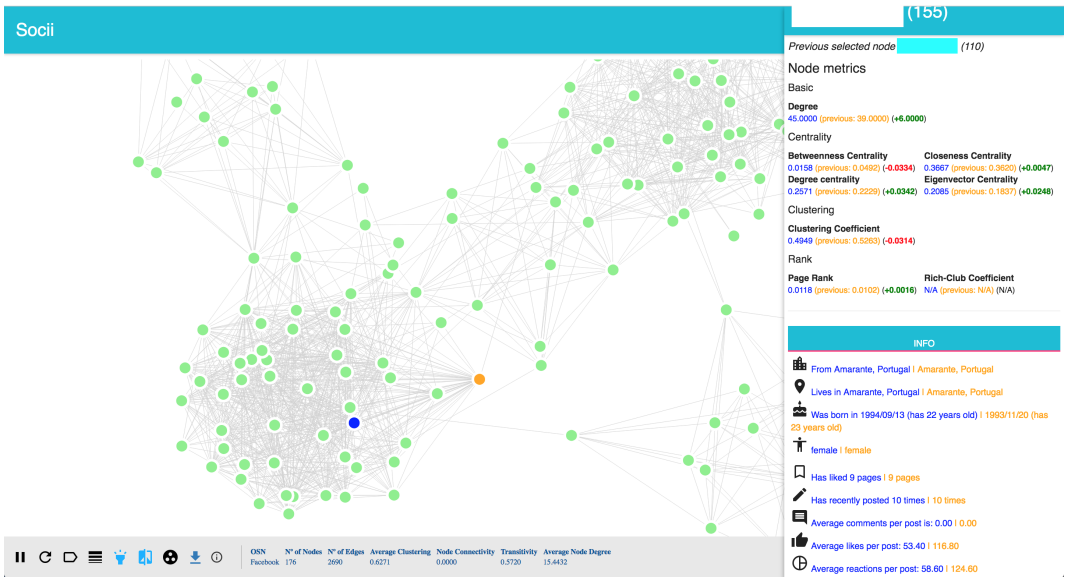


Figure 25: Found most influent and active (in Facebook) node.

Following the previous *"script"* (starting in Figure 24 and ending in Figure 25) one may find using Socii the most influent and active Facebook user. Using node metrics such as **degree centrality** and **average degree centrality** we can easily capture users that have many connections. Also features like network discovery are thought to help users understand nodes underlying connections. OSNs specific metrics (e.g. **average reactions per posts** may indicate us what users are well established in terms of Facebook activity). In this particular case we found that the orange individual is the most influent and active in Facebook (as we can see in Figure 25) where he is compared to another influent individual (blue individual).

Marketing with community detection (Facebook)

A real example within the case studies. Study community preferences.

1. Stop the network.
2. Rearrange the network to better suite visualization needs. For that one may use features such as network drag and drop, network zooming, and node drag and drop.
3. Turn on node discovery (optionally turn on node labels);
4. Open community detection dialog and find the marketing target (this is a Facebook page, it may be a brand, a tv channel, etc.);
5. Color the nodes that like this certain marketing target;

6. Within the colored nodes find the most active and influent (we explained how to obtain these nodes in the previous section).

