Multiple identities detection in online social media

Zaher Rabah Yamak

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de l'INSA ROUEN NORMANDIE

Multiple Identities Detection In Online Social Media

Présentée et soutenue par
Zaher Rabah YAMAK

Thèse soutenue publiquement le 12/02/2018 devant le jury composé de		
Mme. Christine LARGERON	Professeur des Universités / Université Jean Monnet, Saint-Etienne, France	Rapporteure
Mme. Zahia GUESSOUM	Maitre de conférences HDR / Université Pierre-et-Marie-Curie, Paris-VI, Paris, France	Rapporteure
M. Cyrille BERTELLE	Professeur des Universités / Université du Havre, Normandie, Le Havre, France	Examinateur
M. Babak ESFANDIARI	Professor / Carleton University, Ottawa, Canada	Examinateur
M. Laurent VERCOUTER	Professeur des Universités / INSA Rouen Normandie, Rouen, France	Directeur de thèse
M. Julien SAUNIER	Maitre de conférences / INSA Rouen Normandie, Rouen, France	Co-encadrant de thèse

Thèse dirigée par Laurent VERCOUTER et co-encadrée par Julien SAUNIER, laboratoire LITIS

INSA ROUEN NORMANDIE

MIIS

litis

Abstract

Since 2004, online social medias have grown hugely. This fast development had interesting effects to increase the connection and information exchange between users, but some negative effects also appeared, including fake accounts number growing day after day.

Sockpuppets are multiple fake accounts created by a same user. They are the source of several types of manipulation such as those created to praise, defend or support a person or an organization, or to manipulate public opinion.

In this thesis, we present *SocksCatch*, a complete process to detect and group sockpuppets, which is composed of three main phases: the first phase objective is the process preparation and data pre-processing; the second phase objective is the detection of the sockpuppet accounts using machine learning algorithms; the third phase objective is the grouping of sockpuppet accounts created by a same user using community detection algorithms. These phases are declined in three stages: a model stage to represent online social medias, where we propose a general model of social media dedicated to the detection and grouping of sockpuppets; an adaptation stage to adjust the process to a particular social media, where we instantiate and evaluate the *SocksCatch* model on a selected social media; and a real-time stage to detect and group the sockpuppets online, where *SocksCatch* is deployed online on a selected social media.

Experiments have been performed on the adaptation stage using real data crawled from English Wikipedia. In order to find the best machine learning algorithm for sockpuppet's detection phase, the results of six machine learning algorithms are compared. In addition, they are compared with the literature, and the results show that our proposition improves the accuracy of the detection of sockpuppets. Furthermore, the results of five community detection algorithms are compared for sockpuppet's grouping phase, in order to find the best community detecton algorithm that will be used in real-time stage.

RÉSUMÉ

Depuis 2004, les médias sociaux en ligne ont connu une croissance considérable. Ce développement rapide a eu des effets intéressants pour augmenter la connexion et l'échange d'informations entre les utilisateurs, mais certains effets négatifs sont également apparus, dont le nombre de faux comptes grandissant jour après jour.

Les *sockpuppets* sont les multiples faux comptes créés par un même utilisateur. Ils sont à l'origine de plusieurs types de manipulations comme la création de faux comptes pour louer, défendre ou soutenir une personne ou une organisation, ou pour manipuler l'opinion publique.

Dans cette thèse, nous présentons *SocksCatch*, un processus complet de détection et de groupage des *sockpuppets* composé de trois phases principales : la première phase a pour objectif la préparation du processus et le pré-traitement des données ; la seconde phase a pour objectif la détection des comptes *sockpuppets* à l'aide d'algorithmes d'apprentissage automatique ; la troisième phase a pour objectif le regroupement des comptes *sockpuppets* créés par un même utilisateur à l'aide d'algorithmes de détection de communautés. Ces phases sont déclinées en trois étapes : une étape "modèle" pour représenter les médias sociaux en ligne, où nous proposons un modèle général de médias sociaux dédié à la détection et au regroupement des *sockpuppets* ; une étape d'adaptation pour ajuster le processus à un média social spécifique, où nous instancions et évaluons le modèle *SocksCatch* sur un média social sélectionné ; et une étape en temps réel pour détecter et grouper les *sockpuppets* en ligne, où *SocksCatch* est déployé en ligne sur un média social sélectionné.

Des expérimentations ont été réalisées sur l'étape d'adaptation en utilisant des données réelles extraites de Wikipédia anglais. Afin de trouver le meilleur algorithme d'apprentissage automatique pour la phase de détection de sockpuppet, les résultats de six algorithmes d'apprentissage automatique sont comparés. En outre, ils sont comparés à la littérature où les résultats de la comparaison montrent que notre proposition améliore la précision de la détection des sockpuppets. De plus, les résultats de cinq algorithmes de détection de communauté sont comparés pour la phase de regroupement de Sockpuppet, afin de trouver le meilleur algorithme de détection de communauté qui sera utilisé en temps réel.

A ma famille,
A Vous,
Mon père et ma mère,
pour votre support, affection, confiance et vos prières.

Remerciements

Je souhaite remercier en premier lieu mon directeur de thèse, M. Laurent Vercouter, pour m'avoir accueilli au sein de son équipe. Je lui suis également reconnaissant pour le temps conséquent qu'il m'a accordé, ses qualités pédagogiques et scientifiques, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela.

J'adresse de chaleureux remerciements à mon co-encadrant de thèse, M. Julien Saunier, pour son attention de tout instant sur mes travaux, pour ses conseils avisés et son écoute qui ont été prépondérants pour la bonne réussite de cette thèse. Son énergie et sa confiance ont été des éléments moteurs pour moi. J'ai pris un grand plaisir à travailler avec lui.

Je voudrais remercier les rapporteures de cette thèse Mme. Christine Largeron, et Mme. Zahia Guessoum pour l'intérêt qu'elles ont porté à mon travail. J'associe à ces remerciements M. Cyrille Bertelle, et M. Babak Esfandiari, pour avoir accepté d'examiner mon travail.

Ces remerciements seraient incomplets si je n'en adressais pas à mes collègues qui sont devenus mes amis. Je les remercie pour les moments agréables qu'on a partagés ensemble, pour les discussions scientifiques et non scientifiques. Vous êtes nombreux, je vais essayer de vous citer mais excusez moi si j'ai oublié quelqu'un. Merci Ahmad, Imad, Fadila, Guillaume, Jean-Baptiste, Mamdouh, Moukesh, Sahba, Salah, et Zina. Je n'oublie pas de remercier les secrétaires du laboratoire Brigitte et Sandra qui étaient toujours prêtes à m'aider dans les tâches administratives.

J'adresse aussi mes remerciements à ma deuxième famille en France, Bachar et Faiza pour leur accueil chaleureux et leur soutien depuis mon arrivée en France.

Je remercie ma chère épouse Joumana pour son soutien quotidien indéfectible et son enthousiasme contagieux à l'égard de mes travaux comme de la vie en général. Notre famille a grandi en même temps que mon projet scientifique. Au début de ma deuxième année du thèse, on a eu nôtre meilleur cadeau au monde, mon cher Nassim qui a aujourd'hui 2 ans et 4 mois. C'est grâce à lui que j'oublie ma fatigue et mon stress du travail.

Ces remerciements ne peuvent s'achever, sans penser à mon oncle Nasser et sa famille qui m'ont soutenu beaucoup durant la durée du thèse. Je tiens également à remercier mon beau père Ammar et ma belle mère Jinan à m'encourager à finir cette thèse avec perfection.

Finalement, je tiens à remercier mes premières fans : mon père Rabah, ma mère Lina et mes frères Wissam et Jihad. Leurs présences et leurs encouragements sont pour moi les piliers fondateurs de ce que je suis et de ce que je fais.

07 Février 2018
Zaher

Table of Contents

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Context and Motivation

1.1.1 Social medias

Since 2004, the social web has become a dominant force on the Internet. As of 2017, there are 2.78 billion active social media accounts [Simon, 2017] and each internet user has an average of 7 social media accounts [Alex, 2017]. For instance, in 2017, Facebook members represent 79% of all Internet users [Mary, 2017], and there are nearly 1.9 billion unique monthly Facebook users [Andrew, 2017].

Over the years, social media have evolved hugely, putting part of the users personal lives to virtual spaces. However, this evolution also has negative effects. According to the U.S. Bureau of Justice Statistics, an estimated 16.6 million Americans were victims of identity theft in 2012. Financial losses for these victims added up to $24.7 billion. For identity thieves, social media is a hunting ground for targets' personal

information [Ambika Choudhary, 2014].

Through APIs like 'Sign in with Twitter', users can register and connect to third party websites using their twitter account. The simplicity of the process has enabled these websites to benefit from the access to personal information of a large number of accounts. However, as the importance of social media grows, the number of manipulators increases. As a major power, social medias are particularly targeted by a series of attacks, ranging from social engineering [Bisson, 2016] to cyber attacks [Goolsby et al., 2013].

Nowadays, traditional media are no longer the only source of news because major news stories routinely break on Twitter before traditional news media [Mathew, 2012]. Online Social Media (OSM) have therefore become a preferential media for diffusion of information (e.g. mass birthday celebration) or opinions (e.g. on a product or on a public person) and to promote ideas (i.e. activism, call for vandalism or riots). In the case of events that can lead to civil security or homeland security issues, reactivity in detecting these messages, in identifying originators, "barkers", and followers as well as the processes of information diffusion is needed to prevent acts or events before they occur. Similarly, for governments, public personalities and companies, the control of information diffusion has become a major issue.

1.1.2 Manipulation in social medias

A manipulator is a person who uses the rules of life in society, in this case those of the social medias, to obtain advantages and personal benefits or exercise control over one or more users. Manipulation on social media can be made through verbal communication (e.g. video, audio, or text) and/or non-verbal behavior (eg: number of friend requests per hour, or the size/quality of added text. . .) using many techniques like spams or Sybil

attacks [Douceur, 2002].

A frequent manipulation technique uses multiple identities by creating sockpuppets; accounts created and controlled by the same individual for the same deception objective. Sockpuppets are used to perfom various malicious activities such as manipulating presidential election [Jordan et al., 2016], influencing financial market [John, 2017], disseminating spams, phishing URLs, and malware [Thomas et al., 2011a] as well as harvesting private user data [Bilge et al., 2009]. Another problem is that sockpuppets have the ability to influence other users and impose attitudes, behaviors and opinions on them. That makes them a relevant source of communication for organizations that want to promote a brand, product, service, or idea [Patel et al., 2017]. Therefore, the detection of sockpuppets in OSMs is a urgent research problem.

Most works on manipulation detection address individual malicious accounts, but manipulators often act in pairs or larger groups [Vrij et al., 2010]. Due to the difficulty of obtaining ground-truth data about sockpuppets, some researches are built on assumptions about the behavior of sockpuppets, for example, assuming that they have similar usernames [Liu et al., 2016], that they are only used to support one another [Zheng et al., 2011b], or that they write sentences similar to each other [Bu et al., 2013]. Further research examines pairs of sockpuppets controlled by the same sockpuppeteer, and find that sockpuppets are more likely to post at the same time and post in the same discussion than random pairs of ordinary users. In addition, they find that pairs of sockpuppets write more similarly to each other than to ordinary users [Kumar et al., 2017].

OSMs are interested in finding all the accounts corresponding to a single individual inside a single social media [Goga, 2014] in order to protect their platforms. Users are supposed to open only one account in a social media (as stipulated in the Terms of

Service), however some users create multiple accounts. Furthermore, malicious users can impersonate honest users. For both cases, we need grouping techniques to find the accounts of a single individual.

Research questions

To resolve the problem of multiple identities in OSMs, our objective is to detect the sockpuppets and group them with their sockpuppeteer. This implies to study the behaviors of OSM's users, in order to (1) find the behavioral differences between sockpuppets and legitimate accounts and (2) identify the similarities between the behaviors of the sockpuppets created by the same manipulator.

Therefore, the following research questions are raised:

Q1: How to differentiate between the behavior of sockpuppets and legitimate accounts ?

Q2: Can sockpuppets be automatically detected ?