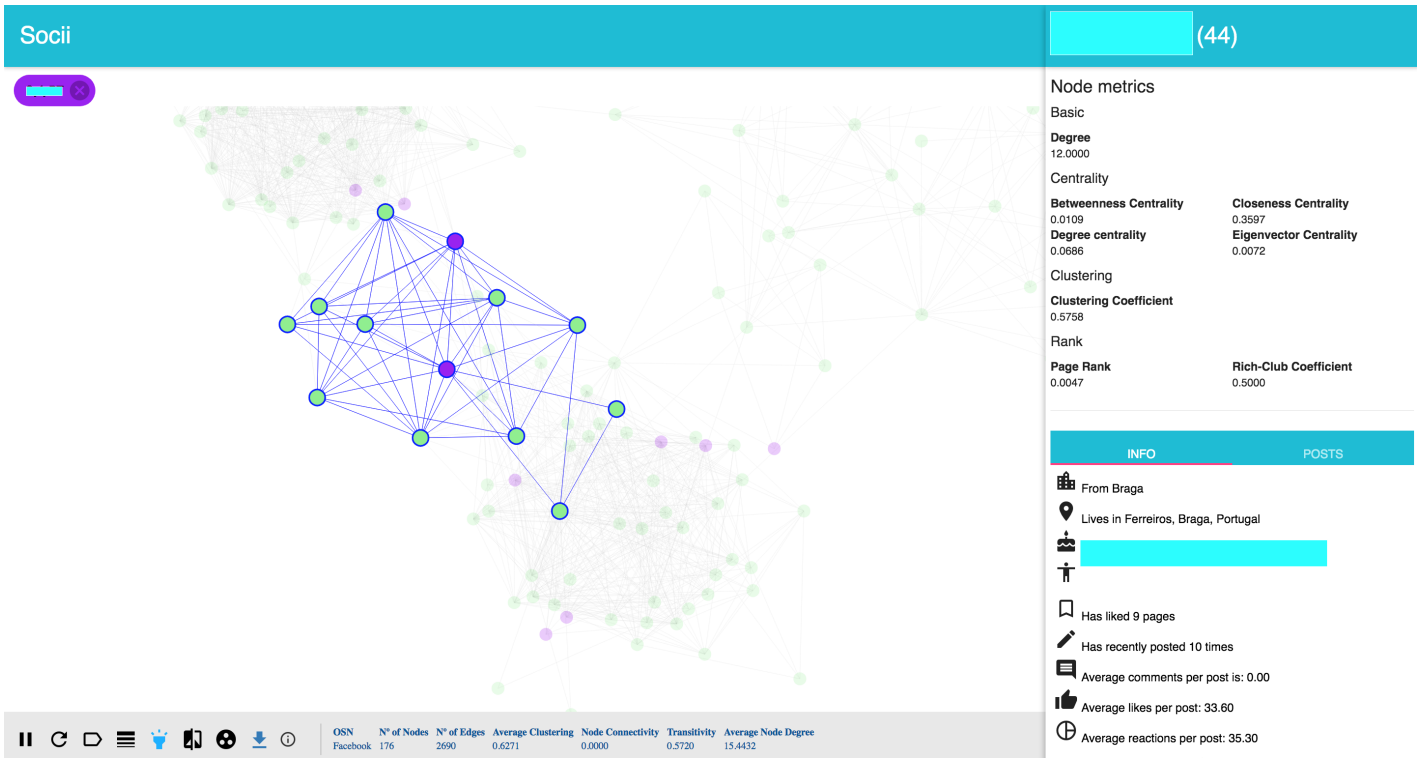


Figure 26: Found target individual to propagate the purple brand.

By performing the following steps we may then plan a brand market strategy upon a certain group of individuals or particular individual (as we can see in Figure 26) that may be implemented in all sorts of forms such as:

- Send newsletters directly to the target users;
- Try and reach the target users personally;
- Save data and cross results with another networks in order to launch public campaign.

Professional discovery with LinkedIn Network

As we did previously with Facebook page likes and other node coloring strategies to color nodes given a certain property, we reproduce a very similar feature in LinkedIn networks. We used community detection to detect nodes that possess a certain skill, this feature could be developed in order to achieve an optimal talent discovery tool but for proof of Socii functionality our simply community detection method will suffice. Also in this section we

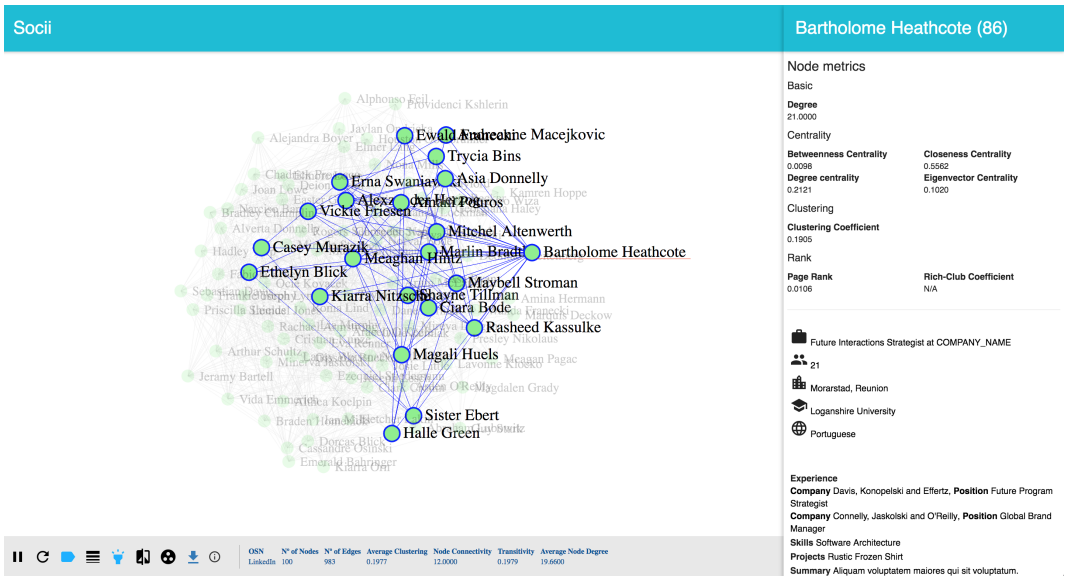


Figure 27: Rendering LinkedIn network and visualize node detail.

will proof the adaptability of our tool by rendering a LinkedIn network, and as we see in Figure 27.

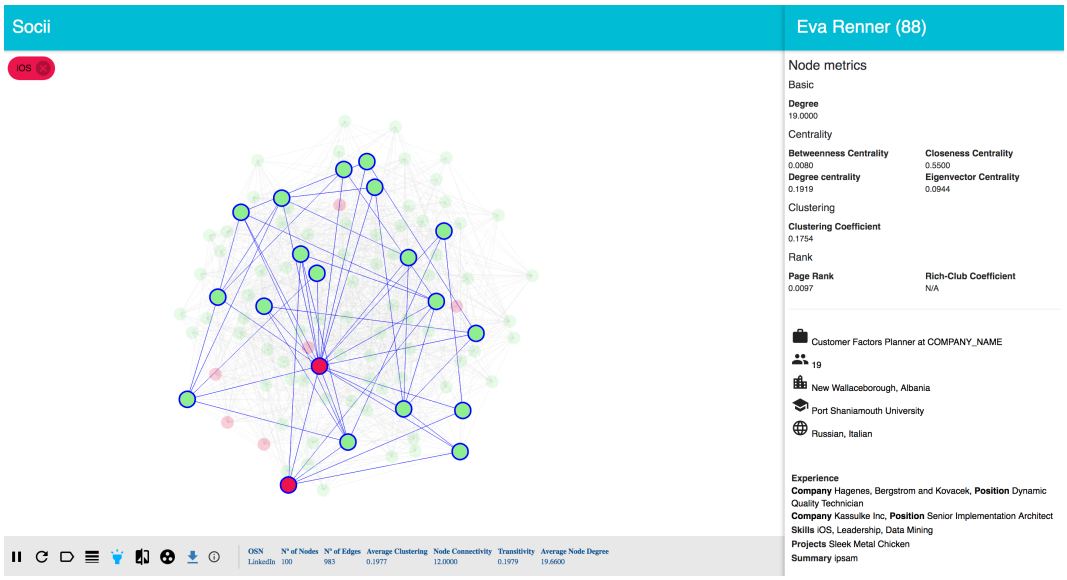


Figure 28: LinkedIn community detection by professional skill.

By picking the community detection in a LinkedIn network we are prompted with a very similar pop up configuration menu to Facebook networks. Here we can pick a particular skill (professional competence) and a color to paint all the nodes in the network that pos-

sess that particular skill.

As we may observe in Figure 28 we may now apply some of Socii previous demonstrated features to choose a particular individual for a suitable position based on either his LinkedIn profile information and/or some OSN metrics also offered by Socii. We then may use node comparison to compare a candidate to another one by some other parameters.

9

CONCLUSION

In this Chapter we look at our work retrospectively and we discuss the outcoming contribution.

At the very start of our work we had limited expectations since the definition of the final product was redefined along the way. We first started studying how SNs came to exist and how they were initially perceived (Chapter 2). After realizing the time and effort that sociologists had already invested in this subject, we started investigating the OSNs that were described as the manifestation of SNs of our epoch, the Internet era. The most relevant OSNs were deeply analyzed in Chapter 3; we looked on how they are composed and what drives users to use them.