Q3: How to relate together sockpuppets?

Q4: Can the sockpuppets that correspond to a single malicious user be grouped together?

Q5: What are the common characteristics between different types of online social media that can be exploited to propose a generic model for sockpuppets detection?

Q6: How to adapt the solution to a specific online social media?

1.1.3 Contribution

In order to find answers to these questions, a process is proposed in this thesis to detect and group the sockpuppets created by a same manipulator.

The main contributions of this thesis are:

C1: A global process to detect and group the sockpuppet accounts: Sockscatch;

C2: A set of features to differentiate between the behavior of sockpuppet and legitimate accounts;

C3: A set of graphs to relate and group sockpuppets that have close behavior using community detection algorithms;

C4: An adaptation process to instantiate the features and select the best algorithms for a particular OSM;

C5: An evaluation of Sockscatch on English Wikipedia.

1.2 Organization

This dissertation is organized as follows.

Chapter 2 - Manipulations in Social Media Chapter 2 introduces a brief history of social medias. Then, it shows the classification of social medias with a brief description of each type. Furthermore, it develops the main social attacks and finally it presents the protection systems of different social medias.

Chapter 3 - Malicious Behavior: Detection and Grouping Chapter 3 presents the state of the art in the field of detection of malicious accounts and communities in online Social Media (OSM). It identifies and categorizes different criteria to detect and group manipulators. Then, it defines the basic concepts of machine learning used in this thesis. It continues by reviewing the solutions that detect a single fake account such as spam accounts and multiple identities accounts. Then, It presents community detection algorithms used to group multiple identities. Finally, it reviews the solutions to group malicious behavior accounts created for the same objective in order to achieve common attacks from different identities.

Chapter 4 - The SocksCatch Process Chapter 4 presents a global overview of the

SocksCatch process, that aims at detecting and grouping sockpuppets. First, it introduces the overall process of *SocksCatch*. Then it proposes a model to represent online social medias, in order to represent their basic elements. Finally, it presents our use case, english Wikipedia, and how to adapt the *SocksCatch* process to it.

Chapter 5 - Sockpuppet Detection Chapter 5 presents the phase of the detection of sockpuppets, where several features justified by our analysis of sockpuppet and legitimate account's behaviours on social medias are proposed. Then, it presents the instantiation of these features on English Wikipedia. Finally, it evaluates these features and determines the best machine learning algorithm to be used in real time on this OSN.

Chapter 6 - Grouping Sockpuppets Chapter 6 details the phase of grouping the sockpuppets created by the same manipulator. This phase comprises two steps: the grouping of sockpuppets based on the actions performed in the same pages then the grouping based on accounts' behavior. This chapter presents also how to adapt and evaluate this phase on English Wikipedia.

Chapter 7 - Conclusions and Perspectives Chapter 7 presents the conclusions of the thesis and exposes some perspectives.

CHAPTER 2

MANIPULATIONS IN SOCIAL MEDIA

Coined by Dale Dougherty in 2004, the term "Web 2.0" refers to a technological evolution that allows a set of new practices on the Internet. After this evolution, internet users can create and publish content; tasks initially reserved to Internet professionals. While Web 1.0 is the first step in the development of the World Wide Web and has structured Internet networks and communication, Web 2.0 marks the real awakening of Internet users, because they are at the heart of exchanges on the Web. Its main new features are sharing and collaboration. *User Generated Content* (UGC) is a concept that derives from these new practices. It is defined as the content created by users such as videos, audio files, photos, blogs, pod-casts, participation in forums, online product reviews [Lendevrie et al., 2010]. Thus, Web 2.0 is the ground conducive to the development of social medias.

Social medias have been the new trend of the 2000s. According to studies, teens now spend up to nine hours a day on social platforms, while 30% of all time spent online is now allocated to social media interaction. The majority of that time is on mobiles - 60% of social media time spent is facilitated by a mobile device [Evan, 2017].

Across today's most popular social media platforms, people are spending 1 hour and 56 minutes daily averages on these platforms where 40 minutes are spent on Youtube, 35 minutes on Facebook, 25 minutes on Snapchat, 15 minutes on Instagram and 1 minute on Twitter [Mediakix, 2016].

In a short time, social medias have become a true media of social interaction. They are based on the creation and the sharing of content and on the users' personal networks. In this document, we define social media according to the terms of [Kaplan and Haenlein, 2010]: "Social Media is a group of Internet-based applications that build on the ideological and technological foundation of Web 2.0 and that allow the creation and exchange of UGC".

The beginnings of social media go back to the late 1970s, even before the creation of the Web: at the time, the first online forums for communication were Bulletin Board Systems (BBS). They operated over phone lines and the system was incredibly slow. The conversation remained mostly local because long-distance rates did apply. But it is at the end of the 1990s that social media really began to develop and democratize, particularly in the United States. Online thematic communities appear to find old classmates, share cooking recipes, and so on. Sixdegrees.com, created in 1997, marks the true birth of Social networking services as they are known today. The main objective of the site was to allow its users to stay in touch with their friends, family members and more distant knowledge through the creation of profiles and the exchange between members of their personal network. After the site was closed in 2000, a series of social networking platforms was launched such as Friendster in 2002, an improved version of Sixdegrees.com promoting friendship. The site has been victim of its success: the proliferation of fake member profiles, known as "Fakesters", led to the abandonment of the site by thousands of

members. Myspace, created in 2003 to compete with Friendster, took advantage of this general disengagement by offering its members a customizable web space. It became the first American social network between 2006 and 2008, surpassing Google in terms of visits. LinkedIn is founded the same year but aims at the creation of professional online networks. 2004 is a key date in the history of social media since it marks the birth of Facebook: a social network aimed to the students of Harvard University in the United States. The site is rapidly gaining momentum and by 2006, anyone who is at least 13 years of age and has an e-mail address can register. With 1.86 billion monthly active members in 2017 [Zephoria, 2017], Facebook is nowadays the leader of social media.

As a major power, it is natural that social medias are now targeted by a series of attacks, ranging from social engineering [Bisson, 2016] to cyber attacks [Goolsby et al., 2013]. For instance, many Facebook accounts have fake profile data [Gao et al., 2010].

Using fake accounts, the deception, also called manipulation, on social medias is performed through many techniques, and it varies from a site to another. A fake account can be either be a hijacked profile with the name, address, and image of another person or an identity created from scratch. They are created by physical persons or robots with very often bad intentions. Beyond the classic wallet scams, one of the new missions of these manipulators is to harm the image of a company or a person, with for example negative opinions and comments. These fake accounts are constantly increasing, social medias and researchers are putting actions in place to protect the users.

In the next section, we present a classification of social medias with a brief description of each type, then we develop the main social attacks and finally we present the protection systems of different social medias.

2.1 Social Media Classification

Social medias are online platforms that feature the capability for users to collaborate in content creation (wikis), information exchange (forums, blogs, comments ...) or content sharing (articles, photos, videos, messages ...).

Social medias use collective intelligence to create online collaboration through interaction between individuals or groups of individuals. These individuals are thus led to create content, to prioritize it in order to share it with other individuals. This content is constantly evolving, linked to changes that other users can make (such as comments or collaborative working documents). The collaboration can take many forms where the user can:

- give an opinion on a subject, product, service or website;
- interact with other users;
- contribute to participatory projects;
- aggregate information for later sharing.

While the content and modes of interaction differ between OSM, they all feature a common grounding: They allow users to share information and interact, both in public and private ways.

In order to achieve this, they are usually based on accounts to uniquely identify users. These accounts have some common properties between all OSM such as usernames and the time of creation of each account.

[Kaplan and Haenlein, 2010] have proposed a classification for social media based on social presence/media richness and self-representation/self-disclosure which include six categories:

1. **Collaborative projects** : the user can add, edit, and remove text content in these websites (e.g.Wikipedia)

2. **Blogs** : the user can share information, content (text, audio, video), or write reviews in these websites (e.g., Twitter, TripAdvisor)

3. **Content communities** : the user can share content like text, audio, photos or video (e.g., YouTube, SoundCloud)

4. **Social Networking sites** : the user can create a profile with personal information and share contents with friends (e.g., Facebook)

5. **Virtual game worlds** : the user can create a virtual profile and play games with other players (e.g., World of Warcraft)

6. **Virtual social worlds** : the user can create a virtual profile and interact with other users as in real life in virtual worlds (e.g., Second Life) .

This classification is based on principles that come from media theory. The first factor is **social presence** theory which was imported from tele-conferencing research. One of the first analytic frameworks applied to computer-mediated communication (CMC) was the work of Short *et al.* in 1976 who argued that various communication media differed in their capacity to transmit classes of non-verbal communication in addition to verbal content [Short et al., 1976]. **Media richness** theory [Daft and Lengel, 1986], also known as information richness theory [Daft and Lengel, 1983], refers to the amount of information that can be transmitted in a given time. The others factors come from **self-representation** and **self-disclosure** theories. **Self-representation** theory states that people want to control their impression formation on others in social interactions [Erving,

1959]. However, **self-disclosure** theory points out the connection between higher quality relationships among people and the development of psychological well-being [Kaplan and Haenlein, 2010].

This first classification in six main types of social medias was expended in 2011 to include micro-blogging which stands "halfway between traditional blogs and social networking sites" [Kaplan and Haenlein, 2011] .

		Social presence / Media richness			
		Low	**Low-Medium**	**Medium**	**High**
Self presentation / Self disclosure	**High**	Blogs	Microblogging	Social networking sites	Virtual social worlds
	Low	Collaborative projects		Content communities	Virtual game worlds

Table 2.1 – Classification of social media typology.

Table 2.1 depicts the classification of social media [Kaplan and Haenlein, 2010, 2011] where the first row presents the social medias that provide users with a lot of freedom for presenting themselves while the second row presents the social medias that force users to adapt to certain roles or have no option for disclosing parts of their identities. Moreover, the social medias vary in social presence and media richness, such as collaborative projects that offer just text and sparse images for communication, while virtual social worlds are aimed to simulate the real world using all types of media.

There is a great amount of literature on social media for all of these classifications [Botha et al., 2011; Mangold and Faulds, 2009; Weinberg and Pehlivan, 2011]. A short description for each type of social medias is provided here for reference:

2.1.1 Blogs

A blog is a kind of online journal that allows the publication of diary-style text entries displayed in a chronological way, and dealing with various themes such as politics, music, cinema, literature, fashion... Any type of content can be published: texts, images, videos, audio clips. Appearing at the end of the 1990s, it is one of the first manifestations of social media. There are many platforms that facilitate the creation of a blog such as Skyblog in France, which popularized the blog among adolescents in the early 2000s.

In general, a single author is behind a blog but in 2010 "multi-author blogs" appeared so that several people can associate and contribute to the same blog. The comments posted by the Internet users allow a permanent interaction between the readers and the blogger. It can also bring together a true community as its audience grows, and even transform itself into a leader of opinion.

The blog is not reserved to a small group of experts but is open to anyone with an Internet connection and willing to express himself on any subject. Its use is simple and truly democratizes the publication of content. Figure 2.1 depicts an example of a blog.

2.1.2 Microblogging

Microblogging is a derivative of the blog, that allows its users to publish small elements of contents such as short sentences, individual images, or video links, to the attention of a circle of "Followers" of the stream. First called tumblelog, it takes the name of microblog around 2006.The functionalities of this type of platform remain relatively basic, the relational aspect is privileged and the community dimension reduced to its simplest expression. Among the most well-known microblogging sites are Twitter, which offers the user the possibility of posting "Tweets", that are short messages limited to 140

Figure 2.1 – The f word's blog.

characters, and Tumblr, closer to blogs, that allow to publish text, photos, videos, etc.

Figure 2.2 depicts an example of microblogging.

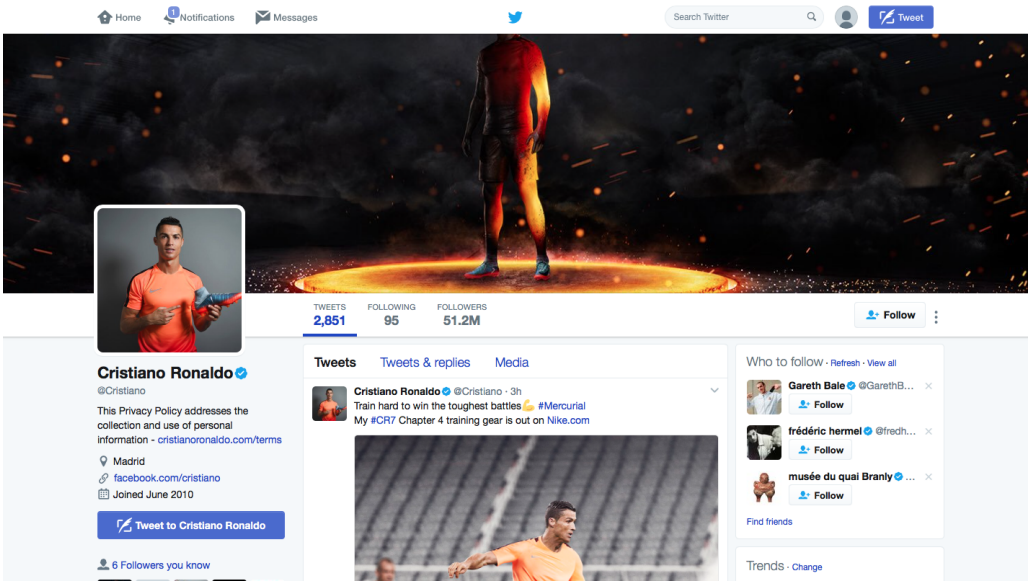

Figure 2.2 – Cristiano Ronaldo Twitter's page.

2.1.3 Collaborative Projects

Collaborative projects are one of the most important social medias in terms of audience and contributions. For example, Wikipedia was visited in April 2015 by 104.40 million of unique US visitors [Statista, 2015].

Collaborative projects also feature unique characteristics in OSM: they can be edited by multiple approved people, all edits are tracked in the page's history, any bad edition can easily be reverted. Examples of such sites are Wikipedia, WikiHow, and Distributed Wikis [Davoust et al., 2015].

The most known collaborative project is Wikipedia, a crowd-sourced encyclopedia, in which anyone can edit most of its articles either with or without creating an account on it.

The sets of pages on Wikipedia, called namespaces, are composed of pages with information or discussions about Wikipedia. Currently, Wikipedia has 34 namespaces [table 2.2]: 16 subject namespaces, 16 corresponding talk namespaces, and 2 virtual namespaces. Each subject namespace has a corresponding talk namespace, for example the main namespace that contains the articles has a namespace called Talk, and the User namespace that contains the user pages and other pages created for personal use also has a corresponding namespace called User Talk. The two virtual namespaces are not related to pages stored in the database. For example, the Media namespace is used to link directly to a file rather than to the file description page.

Figure 2.3 depicts an example of collaborative project.

Subject ID	**Subject namespaces**	**Talk namespaces**	**Talk ID**
0	(Main/Article)	Talk	1
2	User	User talk	3
4	Wikipedia	Wikipedia talk	5
6	File	File talk	7
8	MediaWiki	MediaWiki talk	9
10	Template	Template talk	11
12	Help	Help talk	13
14	Category	Category talk	15
100	Portal	Portal talk	101
108	Book	Book talk	109
118	Draft	Draft talk	119
446	Education Program	Education Program talk	447
710	TimedText	TimedText talk	711
828	Module	Module talk	829
2300	Gadget	Gadget talk	2301
2302	Gadget definition	Gadget definition talk	2303
Virtual namespaces			
-1	Special		
-2	Media		

Table 2.2 – Wikipedia namespaces.

2.1.4 Social Networking Sites

The most widely known and widely used social medias are social networks. These platforms allow contact between users through the creation of personal profiles that friends, family members and acquaintances can access. It also provides instant messaging and chat services. Social networks encourage the exchange of various content such as photos, videos, press articles, websites, opinions, statuses, etc. They promote communication by allowing members to keep in touch and fuel a sense of community.

There are several types of social networks. Some remain fairly general such as Facebook, the first social network. In 2016 only, Facebook increased its global footprint

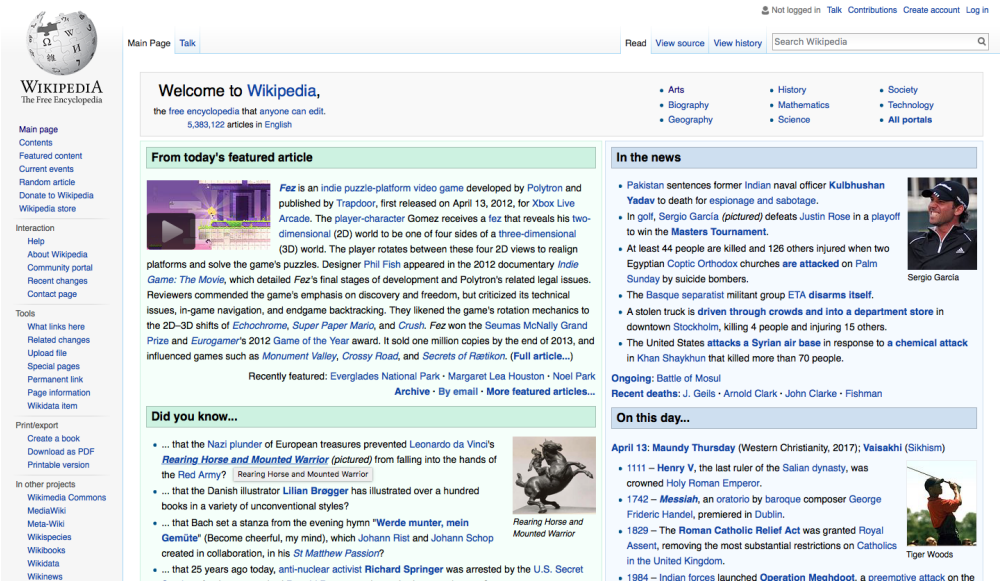

Figure 2.3 – Wikipedia's encyclopedia home page.

by 363 million monthly active accounts, and deliver year-on-year growth of 24%. More than one billion persons use Facebook every day, meaning that more than half of all active Facebook users use the service on a daily basis [Simon, 2017].

Figure 2.4 – Mark Zuckerberg's Facebook page.

Other social networks have a professional scope and allow users to create their Curriculum Vitae online to make it available to thousands of recruiters. This is the case of LinkedIn and Viadeo. Finally, thematic social networks are oriented towards a particular community: Spotify's vocation is the diffusion of music, Buds Forward allows former classmates to meet, and so on. There are even social networks whose access is strictly limited to a privileged community and only through invitation such as *A Small World*.

Figure 2.4 depicts an example of a social network site.

2.1.5 Content Communities

Content Communities are web hosting sites on which users can send, watch and share medias.

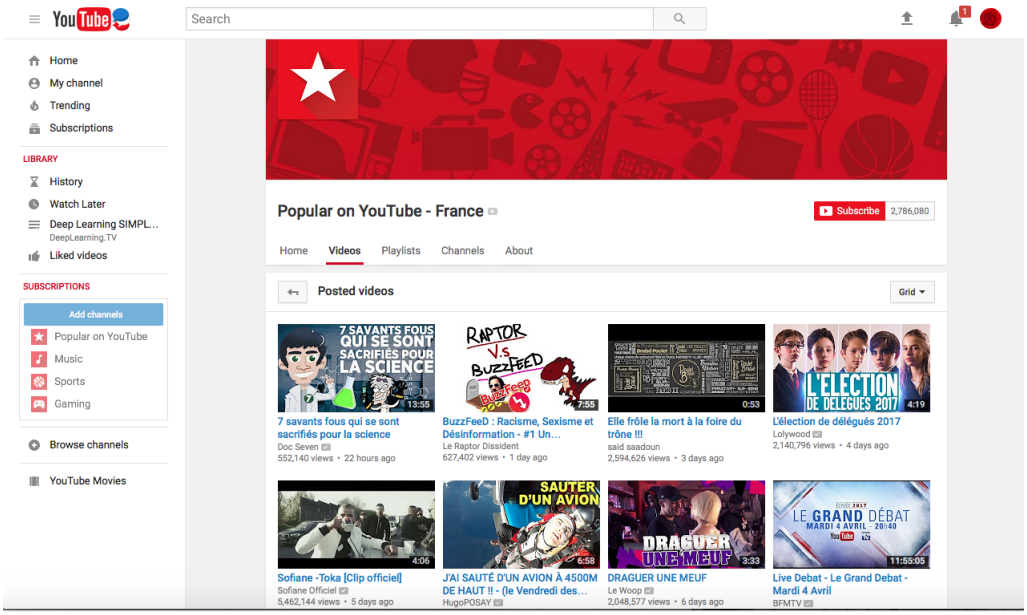

Figure 2.5 – Popular on YouTube - France, YouTube's channel.

Popular content communities include YouTube, Flickr, and SlideShare. After registering an account, individuals upload their materials, describe them, and make them

publicly available. Visitors search the content communities by keyword, subscribe to individual users, and provide feedback on the content.

Youtube can be considered as a social network but it is focused on sharing videos of all kinds. Freshly released video clips reach hundreds of millions of views, from guitar tutorials to broadcasting a large number of videos.

Figure 2.5 depicts an example of content communities.

2.1.6 Virtual Social Worlds

Virtual social worlds are platforms that replicate a world that is fantastic or close to real life and allow players to interact through an avatar. They are comparable to video games that require an Internet connection. Players also have instant chat and discussion forums in order to make a strong sense of community belonging among their members.

Virtual social worlds' do not impose restrictions on how to create, behave or interact, but they are economies in their own right.

With regard to economic activities, in some virtual social worlds the avatars have the possibility of offering products and services which they themselves created (*i.e.* not generated by the owner of the world) and may sell it to other users in exchange for virtual money. Second Life is the most popular virtual social world site. Its virtual currency is called the Linden dollar. Avatars can buy Linden dollars for real money via the SL Exchange at a floating exchange rate.

Figure 2.6 depicts an example of virtual social worlds.

Figure 2.6 – Second Life World.

2.1.7 Virtual Game Worlds

In opposition to virtual social worlds, in virtual game worlds, users are usually required to follow strict rules that determine their behavior and the personalization of their avatar. In addition, virtual game worlds sites do not allow anyone to engage directly into economic activities with other users.

In this category are included platforms "that replicate a three-dimensional environment in which users can appear in the form of personalized avatars and interact with each other as they would in real life" [Kaplan and Haenlein, 2010]. A known example of virtual game world is "World of Warcraft" with more than 10 million subscribers at its peaks on December 2014 [Blizzard, 2014], according to Blizzard entertainment.

Figure 2.7 depicts an example of virtual game worlds.

Social medias are growing day after day and they are gathering the most important number of visitors on the internet. Attracted by this success, among the visitors there is also a number of manipulators that try to benefit from the others, which leads to an

Figure 2.7 – World of Warcraft game.

increase of the number of social attacks. In the next section we present the different social attacks existing on social medias.

2.2 Social Attacks

Social attacks are the techniques that aim to broadcast false information in order to manipulate the opinions, or to access confidential information or certain assets by manipulating people who have access to it directly or indirectly. They are applied to the field of informatics via telephone, e-mail and social medias.

The human factor is the central point of these attacks using social engineering. In essence it is about the intelligent manipulation of human natural propensity to trust. Relationships of trust based on no real interaction history are set up in a calculated way, most often by simple discussion and subsequently exploited to derive maximum benefit from the situation.

Social attacks vary according to the type of social media, for example, Facebook users

are subject to social spam, taking advantage of sharing and tagging mechanisms from a big number of users on their spam posts, while linkedIn users are more subject to spear phishing because of the mixed professional/personal nature of the shared information[1].

We use the comprehensive communication model proposed by Madhusudan in 2003 to classify manipulation techniques for social media and to evaluate their effectiveness in carrying out manipulation [Madhusudan, 2003].