Then we needed to know how these social structures are studied and interpreted from a scientific perspective; this led us to investigate the work already done in the field of SNA (Chapter 4). SNA provides the background to unable us to map social structures in mathematic abstractions. The first step in this analytical process always consists in representing the network by means of a graph. From there on some well established metrics such as *centrality* or *clustering coefficient* shall be evaluated, depending in what we want to perceive. These well established metrics help us to discover a series of facts about a network, such as *how influent are individuals*, *how many communities there are*, is the network dense, as well as all other analysis issues described in Chapter 4.

The background provided by that survey on SNs and their formal analysis paved the way to design our solution, Socii, in order to define a useful tool that would help users understanding their social structures based on the analysis of the identified OSNs. In Chapter 5 we discussed our proposal at the most conceptual way discussing its architecture in terms of a block diagram. In Chapter 6 we defined the requirements and their respective priorities in order to obtain a minimum viable product at the end of the project.

At the same time that we were developing our tool we documented all the relevant technicalities in Chapter 7. Finally, having Socii been implemented and tested, we reported all the attained results in Chapter 8. This Chapter contains a *walkthrough* of the functionalities of our tool; it also includes case studies that demonstrate how the end user can take profit

of Socii to obtain concrete results. With this we prove the utility of our final product, with the following main features:

- **Configurable/Parameterized analysis** - we offer our users the possibility to parameterize what metrics they want to calculate upon a given network, this configuration step is transversal to the underlying OSNs that the user wants to analyze.
- **Clear and intuitive social graph vizualization and interaction** - we built a specialized visual web component that is flexible enough to provide the user a set of visual features such as node coloring, node discovery, node dragging, node labelling etc.
- **Organized overview upon SNAs and OSNs data** - we implemented visual components that aggregate both SNAs metrics and OSNs information giving the user the opportunity to cross information from both worlds and derive conclusions from intersecting that information. We also integrated features that allow users to compose specific visualization scenarios such as coloring nodes by some common OSNs property.

9.1 THE MAIN OBSTACLE FOR SOCII

As we have explained along the dissertation, since the beginning we based our work on OSNs, developing a platform that is data driven meaning that it is built on the assumption of available and accessible data, however in reality this is not happening. Actually the OSNs we identified and described in Chapter 3 are not *"opening the doors"* to the community providing powerful APIs in order to make social public data available. That is why we went through the technical and architectural struggle of feeding in Socii networks through a extraction pipeline built on top of web crawlers, that are known and probed to be very slow and error prone. If OSNs such as Facebook or LinkedIn provided access to their social information via user friendly and opened APIs, Socii final results could be much more positive and surprising.

9.2 ALTERNATIVE TECHNICAL APPROACHES THAT COULD IMPROVE SOCII

In this section we will explore alternative approaches that can be implemented in order to improve certain bottlenecks of Socii such as performance. We will list these alternatives explaining both what Socii could gain and loose by selecting those paths.

9.2.1 *Visualization*

Using WebGL for network visualization and interaction

Web GL (Marrin, 2011) shall be the best option to build Socii if instead of a two dimensional network representation we choose to go on to the third dimension. This would resolve the node overlapping problem and could make the network discovery task a simpler process, since nodes would have more space to rearrange themselves. Open source projects such as *Graphosaurus* [1] would be helpful on this implementation, since it offers *"out of the box"* tools for developers to visualize three dimensional graphs.

9.2.2 *Performance*

Using server side rendering

Server side rendering is a technique where visual components (templating work) is done in the server side. This normally brings to web applications improvements in terms of the time spent in rendering and building templates, work that is usually done by the client according to architectural definition of more recent front end frameworks and libraries.

Server side rendering, in our specific case, could be a good approach since all the heavy calculations for positioning nodes may be done on the server instead of being done on the browser. This would however have impacts in terms of scalability if we had too many users requesting the rendering of huge networks.