2.2.1 Manipulation Model

The model (shown in figure. 2.8) consists of a sender (S), the content or message (I), the channel through which the communication takes place (C), and the receiver (R). If the received model is different from the expected model, then a manipulation has happened. The objective of a manipulation is to give information and / or modify R's behavior. Hence, the manipulation is achieved if one of the S, I, C elements is manipulated. [Tsikerdekis and Zeadally, 2014b] apply the model of [Madhusudan, 2003] to the different types of social media. They show which elements of the model can be manipulated with minimal (i.e. low cost) efforts and at the same times with a fairly high success rate (shown in table 2.3). We develop in the next sections the characteristics and vulnerabilities of each element with regard to social medias types.

2.2.1.1 Content (I) Manipulation

Manipulating the content or the message is the most frequent way of deceiving others. This can be achieved in social media by falsifying information, especially for social media that mainly focus on content like blogs, content communities and microblogging, because

[1] https://us.norton.com/spear-phishing-scam-not-sport/article

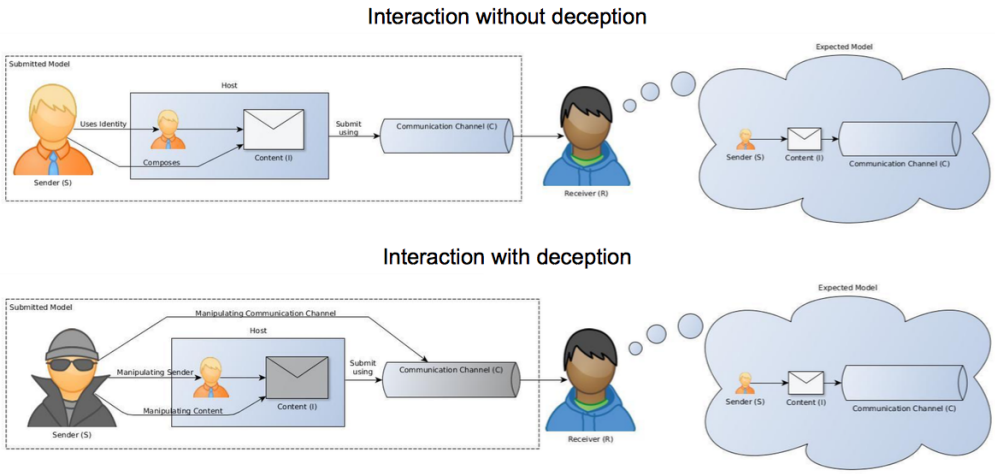

Figure 2.8 – Interaction without/with deception.

Social Media Type	Low difficulty	High deception success
Blogs	S, I	S, I
Collaborative projects	I	-
Microblogging	S, I	S, I
Social networking sites	S, I, C	S, I, C
Content communities	I	I
Virtual social worlds	S, I, C	S, I, C
Virtual game worlds	I, C	C

Table 2.3 – Manipulation of Sender's identity information (S), Content (I), and Communication channel (C) with low difficulty and high deception success.

they are very sensitive to such manipulation due to their structure based on contents added by users.

However, table 2.3 shows that projects like Wikipedia are less likely to be affected by this kind of manipulation (i.e. manipulate I). This is because the content of these types of social media is built using the community of contributors, thus allowing many people to review the content. Virtual game worlds also have a low probability for deception success because of the strongly narrated elements such as having a specific role that forces a player to a specific line of actions.

2.2.1.2 Sender (S) Manipulation

Sender manipulation is achieved by manipulating the sender's identity information (S). Impersonation is a common way to support identity deception. The manipulator tries to access an account to obtain all its information such as age, address, e-mail and cell number. The manipulator then tries to authenticate itself in place of its victims. The success of this operation means that the manipulation is successfully done.

Social medias designs that feature in-built high self-presentation and self-disclosure enable sender deception at a low cost. In the same way, new users are not controlled in most blogs and microblogging sites, so that it may lead to stole identities. However, the damage caused by manipulators on these types of social media is low and on the long term the success of this deception is not guaranteed.

In the case of social networking sites and virtual social worlds, the cost of manipulation increases because the access to an individual's social network can lead to gaining people's trust in the network and getting information from them. The target in this case may not necessarily be the person whose identity is stolen but others within his social network.

Social media with low self-presentation and self-disclosure, such as collaborative projects, content communities and virtual game worlds, are likely to be more protected in terms of identity theft, because they are task-based. A user, who wants to obtain access to the impersonated identity's social network will have to perform just as well as the identity being impersonated in tasks and "act the part". The cost is likely to be high and the success of the deception low and short-term.

2.2.1.3 Communication Channel (C) Manipulation

Manipulating the communication channel is the most difficult because it requires a high level of technical competence, which increases the cost of handling. Attacking the communication channel means modifying in-transit messages, re-routing of traffic, eavesdropping, etc. Jamming communications have been used in virtual game worlds. Podhradsky *et al.* argued that multi-player games in consoles can be hacked in order to provide access to a user's Internet Protocol address. Once the intruder has access to the host, he/she can kick the player out and proceed with an identity theft manipulation. In this case the goal of a deceiver may not be to obtain information but to damage a victim's reputation [Podhradsky et al., 2013].

On social media the manipulator may introduce an instability or delays in a video or an audio in order to conceal his manipulation. On the other side, victims may consider the problem concerns the connection without making attention to the risk of attack.

Social medias with high media richness, such as virtual social worlds and virtual game worlds, are more subject to this kind of attacks because they tend to rely a lot on client software. In addition, social medias that rely on server applications, such as social networking sites and content communities, are safer and less prone to channel manipulation because exploits rely on vulnerabilities of web browsers and web servers are generally hardened and made more secure.

This manipulation has a quite high cost. However, the likelihood of success is also high especially for well-orchestrated attacks.

2.2.2 Online Manipulation

There are different techniques that can be used to manipulate others in social media environments. The main techniques are:

1. Social spam.
2. Identity theft.
3. Spear phishing.
4. Multiple fake accounts (Social sibyls / Sockpuppets).

2.2.2.1 Social spam

Spams are classically irrelevant or unsolicited messages sent over the Internet. These are usually sent to a large number of users for a variety of use cases such as advertising, spreading malware, etc.

While the most widely recognized form of spam is email spam, the term is applied to similar abuses in social medias.

For several years, social medias have been developing with high success. Social spam, that is, social media spam, has increased dramatically in recent years. Today 40% of accounts and 8% of messages on networks are considered to be spam [Rip, 2011]. This percentage varies according to the networks.

The key of social spam is the fake accounts. In order to attack legitimate users they spread fake messages or malicious links created with the intent to harm, mislead or damage a user or their device.

Botnets now regularly steal credentials for social network accounts and use them to transmit spam on social networks [Fred, 2009; Mayur, 2009].

According to Impermium [Rip, 2011], spam via networks is much more efficient than spam called "classic" by e-mail. The networks therefore have to fight against this new threat by checking more seriously the identity of their users. This is particularly the case of Twitter, that is increasingly concerned by spam messages. False users use short links to direct the user to viruses and other threats.

2.2.2.2 Identity theft

Identity theft is the act of taking the identity of another living person without his knowledge and sometimes without even him noticing. However, for a large part of the population, identity theft remains a risk badly known.

The personal information is public by default on most social medias, which means that everyone can find sensitive information about user accounts. Names, dates of birth and personal addresses can then be used to access bank accounts and credit cards, for example, by estimating the "secret questions" used for access in several online services. Real-world thieves have even started using geographic information in status updates, photo-tags, and "check-ins" to plan home burglaries when occupants are out of town [Eric, 2012].

Another example of Identity theft manipulation is to hack into the bank account and thus contract a loan for the victims, withdraw money or open phone lines to their name and never pay the bills. It can also generate false papers such as driver licenses, gray cards, identity cards or passports and insurance card. In 2014, 550 000 people declared themselves victims of identity theft [Kelli, 2015].

2.2.2.3 Spear Phishing

Phishing is a cybercrime in which a target or targets are contacted by email, telephone or text message by someone posing as a legitimate institution to lure individuals into providing sensitive data such as personally identifiable information, banking and credit card details, and passwords[2].

Spear phishing is an advanced form of phishing that takes up the mechanism by tailoring them to specific targets to increase their efficiency.

Spear phishing is opposed to traditional phishing, in the sense that it does not target thousands of Internet users at random, but only one - or a few - selected in order to obtain very precise information. This attack is aimed above all at particular profiles, which work for particular organizations (companies, NGOs, governments, etc.). If the objective is often to obtain confidential information about a professional sector, attackers also use personal information to trap their victims. They collect personal information about social media accounts about friends, family and colleagues.

Once manipulators have targeted a person (or a group of individuals), and have gathered a lot of personal information about them, they write and send a fake, highly personalized email, which makes the detection of the trap difficult and convince the user to visit a fake website that is under the control of the attacker. Millions of dollars have been lost because of this type of attack [Kim, 2011].

2.2.2.4 Social Sibyls

Sybil attack is an attack wherein a reputation system is subverted by forging identities in peer-to-peer networks [3].

[2] http://www.phishing.org/what-is-phishing

[3] $https://en.wikipedia.org/wiki/Sybil_attack$

Social Sibyls is the particular type of sybil attack that is performed on social medias: malicious users create multiple identities with the aim of increasing their own influence in the social medias.

Malicious activity on social medias is mainly based on fake Sybil accounts, which are created and controlled en-mass by attackers. Facebook estimated in 2012 that there are at least 83 million sibyls on their service, which represents nearly 9% of the user base [Technology, 2012]. In addition, the social Sibyls have become a marketable commodity by themselves. There is now a thriving black market for sibyls on social medias [Wang et al., 2012b]. Attackers can buy popularity using fake sybil accounts, which has become very easily accessible, especially for Facebook and Twitter. In effect, the problem of popularity by Sibyls has an impact on the market of social advertising. A survey conducted by the BBC revealed that companies lose huge sums of money for paid Facebook ads that get clicks and likes from Sibyls instead of real people. This causes a great danger for most social medias as they depend on advertising as their main source of income.

Fake sybil accounts are used to attack en-mass to conduct social spam, identity theft or phishing attacks. This malicious behavior relies on sockpuppetry. We develop this concept in the next section.

2.2.3 Sockpuppetry

For centuries, the use of pseudonyms and other false identities exist. As mentioned before, the rise of Web 2.0 opened the door to create forums in which people can exchange ideas and remain anonymous at the same time if they wish. A unique user can create and use several accounts with different pseudonyms. This new form of creating fake

identities is named sockpuppetry.

On July 9, 1993 the term sockpuppet was used for the first time and it came into general use by 1996. *Oxford English Dictionary* defines a "sock puppet" as a person who is controlled by someone else. In 2000, *U.S. News and World Report*, extend the idea of sockpuppeteer as a *minion*.

The sockpuppets being the multiple fake accounts created by the same user, they are the source of several types of manipulation such as those created to praise, defend or support a person or an organization [Stone and Richtel, 2007] or to manipulate public opinion [Elsner, 2013].

Nowadays, online voting has become very common. Malicious users create sockpuppets in order to influence the voting results by submitting multiple votes in favor of the sockpuppeteer. In other cases, they support each others by arguments, manipulate opinion or create the online illusions that many users support the sockpuppeteer or an opinion. This behavior belong to the "Sybil attack" category.

In addition, sockpuppets are used in marketing. In this case, the sockpuppeteer will create sockpuppets in order to support a product or on the contrary in order to fight another product. This type of sockpuppets is known as "Meatpuppet".

Another use of sockpuppets is to make a particular point of view look foolish. The idea is to create a negative sentiment against some ideas by formulating an easily refutable argument and then assigning it to its opponent. These sockpuppets are called "Straw Men".

Generally, sockpuppets are not accepted in user terms of services. In order to ban multiple identities accounts, the websites put in place systems of control and systematic suppression of these reputed malicious users. This is especially true for online gaming,

where having a sockpuppet may distort the outcome.

The sockpuppets behavior has particular characteristics because of their attack objectives [Kumar et al., 2017]:

- the user takes part in similar discussions and have the same opinions as the user's main identity;
- he knows remarkably well the methods, rules and members of the community, more than what could be expected of a newcomer;
- he may have the same IP address or the same Internet service provider;
- sometimes, the various accounts have similar names: zazaScok, zazaSockPuppet, zazaE, 1zaza1, and so on;
- all identities have the same hourly habits;
- the vocabulary and the grammar of the various sockpuppets are often close.