Using web workers for heavy front end background processing

Modern browsers are close to fully support all the HTML5 new features, this including web workers (Hickson, 2017). These new technologies allow the browser to run a script operation in background thread separate from the main execution thread of a web application (Mozzilla, 2017). For Socii it would be very helpful to have some place where to run calculations as asynchronous tasks. This could be used for example, to metrics calculations instead of the current approach where we need to make an http request to the metrics microservice in order to fetch network metrics.

1 A three-dimensional static graph viewer: https://www.npmjs.com/package/graphosaurus

9.3 SOCII USAGE AND APPLICATIONS

We have already described some case studies in Chapter 8 where we demonstrated some of the potential uses of our tool. In this section we will meditate and speculate upon Socii potential of usage across several fields of study. So what could be Socii real applications?

- **Sociology general studies, social analysis** - Basically where Socii is merely a SNA tool used by scientists and students of the field.
- **Migratory flux of population** - Having a tool such as Socii that allows us to have a macroscopic overview upon social networks we could study population migratory flux (using community detection for example) to understand what is the shape and trends of population migration across the globe. At the time of this writing this could be helpful for example on the detection of refugees communities where we could find what communities were formed with existing and stable communities of other countries and how this affects both refugees and the local population.
- **Society happiness studies/Depression detection among youngsters** - We could use Socii to detect cases of depression among youngsters. This is today unfortunately a very common disease that urges among young people and that could be prevented by monitoring social networks usage among these youngsters and being alter for strange/abnormal behaviors.
- **Terrorism awareness/detection** - Using a similar strategy to the one we used to detect refugees communities we could analyze data and look for strange patterns of interactions concerning individuals origins. A simple example could be an individual with nationality X that belongs to a normal network and suddenly starts to create online connections with individuals of the nationality Y. Being this nationality blacklisted as possible association with terrorism, we could see this as a potential threat.
- **Marketing** - As discussed in Chapter 8 one of the use cases was marketing. Actually we could use Socii to detect potential target audiences for a given brand, product or service.

9.4 FUTURE WORK

In Chapter 6 we already described a lot of improvements that can be done regarding Socii evolution: the implementation of all the requirements that were not marked as **MUST** requirements are upgrades to the Socii tool that we see as relevant future work. Other ideas to move this project forward are:

- Improve network extraction process and allow users to build their networks on the fly;
- Adapt the current approach on Socii that builds social structures based on individuals relations to do the same thing with terms/keywords building a network of co-related keywords within a restrict domain/theme;
- Migrate from using Socii web crawlers to consuming directly these APIs if eventually OSNs make their social APIs available. This would considerably increase user experience and allow us to fulfill the first item of the *future work list* that is to allow users to quickly build their networks.
- Understand better Socii positioning among the social analytics world and try to find new and innovative applications for this tool.

As an alternative study to the previous list we foreseen, from analyzing various OSNs in Chapter 3 we have seen a possible research project on creating a framework for building and managing OSNs automatically and effortlessly. This framework could allow OSNs to be created on the fly with a model based approach, where the user/programmer just needs to insert a model and the OSNs would be generated.

BIBLIOGRAPHY

John Arundel Barnes. *Class and committees in a Norwegian island parish*. Plenum New York, 1954.

Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.

Vangie Beal. Webopedia definition for social network, 2016.

Kent C Berridge and Terry E Robinson. What is the role of dopamine in reward: hedonic impact, reward learning, or incentive salience? *Brain Research Reviews*, 28(3):309–369, 1998.

Stephen P Borgatti. Netdraw: Graph visualization software. *Harvard: Analytic Technologies*, 2002.

Michael Bostock. D3. js. *Data Driven Documents*, 492, 2012.

Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. Graphml progress report structural layer proposal. In *International Symposium on Graph Drawing*, pages 501–512. Springer, 2001.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

James Clark, Steve DeRose, et al. Xml path language (xpath) version 1.0, 1999.

Dai Clegg and Richard Barker. *Case method fast-track: a RAD approach*. Addison-Wesley Longman Publishing Co., Inc., 1994.

danielcaldas. react-d3-graph. *Interactive and configurable graphs with react and d3 effortlessly*, *https://danielcaldas.github.io/react-d3-graph/docs/index.html*, 2017.

Pinterest Developers. Pinterest developers page. https://developers.pinterest.com/, 2016. Online accessed 29 October 2016.