In some cases, sockpuppetry is not used for malicious objective, but for protection. For example a sockpuppet software for the US military has been under development since 2011. "*The technology support classified blogging activities on foreign-language websites to enable United States Central Command (CentCom) to counter violent extremist and enemy propaganda outside the United States*" according to Commender Bill Speaks [Fielding and Cobain, 2011].

In the next section we present the current protection solutions used by the social medias in order to protect their users and block the sockpuppets on their sites.

2.3 Protection Solutions

The majority of social medias are based on a centralized architecture in which users' data is stored on servers controlled by the social media providers. Facebook is often attacked in order to steal and sell users profiles and private messages [M., 2014].

However, Facebook and other centralized social medias offer solutions to protect the privacy of users. Many of these solutions are based on the contributions of their users. In addition, they create a reporting system where every user can send a notification or a report to the administrators if he find a malicious content. Some social medias created their own tools to help them to detect malicious behavior.

2.3.1 Protection of user information

Social medias strive to inform users on how they can protect their information and what users can do to contribute to this protection. That is why social medias offer guides about the protection of user information in the most user-friendly way possible.

Facebook. Facebook redesigned its guide to security; in the 'How to Keep Your Account Secure' section they offer new recommendations on how to prevent cyber-attacks through interactive graphics. In order to ensure everyone can read these tips, they are available in 40 languages and users can share them on their profile. The guide also warns users of the possibility of a phishing attacks.

LinkedIn. Facebook isn't the only social network which has improved its security information recently. LinkedIn also has a 'Security Blog' with helpful guidelines.

Twitter Twitter show its way of protecting users information in their help center where there is a wide security and protection section, users can access it from the 'help and support' tab in their profile.

Google Google propose a complete manual included in the web 'How to stay safe and secure online' to explain how to prevent cyber-attacks protecting user's passwords, checking Gmail's settings or verifying the emails' sender if the user thinks it might be a scam.

All these guides help the user so that he to knows how to protect his information using a privacy setting section, or by explaining their reporting systems,

2.3.2 Privacy setting

On the most widely used social medias of the market, the private information and personal exchanges can sometimes be more visible than one would like because of a voluntary lack of adjustment of the parameters of confidentiality, by the social medias. In fact, advertisment enterprises benefit from the tracking of users' information [Willis, 2014].

Social medias such as Facebook, Twitter, LinkedIn and SnapChat leave the choise to their users to adjust the protection of their personal information by explaining to them how to control their online presence.

For example, LinkedIn give the authority to their users to select who can see their activities, *i.e.* all the contents that users have shared, liked and commented on during their online activities. In addition, users can select who can see their relationships, so that it is possible to set the visibility of users relationships to the visitors visiting their profile. They are also able to ensure that their address book is exclusively private, and not accessible to their contacts. Figure 2.9 depicts an example of privacy setting section in LinkedIn.

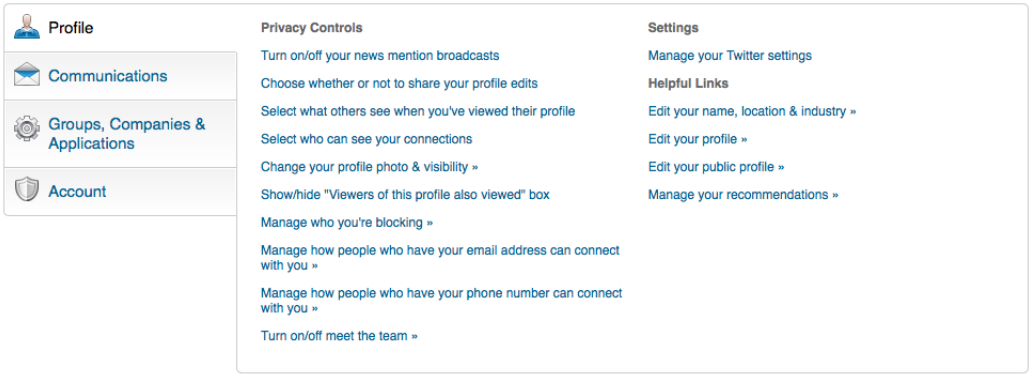

Figure 2.9 – LinkedIn's privacy setting section.

2.3.3 Reporting systems

All users of social medias have the option to report a malicious post or a malicious user to the social media's security team, who then check if the reported post or user is malicious or not and ban it if so.

For example, currently, in Wikipedia, any user that finds a malicious contribution can revert it and if he considers an account to be a sockpuppet, can open an investigation page about this account and report it to the 'check user', a privileged set of users who have access to the IP-address of all accounts, in order to discuss whether it is a sockpuppet or not. The administrators then check manually if this account is a real sockpuppet or not, by studying the account's behavior and the associated IP-addresses in order to block it with all the affected accounts.

2.3.4 Detection tools

Most of current social medias solutions to detect manipulators depend on the contributions and help from their legitimate users. In addition, some social medias such as Facebook and Twitter have detection tools that help their security team to detect the malicious users [Stein et al., 2011].

After receiving a report on a malicious posts or users, the security team checks this report manually and use their detection tool to make sure that the reported user is a manipulator or not. The scope and the specifications of these detection tools are not communicated by social medias, in order to keep the manipulators from learning how to skip from detection.

For example, Facebook [Stein et al., 2011] gives only an overview of their immune system, a defense system against phishing, fraud and spam. The system performs real-time checks and classification on every read and write action on Facebook's database, in order to protect its users from malicious activities. This service is very high level because it takes into account some important details of the undesirable activities on the Facebook site namely false profiles, harassment, compromised accounts, malware and spam. Boshmaf *et al.* show that this system is able to block only 20% of the accounts used by socialbots [Boshmaf et al., 2011].

Facebook has another solution called the EdgeRank algorithm [Bilge et al., 2009] which assigns to each post a score according to certain characteristics, such as the number of "likes", the number of comments and the number of reactions posted by a user. Therefore, the higher then EdgeRank score is, the lower the chance of the user being a spammer is. The disadvantage of this approach is that spammers could also build up a network and post comments between them in order to get a high EdgeRank score.

Twitter was said to be working on a spam bot defense system a few years ago [TwitterInc., 2012], an idea that was shut down for reasons unknown [Zackn, 2017].

2.4 Conclusion

In this chapter, we presented the six types social medias and the manipulation model that can be applied on them. We described also the main social attacks used by manipulators on social medias. Finally, we presented the solutions proposed by social medias in order to protect their users from manipulations.

The social medias solutions against manipulation are not fully automated: a lot of time and effort is spent by human administrators to detect and ban the manipulators. The mean of life time of a spam in Twitter until detection is about 269 days [Yardi et al., 2009], while it takes approximately 75 days in average for a sockpuppet account to get blocked on Wikipedia [Tsikerdekis and Zeadally, 2014a]. This long life gives the manipulator the opportunity to do many attacks before the detection.

In the next chapter, we present the solutions proposed in the literature to detect automatically the manipulations and the manipulators.

CHAPTER 3

MALICIOUS BEHAVIOR: DETECTION AND GROUPING

As elaborated in the previous chapter, most of social medias have their own solutions to protect their system, but these solutions are not fully automated, thus wasting time and energy in order to detect manipulators. This highlights the importance to create fully automated algorithms to detect these manipulators.

This chapter presents a review of the literature on malicious behavior automatic detection.

Some attacks on social medias are performed using single malicious account while other use groups of malicious accounts. The most well known attack using a single account is the spam. Some manipulators create many malicious accounts in order to spam using multiple identities, distribute malware and perform identity fraud [Richmond, 2010]. Other manipulators create false accounts to increase the visibility of niche content, forum posts, and fan pages by manipulating votes or view counts [Norajong, 2010; Sture, 2010].

A group of sockpuppets, *i.e. multiple accounts*, is a list of accounts created by the

same user or by many users with the same objective. Two issues have to be tackled to detect malicious behaviors in social media: the first focus on the detection of single malicious accounts while the second focus on the detection of communities or groups of malicious accounts.

Before starting the overview of the literature solutions, we identify and categorize the different criteria to detect and group manipulators. Then, we define the basic concepts of machine learning that are used in the following sections. We continue by reviewing the solutions that detect a single fake account such as spam accounts and multiple identities accounts.Then, we present the community detection algorithms used to group multiple identities. Finally, we review the solutions to group malicious behavior accounts created for the same objective in order to achieve common attacks from different identities.

3.1 Analysis Grid

This section identifies and categorizes the different criteria to detect and group manipulators. We build a set of six main criteria to evaluate the solutions proposed in the literature.

Analysis approaches (AP). Many solutions were created in order to detect or group the malicious behavior in social medias. In order to achieve this, two main analysis approaches are used: verbal communication (VC) analysis and non-verbal behavior (NVB) analysis.

Detection through verbal communication relies on the analysis of textual content, such as studying the links in post on Facebook, detecting sentiments from tweets on Twitter, or checking repeated words from Reddit's comments.

Detection through non-verbal behavior does not take into account the semantics of

the users interactions. Its focus is on the other aspects of the behavior of the user using descriptive values, such as the number of friends' requests on Facebook, the average of tweets every day, the average of added characters in each comment on Reddit.

Method Objective (MO). Many methods detect single malicious accounts (SMA) such as spammers or sockpuppets, others focus their work on the grouping of multiple identities (GMI).

To evaluate their detection method, most researchers use machine learning algorithms. When the focus is on the grouping of multiple identities accounts the methods use graph structure metrics.

Public information (PI). The proposed solutions are built on data from social medias, where these data are accessible for anyone (public) or accessible only to authorized users (private) such as system administrators. The dependency on private information has positive and negative effect. Most of the time the use of private information such as IP address will guide to better accuracy, but the negative point is that these solutions are applicable only by the social media engineers. This generates difficulties in the reproduction and generalization of these methods.

Generalization (G). The proposed methods vary in terms of generalization. Some of them are created for a specific social media site or a particular attack and thus cannot be applied on other social sites. Other methods study the malicious behavior in different OSMs and propose a general solution that can be applied on most of OSMs.

This set of the criteria raised in our study makes it possible to establish the following analysis grid. Table 3.3 in section 3.6 use this grid to compare the different approach based on the mentioned criteria.

			MO			
Category	Approach	AP	SMA	GMI	PI	G

Table 3.1 – Analyses grid for the criteria of evaluation for the detection and grouping of manipulation solutions. The abbreviations used to present the criteria are: AP : Analysis approaches; MO : Method Objective; SMA : Single malicious accounts; GMI : Grouping of multiple identities; PI : Public information; G : Generalization.

3.2 Machine Learning

Machine learning is the subfield of computer science that, according to Arthur Samuel in 1959, gives "*computers the ability to learn without being explicitly programmed*". Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the construction of algorithms that can learn from and make predictions on data. Such algorithms overcome strictly static program instructions by making data-driven predictions or decisions, through the building of a model from sample inputs[1]. In machine learning a *dataset* of observations, called *vector*, comprises a number of variables called *features* or *attributes*.

Supervised learning is the machine learning task of inferring a function from *labeled* training data. The *training data* consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value. A supervised learning algorithm uses the training data to produce an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the *class* labels for unseen instances[2].

A subclass of supervised learning problems is *binary classification* , where there are only two labels for class features. For example, a post or comment in a social media may be either spam or legitimate. *Multi-class classification* is the supervised learning with

[1] $https://en.wikipedia.org/wiki/Machine_learning$
[2] $https://en.wikipedia.org/wiki/Supervised_learning$

multiple labels for class feature.

Table 3.2 shows a small loan application dataset with four attributes. The first attribute is Age, which has three possible values, young, middle and old. The second attribute is Has_Job, which indicates whether an applicant has a job. Its possible values are true (has a job) and false (does not have a job). The third attribute is Own_house, which shows whether an applicant owns a house. The fourth attribute is Credit_rating, which has three possible values, fair, good and excellent. The last column is the binary Class attribute, which shows whether each loan application was approved (denoted by Yes) or not (denoted by No) in the past.

ID	**Age**	**Has_job**	**Own_house**	**Credit_rating**	**Class**
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Table 3.2 – An example of a supervised learning for a loan application dataset.

From this data, the machine learning algorithm has to learn a classification model that can be used to classify future loan applicants. That is, when a new customer comes into the bank to apply for a loan, after inputing his/her age, whether he/she has a job, whether he/she owns a house, and his/her credit rating, the classification model should predict whether his/her loan application should be approved.

3.3 Spam Detection

Spam detection has been for twenty years a prominent field of application for the machine learning algorithms [Cohen et al., 1996; Cortes and Vapnik, 1995; Drucker et al., 1999; Joachims, 1998; Pantel and Lin, 1998; Sebastiani, 2002].

3.3.1 Web and e-mail Spam

Electronic spam is an unsolicited electronic communication, generally posted in large quantities on the same site or sent via e-mail for advertising purposes.

There is a large body of prior work on measuring e-mail spam [Anderson et al., 2007; Kanich et al., 2008; Kreibich et al., 2009; Xie et al., 2008]. The first publications on the classification of e-mail spam based on automatic learning date from 1996 [Cohen et al., 1996].

The *Ripper* method [Cohen et al., 1996] is a classifier capable of defining the context of a word. This classifier uses a set of rules (presence or absence of dictionary words) that is automatically constituted during the learning phase. Ripper also performs an optimization phase to readjust the rule sets.

In 1998 two methods [Pantel and Lin, 1998; Sahami et al., 1998] propose the use of a Naïve Bayesian (NB) classifier [Russell and Norvig, 1995] for spam filtering. Sahami *et al.* select 500 characteristics of each message and find that the binary classification (ham, spam) is more efficient than the multi-class classification based on the type of the message (eroticism, phishing, etc.) [Sahami et al., 1998].

A method in 2002 [Sebastiani, 2002] was interested in the recognition of spam by the thematic classification of messages. It represents the message to be classified in the form of a vector of features. Then it evaluates the distance (usually Euclidean or

Manhattan distance) between this vector and the similar vectors of each class (spam or not), associating the message with the one that has the closest prototype vector [Drucker et al., 1999]. This method and Ripper showed similar results, but there is an improvement when learning is done individually for each user rather than collectively for all users.

In the literature, several research studies have confirmed the effectiveness of Support Vector Machine (SVM) [Cortes and Vapnik, 1995] approaches for filtering spam. Kolcz focused on class-specific classification errors [Kolcz and Alspector, 2001], while Islam *et al.* select attributes to improve the classification of e-mail spam [Islam et al., 2005]. Joachims improved linear SVM methods on large scale problems and describes its efficient implementation in a generalized classification context [Joachims, 1998]. [Sculley and Wachman, 2007] avoided working on synthetic data and carried out more realistic tests in the learning of messages and the detection of spam. To this end, he deployed a method called ROSVM (Relaxed Online SVM) which slightly improved the efficiency of the classification of spam especially by reducing greatly the computational cost of updating the hypothesis, in particular by training only on actual errors.

Recently, Zhiwei *et al.* propose a method that uses data mining. The objective is to find the optimal hybrid model with the higher accuracy. The experimental results show accuracy improvement in email spam detection by using hybrid techniques compared to single classifiers.

3.3.2 Social Media Spam

Since the emergence of social medias, and after the large body of work on e-mail spam, researchers have begun measuring social spam as well. Studies such as [Thomas et al., 2011a; Yardi et al., 2009] have focused on measuring spam on Twitter.

Monarch [Thomas et al., 2011a] is a system for identifying spam links on Twitter. The application processed 55 GB of data collected from about 250 000 URLs and processed 107 features/dimensions related to the network and content properties of the pages at each URL. They identify spam campaigns in real time using statistical learning algorithms. Another experiment run on Twitter spam is proposed by [Yardi et al., 2009]. They create a hash tag[3] on Twitter, and observe the interaction of the spammers with it in order to examine their behavioral patterns. They also study some features that might allow one to distinguish a spammer from legitimate users, such as the number of friends and followers. This study reveals that the behavior of spammers is different from legitimate users, especially by the way tweets are produced, the attitudes of the followers and the fans of their pages.

In the same vein, Whittaker *et al.* describe their scalable learning method to detect phishing and improve Google's blacklisting system. They train the classifier on a noisy dataset consisting of millions of samples. They identify more than 90% of phishing pages. However, many social media messages still link to malicious sites, and detecting them remains a challenge [Whittaker et al., 2010].

Spam in blog comments has also attracted the attention of researchers who have deployed automatic learning methods [Kolari et al., 2006; Nagamalai et al., 2010], respectively SVM algorithm and Bayesian classifiers. Mishne *et al.* use linguistic modeling to find semantic differences between blogs containing spam links in comments and the targeted sites to which they refer (for example, containing adult content) [Mishne et al., 2005]. Similarly, Markines *et al.* have applied machine learning algorithms (SVM and Adaptive Boosting [Freund and Schapire, 1995]) to detect spam in the social bookmarking

[3] a word or phrase preceded by a hash sign (#), used on social medias to identify messages on a specific topic.

site BibSonomy. This study leverages features that are unique to social bookmarking sites to isolate spam, such as tag frequency and popularity [Markines et al., 2009].

Facebook also attracts researchers to detect its spammers because it contains the highest number of spam of all social networks [Adam, 2013]. An example is Gao *et al.*, who start by filtering the data crawled from Facebook to keep only the posts that contain URLs, their hypothesis being that every spammer will try to redirect the social media user to an outside fake site. Then, they link the similar posts by checking if they share the same destination or the same text content. Finally, they cluster 1 402 028 posts with links by checking if the link is guiding to a fake website or not, in order to detect the malicious users and posts. The authors obtain interesting results with this method, since they detect 93.9% of malicious wall posts on their test dataset [Gao et al., 2010].

After a study by Hu *et al.* on Sina Weibo, the most used social network in China, three characteristics inherent to the behaviors of spammers can be identified: their aggressiveness in advertising, the duplication of the contents of others posters and the aggressive behavior of the followers. In their work, the authors adopt a particular approach, based on the combination of models to construct groups of classifiers. The different groups of classifiers formed are then tested on each of the three characteristics generated, and function jointly as a single anti-spam classifier. The goal of this combination of classifiers is to improve the performance of the spam filter. The classification accuracy obtained is 82.06% [Hu et al., 2013].

Bosma *et al.* use reports of spam generated by users as a tool for building an unsupervised environment for the detection of spam on social networks [Bosma et al., 2012]. Their approach is to count the number of reports issued against a suspicious user, and then weight these reports based on the reputation of the user. The determination of the

reputation and the reliability of users in social networks have been well studied [Antonio et al., 2012] and provides promising features in the classification of social spam. Their model uses a Bayesian classifier and links messages with similar content, regardless of other characteristics. This study is one of the few that has served as a test model on non-public data, including private messages, spam reports and user profiles of a major Dutch social networking site.

A more recent study improves the identification and filtering of spam in social networks using probabilistic methods [Xu et al., 2013]. The concept consists in using the established social relationship between the sender and the receiver to determine their proximity, and deduce the confidence value. This value is a determining factor in the calculation of probabilities which serve as an indicator for the classification.

In summary, despite the large number of studies that have focused on detecting social spam, the social medias are still rife with Spam even with all the spam detection methods, because the creation of fake accounts is still easy as the identity verification process can be bypassed [Scott, 2017]. Furthermore, most of the proposed methods are not implemented and applied in real time by the developers of social medias [Tsikerdekis and Zeadally, 2014b].

Another issue is that the attackers, most of time, do not use only one account for attacks, but create multiple fake identities [Kelly, 2012].

Others tools are needed that leverage deeper, more nuanced data, in order to bolster robustness against evolving attacker strategies and detect the attacks using multiple identities.

3.4 Multiple Identities Detection

Multiple identities detection approaches in OSMs can be classified into two categories: feature-based approach and social-graph-based approach. In the first approach, researchers focus on extracting features from users' profiles [Yang et al., 2014] (such as acceptance rate of incoming/outgoing requests, invitation frequency, etc.) and observing user activity patterns [Wang et al., 2013] to derive a machine learning classification to identify fake/Sockpuppet user accounts. In social-graph based approaches, researchers use community detection algorithms in order to detect and group sockpuppets sharing the same interest.

Attacks using multiple identities are one of the most prevalent and practical attacks against systems [Douceur, 2002]. In this kind of attack, a user creates multiple fake identities, known as Sockpuppets or Sybils, to unfairly increase their power and influence within a target system or community. Attacks using multiple identities proved their efficiency against Peer-To-Peer (P2P) systems [Lian et al., 2007], anonymous communication networks [Bauer et al., 2007], and sensor networks [Newsome et al., 2004].

Social medias are also targets of multiple identities attacks. Sockpuppets have been identified on most social media sites such as Wikipedia [Solorio et al., 2013a; Tsikerdekis and Zeadally, 2014a], Facebook [Wang et al., 2012a], Twitter [Alsaleh et al., 2014; Crabb et al., 2015; Thomas et al., 2011b], Renren [Yang et al., 2014], Forums [Bu et al., 2013; Liu et al., 2016] and Disqus [Zheng et al., 2011a].

Closely related to sockpuppet detection is author identification, that is the task of identifying the original author of a document [Johansson et al., 2015; Juola, 2012; Narayanan et al., 2012; Novak et al., 2004; Paul et al., 2015; Qian and Liu, 2013]. Recently, researchers have used multiple accounts of users across different social platforms

to identify malicious users [Venkatadri et al., 2016]. In addition, they identify accounts operated by the same user across different social platforms [Jain et al., 2013; Silvestri et al., 2015; Zafarani and Liu, 2013; Zheng et al., 2011a].

Some researches [Kumar et al., 2017; Solorio et al., 2013b; Tsikerdekis and Zeadally, 2014a; Yang et al., 2014; Zheng et al., 2011a] focus on detecting sockpuppetry through author identification. They identify sockpuppets as accounts, without linking them to others sockpuppets created by the same user. This detection, as for the case of spams, is most of the time done through supervised machine learning algorithms and with binary classification to distinguish between *fake* and *legitimate* accounts.

The detection of sockpuppets on social medias was introduced in 2011 by Zheng *et al.* who propose to detect sockpuppet pairs in one forum and in two different forums [Zheng et al., 2011a]. The authors use a dataset from Uwants and HK discuss. On the basis of Detection Score, they try to find out similar keywords used by different people. They conclude that their method can be used for monitoring the online discussion forum but that the results have to be checked by the police and the forum moderators, because there are still a lot of sockpuppet pairs that the method cannot detect.

Solorio *et al.* use natural language processing techniques to detect on Wikipedia the users who maintain multiple accounts based on their verbal communications. The features rely on the text analysis such as alphabet count, number of tokens, emoticons count or the use of words without vowels (e.g. try, cry,...). These features are tested on all revisions made by the users on pages throughout Wikipedia. A Support Vector Machine (SVM) algorithm shows 68.83% overall accuracy using an experimental dataset of 77 cases of legitimates users and sockpuppets [Solorio et al., 2013a]. Further research expands the dataset to 623 cases, where the reported best result gives 73%

F-Measure [Solorio et al., 2013b]. These approaches only use verbal behavior features, but these are not always accurate because the manipulators can change their writing method in order to skip from automatic detection.

Tsikerdekis and Zeadally propose a method to detect identity deception through the use of non-verbal user behavior such as the number of revisions made by a user in a specific time window since their initial registration to the website, and the total number of bytes added and total number of bytes removed. They use publicly available data from Wikipedia, and classify them on machine learning algorithms. For testing purpose, they sample 7 500 cases of sockpuppets and include several machine learning algorithms such as Support Vector Machine (SVM), Random Forest (RF) and Adaptive Boosting (ADA). The overall accuracy is 71.3%. This detection method focuses only on the sockpuppet account itself and does not group the sockpuppets created by the same user [Tsikerdekis and Zeadally, 2014a].