Cambridge Dictionary. Cambridge dictionaries online, 2002.

Selenium Documentation. Selenium webdriver. *Selenium HQ, Feb*, 2013.

Maeve Duggan. The Demographics of Social Media Users. http://www.pewinternet.org/2015/08/19/the-demographics-of-social-media-users/, 2015. Online accessed 29 October 2016.

Jessica Elgot. From relationships to revolutions: seven ways Facebook has changed the world. https://www.theguardian.com/technology/2015/aug/28/from-relationships-to-revolutions-seven-ways-facebook-has-changed-the-world, 2015. Online accessed 29 October 2016.

Nicole B Ellison et al. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

Facebook. Project palantir. https://www.facebook.com/video/video.php?v=37403547074&ref=nf, 2008. Online accessed 5 December 2016.

Facebook. React. *A Javascript library for building User Interfaces*, 2017.

Alejandra Guzman Farida Vis. 6 ways social media is changing the world. https://www.weforum.org/agenda/2016/04/6-ways-social-media-is-changing-the-world/, 2016. Online accessed 29 October 2016.

Andrew T Fiore and Judith S Donath. Homophily in online dating: when do you like someone like yourself? In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, pages 1371–1374. ACM, 2005.

The Guardian. Linkedin bought by microsoft for $26.2bn in cash. https://www.theguardian.com/technology/2016/jun/13/linkedin-bought-by-microsoft-for-262bn-in-cash, 2016. Online accessed 22 October 2016.

Aric Hagberg, Dan Schult, Pieter Swart, D Conway, L Séguin-Charbonneau, C Ellison, B Edwards, and J Torrents. Networkx. high productivity software for complex networks. *Webová strá nka https://networkx. lanl. gov/wiki*, 2013.

Jeffrey Heer and Danah Boyd. Vizster: Visualizing online social networks. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 32–39. IEEE, 2005.

Ian Hickson. webworkers. *W3C Working Draft 24 September 2015 https://www.w3.org/TR/workers/*, 2017.

Ariya Hidayat. Phantomjs: Headless webkit with javascript api. *WSEAS Transactions on Communications*, 2013.

MongoDB Home page. Mongodb. *NoSQL Database [online]*, 2009.

NodeJS Foundation Home page. Nodejs. *Node.js JavaScript runtime*, 2017.

Dimitris V. Kalamaras. Socnetv. http://socnetv.org/, 2004.

Martin Kilduff and Wenpin Tsai. *Social networks and organizations*. Sage, 2003.

Jérôme Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350. ACM, 2013.

Fuchun Lin, Yan Zhou, Yasong Du, Lindi Qin, Zhimin Zhao, Jianrong Xu, and Hao Lei. Abnormal white matter integrity in adolescents with internet addiction disorder: a tract-based spatial statistics study. *PloS one*, 7(1):e30253, 2012.

Steve Borgatti Lin Freeman, Bruce MacEvoy. Ucinet software. https://sites.google.com/site/ucinetsoftware/home, 2002.

Marak. faker.js. *Generate massive amounts of fake data in the browser and node.js*, 2014.

Marktest. Os portugueses e as redes sociais. http://www.marktest.com/wap/private/images/Logos/Folheto_redes_sociais_2016.pdf, 2016.

Chris Marrin. Webgl specification. *Khronos WebGL Working Group*, 2011.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

Mozzilla. mdnwebworkers. *Web Worker definition Mozilla https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API*, 2017.

Eyal Ophir, Clifford Nass, and Anthony D Wagner. Cognitive control in media multitaskers. *Proceedings of the National Academy of Sciences*, 106(37):15583–15587, 2009.

Pinterest. Pinterest about page. https://about.pinterest.com/en, 2016. Online accessed 29 October 2016.

Stanford Pritchard Lab. Structure software. http://pritchardlab.stanford.edu/structure.html, 2000. Online accessed 5 December 2016.

Aaron Retica. Homophily. http://www.nytimes.com/2006/12/10/magazine/10Section2a.t-4.html, 2006. Online accessed 5 November 2016.

Armin Ronacher. Flask (a python microframework), 2015.

Jeffrey Travers and Stanley Milgram. The small world problem. *Phychology Today*, 1:61–67, 1967.

Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.