Yang *et al.* integrate a real time detection of sybil accounts in the Chinese OSM RenRen by studying non-verbal behavior. They characterize the RenRen accounts according to the following features: Invitation Frequency, Outgoing Requests Accepted, Incoming Requests Accepted and a measure of the mutual connectivity of a user's friends. They apply a SVM classifier to their ground truth dataset of 1 000 normal users and 1 000 Sybils. They randomly partition the original sample into 5 sub-samples, 4 of which are used for the training of the classifier, and the last is used to test the classifier. The results show that the classifier is very accurate, correctly identifying 99% of both Sybil and non-Sybil accounts. This technique works well on Renren but the chosen features cannot be generalized directly. For example, Sybils on Twitter do not need to create social connections, and instead send spam directly to any user using "@" messages [Yang et al.,

2014].

Recently, Kumar *et al.* study sockpuppetry behavior across nine discussion communities. They show that sockpuppets differ from ordinary users in terms of how they write and interact with other sockpuppets. They also show that pairs of sockpuppets controlled by the same individual are more likely to interact in the same discussion at the same time than pairs of ordinary users [Kumar et al., 2017].

3.5 Grouping multiple identities accounts

In the previous sections, we presented the studies that identify the malicious accounts using machine learning to separate malicious and legitimate accounts. As mentioned above, the manipulators may create multiple identities to make their attacks. Hence, several works focus on grouping the malicious accounts created for the same attacks in order to stop these attacks from their source. The approaches are based on social-graph and more precisely they use community detection algorithms.

In this section, we present the most known community detection algorithms, then we review the proposed approaches that group the multiple fake identities.

3.5.1 Community detection

Community detection is a growing field of interest in the area of social medias applications. A large number of algorithms for community detection have been introduced in recent years [Largeron et al., 2017; Lund, 2017; Zardi et al., 2016].

Since there are many proposed approaches, we study in this section an overview of those that have received the most interest from the scientific community [Fortunato, 2010]. These approaches illustrate the diversity of methods and give an overview of the

techniques classified according to their methodological principles [Danon et al., 2005; Lancichinetti and Fortunato, 2009; Papadopoulos et al., 2012]. Many of the algorithms try to maximize modularity [Nabaa et al., 2010], a concept proposed by [Newman and Girvan, 2004] and described in the next section. We continue by presenting the hierarchical methods and then those related to the optimization of quality measures. Finally, we focus our attention on model-based algorithms.

Before proceeding to the examination of the different methods, let us clarify the relationship between the problem of the detection of communities and the partitioning of graphs [Papadopoulos et al., 2012]. On the one hand, graph partitioning consists in grouping the nodes of a graph into a generally predetermined number of homogeneous subgroups by minimizing the number of links between the different groups. On the other hand, the community detection algorithms perform the same operation with or without requiring the knowledge of the number of communities.

A network graph is composed by a set of nodes that represent the accounts and a set of links between these nodes called edges. The interactions between accounts are represented as the edges between nodes. In addition, in weighted graph each edge have a weight, that represent the robustness of link between the two related nodes.

3.5.1.1 Modularity

Modularity has been introduced by Newman to measure the quality of community detection algorithms [Newman and Girvan, 2004]. He defines it as the following:

$$Q_n = \frac{1}{2*m}\sum_{i,j}(a_{i,j} - \frac{d(u_i)*d(u_j)}{2*m})\delta(u_i,u_j) \tag{3.1}$$

where $a_{i,j}$ is 1 if the nodes u_i and u_j are linked and 0 is not. The variable $d(u_i)$ is the number of neighbors of u_i, m is the total number of edges in the graph, and δ is 1 if u_1

and u_j are members of the same community and 0 if not.

In the case of weighted networks, Blondel provide the following formula [Blondel et al., 2008]:

$$Q = \frac{1}{2 * m} \sum_{i,j} (wn_{i,j} - \frac{k_i * k_j}{2 * m}) \delta(g_i, g_j) \tag{3.2}$$

where $wn_{i,j}$ represents the weight of the edge between account u_i and account u_j, k_i and k_j are the sum of the weights of the edges attached respectively to nodes u_i and u_j, m is the sum of all of the edge weights in the graph, g_i and g_j are the communities of the nodes and δ is a simple delta function.

Modularity is one measure of the structure of networks or graphs. It was designed to measure the strength of division of a network into modules (also called groups, clusters or communities). Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.

3.5.1.2 Hierarchical agglomerative algorithms

The general principle of hierarchical classification algorithms consists in grouping the nodes iteratively into communities. At the beginning, each node is a community (there are as many communities as nodes). The communities are merged in pairs until a large community that contains all the nodes of the graph is obtained. At each step of grouping two communities, a metric (quality function) is calculated. The partitioning with the highest value of the metric considered represents the best partitioning of the graph into communities. The result of hierarchical algorithms is a dendrogram, a tree diagram frequently used to illustrate the arrangement of the clusters, containing the history of community joins at each stage. In the following, we present $Walktrap$ and $Louvain$ as

examples of agglomerative algorithm, because they have interesting results [Fortunato, 2010].

Walktrap. The $Walktrap$ algorithm [Pons and Latapy, 2006] is an approach based on random walks. The general idea is that if a random walk is performed on the graph, then it is more likely to stay within the same community because there are only less edges that lead outside of this given community than edges that lead to nodes inside the community. Starting with n communities each containing only one node, the algorithm looks for the two closest communities, merges them and recalculates the distances. $Walktrap$ calculates for each pair of vertices (i,j) a weight w_t (i,j) corresponding to the probability that a random walk goes from i to j in t steps. It deduces a distance between the pairs of nodes, described in the following equation:

$$dist_t(i,j) = \sqrt{\sum_{v \in V} \frac{|w_t(v,i) - w_t(v,j)|^2}{k_v}} \tag{3.3}$$

where k_v is the number of neighbors of node k.

Merges are performed until only one community covering the whole graph (see Figure 3.1) is obtained.

Finally, for each of the intermediate cuts, the algorithm evaluates the modularity and retains the scale that gives the highest modularity (see Figure 3.2).

Louvain. The $Louvain$ community detection algorithm [Blondel et al., 2008] is a greedy optimization method that attempts to optimize the modularity of a partition of the network. The optimization is performed in two steps. Firstly, the method looks for small communities by optimizing modularity locally. At the beginning each node is considered as one community. The nodes are scanned in a random order. Then, the method verifies for each node if its placement in the community of one of its neighbors brings a gain in modularity. If this is the case, the node is moved to that community,

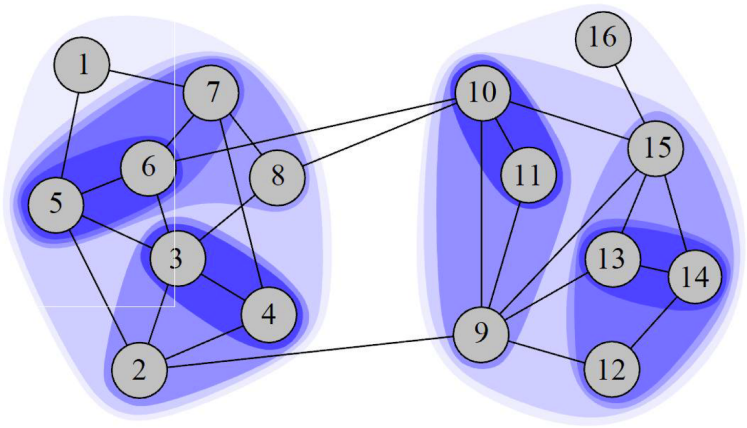

Figure 3.1 – Example of a hierarchical structure of communities found by Walktrap (Extract from [Pons and Latapy, 2006]). The corresponding dendrogram is shown in Figure 3.2.

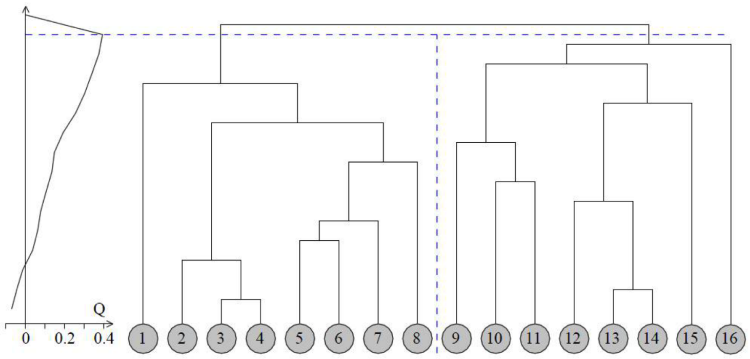

Figure 3.2 – Dendrogram associated with the communities of Figure 3.1 found by the Walktrap algorithm (Extracted from [Pons and Latapy, 2006]). The Q results represents the modularity. The partition presented by a horizontal section of the dendrogram is the one with the maximum modularity.

otherwise it remains in the old community. Secondly, it aggregates the nodes belonging to the same community and builds a new network whose nodes are the communities. These steps are repeated iteratively until the modularity does not grow anymore and a hierarchy of communities is produced.

3.5.1.3 Hierarchical divisive algorithms

The divisive algorithms consist in splitting a network into several communities by iteratively eliminating the links between the nodes. They begin with a single community (the whole network), at the top of the dendrogram, until n communities with a single node representing the leaves of the dendrogram are found. At each iteration, any connected network is considered as a community. Existing methods differ in the choice of the links to be eliminated and in the weight given to the links. We retain the $Edge - Beetweness\ Centrality$ as an example of a divisive algorithm.

Edge-Betweenness Centrality. The Edge-Betweenness Centrality algorithm [Newman and Girvan, 2004] is the most classical separative method, based on a measure of centrality. For a given edge e_i, this measure is calculated as:

$$EBC(e_i) = \sum_{i,j,i \neq j} \frac{NP_{u_i,u_j}(e_i)}{NP_{u_i,u_j}} \tag{3.4}$$

Where $NP_{u_i,u_j}(e_i)$ is the number of shortest paths from nodes u_i to u_j through the edge e_i and NP_{u_i,u_j} the total number of shortest paths from nodes u_i to u_j.

This measure is defined as the number of the shortest paths of the graph that pass through this edge. There are, in fact, few edges connecting the different communities and the shortest paths between two nodes of two different communities have a high probability of passing through these edges. The proposed algorithm calculates the centrality of intermediation for each link of the graph, and then removes the link with the strongest centrality (i.e. having the highest EBC value). Then the centrality of intermediation for all the links of the resulting graph is recalculated and the process is iterated until all the links have been removed and there are as many communities as nodes in the graph (see Figure 3.3).

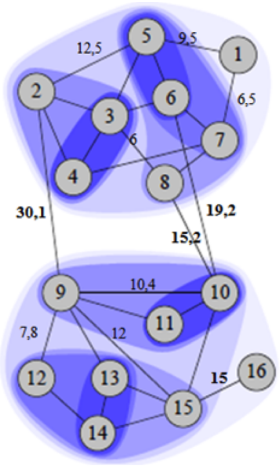

Figure 3.3 – Edge-Betweenness Centrality calculation in a network

However, the execution time of this algorithm is very high. The calculation of centrality of intermediation is costly and has to be executed at each iteration. The algorithm has a complexity of the order of $O(m^2n)$ where m is the number of links and n is the number of nodes. It can therefore only be used on small graphs.

3.5.1.4 Quality optimization algorithms

There is a number of methods that are founded on the optimization of graph-based measures of community quality, and especially the measure of modularity. Approximate modularity maximization schemes abound in the literature. Apart from the seminal greedy optimization technique [Newman, 2004], and speeded up versions of it, such as max-heap based agglomeration [Clauset et al., 2004] and iterative heuristic schemes [Blondel et al., 2008], more sophisticated optimization methods have been proposed. For instance, we can cite extremal optimization [Duch and Arenas, 2005], simulated annealing [Massen and Doye, 2005] and spectral optimization [Newman and Girvan, 2004]. Methods aiming at the optimization of local measures of community quality, such as local and subgraph modularity [Luo et al., 2008], also belong to this category.

One of the fastest and most well known algorithm [Fortunato, 2010] is $Fast\ Greedy$.

Fast Greedy. The Fast Greedy algorithm [Clauset et al., 2004] uses the modularity function Q defined in equation 3.1. Each possible division of the graph has a value Q. This value is high for a good division of the graph and weak for the bad divisions. By observing all the possible divisions of the graph, we note that the exhaustive computation of Q takes an exponential amount of time in the number of nodes. This method can be used on networks of small size (about twenty or thirty nodes) to have a reasonable time of execution.

Optimization strategies have been proposed to improve the execution time of this algorithm. It begins by considering each node as a community apart, then merge the communities into pairs, choosing at each stage the pair of communities whose fusion gives the largest increase of Q in relation to the previous iteration. Since the merging of non-adjacent communities never produces a larger increase in Q, several possible mergers are avoided. In this case, there are m pairs and $n-1$ possible mergers to link all communities together. Thus, the total execution time is in $O(mn)$. The algorithm stops when it is not possible to increase the modularity any more. The maximum value of Q corresponds to the best community structure.

3.5.1.5 Model-based algorithms

A large category of methods considers a dynamic process taking place on the network that reveals its communities, or that an underlying model of statistical nature can generate the division of the network into communities. Examples of dynamic processes are label propagation [Raghavan et al., 2007], synchronization of Kuramoto oscillators [Arenas et al., 2006], and diffusion flow, better known as Markov Cluster Algorithm [Van and

Stijn, 2001]. In addition, community detection can be cast as a statistical inference problem [Hastings, 2006], assuming some underlying probabilistic model, such as the planted partition model, that generates the community structure and estimates the parameters of this model.

Other model-based approaches rely on the principle that a good clustering is determined by a low encoding cost. Thus they perform community detection by finding the cluster structure that results in the lowest possible cluster encoding cost [Chakrabarti, 2004; Rosvall et al., 2009]. We detail two of most known and efficient algorithms [Fortunato, 2010] below : $Label\ Propagation$ and $InfoMap$.

Label Propagation. The Label Propagation community detection algorithm [Raghavan et al., 2007] is an approach in which every node is assigned one of k labels. The method then proceeds iteratively and re-assigns labels to nodes so that each node takes the most frequent label of its neighbors in a synchronous manner. The method stops when the label of each node is one of the most frequent labels in its neighborhood. It is very fast but yields different results based on the initial configuration (which is decided randomly).

InfoMap. The Infomap algorithm [Rosvall and Bergstrom, 2008] is often considered as the most effective model-based algorithm. It seeks to find an optimal encoding of the graph. To do this, it seeks to minimize the length of the description of the path traveled by a random walker, the nodes being described according to a prefixed numbering. Each node is thus described by the identifier of the community to which it belongs and a unique identifier within this community.

When describing the path of a walker, it begins by giving the identifier of the community in which it is located, then the identifier of the node on which it is. As long as

the walker remains within the same community, it can describe the new node that it reaches by its "local" identity within the community, without repeating the identifier of this community. When its move takes it out of the community, to describe the new node on which it is on, it give the identifier of the new community, then the identifier of the node within the community. It is a simple system such as that used for postal addresses: there are many streets in France called Victor Hugo (node identifier), but they can be differentiated by the element in which they are included (city name, corresponding to the community identifier). The operation of the algorithm is illustrated in figure 3.4.

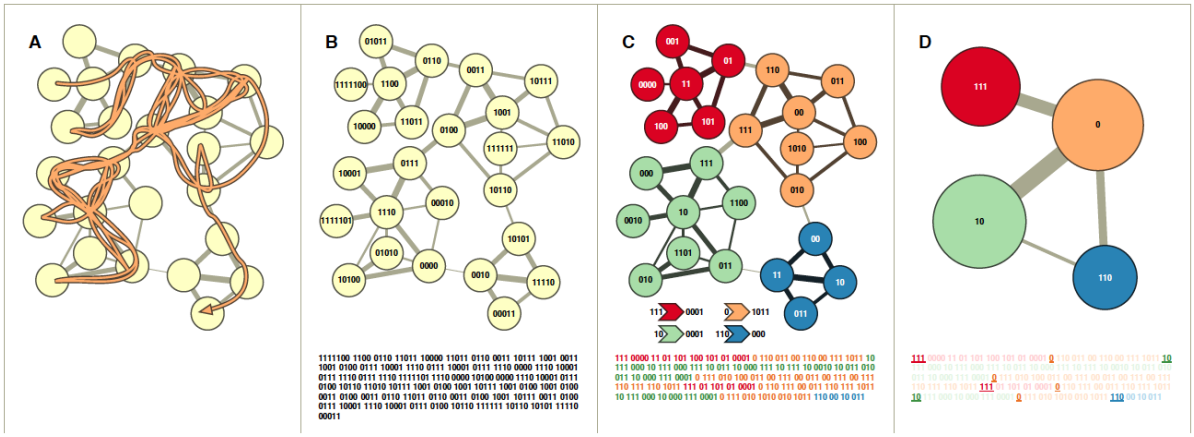

Figure 3.4 – Illustration of the InfoMap process (from [Rosvall and Bergstrom, 2008]). The image A represents the path of a random walker, which one wishes to encode. The image B presents a basic means of making this encoding: a binary identifier is assigned to each node, and the sequence of the identifiers is used to encode the path (in black, below). The images C and D show how to optimize this encoding by assigning identifiers to the clusters, and, for each node, an internal identifier has its cluster. We can see that the encoding (below, colored) is more efficient than in case B. This is the solution leading to the minimal encoding that InfoMap seeks.

In addition to its efficiency, InfoMap is very fast and thus allows to process very large graphs.

These presented methods have been used by researchers [Liu et al., 2016; Stringhini et al., 2015] in order to group the malicious accounts, and especially sockpuppets.

3.5.2 Grouping Sockpuppets

As mentioned in chapter 2, sockpuppets are the multiple fake accounts created by the same user in order to manipulate on social medias. As presented before, Social sybils are a special type of sockpuppets created in order to influence voting results by submitting multiple votes, or in order to support each others by arguments. They can also manipulate opinions by creating an online illusion that many users support an opinion.

To address the problem of multiple identities on OSMs, researchers have developed graph-based algorithms for the grouping of sockpuppets in different types of OSM such as online discussion forums [Bu et al., 2013; Zheng et al., 2011a] and social networks [Gani et al., 2012; Liu et al., 2016; Viswanath et al., 2010; Zheng et al., 2011a].

Sybil Community Detectors. SybilGuard [Yu et al., 2008b], SybilLimit [Yu et al., 2008a], and SybilInfer [Danezis and Mittal, 2009] are algorithms that perform decentralized detection of Sybil nodes on social graphs. In fact, all of these algorithms are based on a common hypothesis that sybils have difficulty forming relations with legitimate accounts (honest nodes).

The rarity of these relations causes the sybil nodes to be outliers in the graph, *i.e.* there is a small quotient-cut separating the sybil nodes from the "honest" regions of the graph. SybilGuard, SybilLimit, and SybilInfer all leverage specially engineered random walks for this purpose. Although all of these algorithms are implemented differently, they all generalize to the problem of detecting communities of Sybil nodes [Viswanath et al., 2010].

SybilRank [Cao et al., 2012] allows OSM to rank users according to their perceived likelihood of being fake (Sybils). SybilRank can scale to graphs with hundreds of millions of nodes. The authors implemented SybilRank in the Italian telecoms operator *Tuenti*.

They observe large clusters of Sybil users in regular topologies (star, mesh, tree, ...) that are connected to the honest communities through a limited number of attack edges. SybilRank is notable because it is the only Sybil community detector that has been successfully evaluated and deployed on a real social graph against attackers, while the other studies all base their results on simulated attacks.

Sockpuppets Communities. Other researchers base their approach on the usernames that sockpuppets use (e.g. considering that they are similar [Liu et al., 2016]), their opinion towards topics (e.g. considering that they have the same opinion [Bu et al., 2013]), and their interactions (e.g. considering that they reply in support of each other's posts [Zheng et al., 2011a]).

Liu *et al.* propose to detect sockpuppets groups by analyzing the sentiment orientation of user accounts based on emotional sentences written in their posted comments. A graph with edges between the user accounts that have similar orientations to most topics is built. Then, after analyzing the behavior features of user accounts according to their posts and comments, they propose a method based on multiple random walks to remeasure the weight of each edge iteratively. Finally, they detect sockpuppet groups (*gangs*) using multiple community detection algorithms. They evaluate the modularity and the community size of their method on two Chinese forum's datasets using two community detection algorithms [Liu et al., 2016].

Bu *et al.* propose a sockpuppet detection algorithm which combines authorship-identification techniques and link analysis. Firstly, they create a social network model in which links between two accounts are built if they have a similar attitude to most topics. Finally, the link-based community detection for sockpuppet network is performed. The algorithm is evaluated on two Chinese datasets, but the accuracy of the detection is low;

between 13.1% and 44.5% depending on the selected dataset [Bu et al., 2013].

Other researches group malicious accounts by studying the social behavior [Beutel et al., 2013; Cao et al., 2014] or by checking the common IP addresses [Stringhini et al., 2015].

Stringhini *et al.* developed a system to detect malicious communities such as botnets acting on online services. This approach uses a bipartite graph between the set of online accounts and the set of IP addresses. Then, it performs clustering in order to find malicious communities that are accessed by a common set of IP addresses using a community detection algorithm (Louvain method described in section 3.5.1.2). Finally, it is evaluated on two real-world datasets. This method can detect accounts even if they do not perform any malicious activity as it works only on accounts accessed by multiple IP addresses. However, it does not distinguish between fake and compromised accounts [Stringhini et al., 2015], and it uses non-public information.

Two other works, SynchoTrap by [Cao et al., 2014] and CopyCatch by [Beutel et al., 2013], also focus on grouping malicious identities. These techniques look at the behavior of a group of accounts and check whether they have anomalous or malicious behavior. However, they are both limited, because they focus on detecting a specific attack behavior : *loosely synchronized actions*, where a group of malicious identities behave similarly at around the same time. For example, a group of sockpuppets following a set of Instagram profiles at around the same time can potentially be detected by such schemes.

Credit Networks. The previously mentioned algorithms are designed to detect multiple identities on OSMs, presumably so they can be banned. Other algorithms are designed to make social networks resistant to sockpuppetry. SumUp [Tran et al., 2009], GateKeeper [Tran et al., 2011], and Bazaar [Post et al., 2011] use max-flow

techniques [Goldberg and Tarjan, 1988] to limit the influence of Sybils attempting to attack social applications. SumUp focuses on collecting votes from legitimate accounts in a polling application, while Bazaar is designed to exclude fraudulent accounts from online marketplaces. SumUp assumes that all graph edges have a weight of one (e.g. one user, one vote), but GateKeeper and Bazaar expand upon this by assigning custom weights to the edges (using tickets in the former case, and positive transaction feedback in the latter case).

3.6 Synthesis

In the introduction, we proposed a set of criteria to evaluate the researchers approaches with respect to the issue of automatic detection and grouping of sockpuppets. In this section, we develop the proposed analysis grid by evaluating the presented solutions.

Table 3.3 is a sum up of the related works presented in this chapter based on the mentioned criteria.

This table shows that the methods used to detect the spams use verbal communication analysis more than non-verbal behavior, and that in addition they only detect single accounts.

The methods that detect multiple identities share the same criteria as spams detection methods, with one difference in the analysis approaches: multiple identities solutions use non-verbal behavior more the than the verbal communication.

The final category is the grouping of multiple identities where the proposed solutions use most of time the non-verbal behavior. By design, they cannot detect the single accounts. Most of grouping solutions use public information, but some of them are applicable to all social medias, and the others are proposed for a specific platform of

			MO			
Category	Approach	AP	SMA	GMI	PI	G
Spam	[Cohen et al., 1996; Islam et al., 2005; Joachims, 1998; Kolcz and Alspector, 2001; Pantel and Lin, 1998; Sahami et al., 1998; Sculley and Wachman, 2007; Sebastiani, 2002; Thomas et al., 2011a; Whittaker et al., 2010]	VC	✓	x	x	x
	[Gao et al., 2010; Hu et al., 2013; Markines et al., 2009; Thomas et al., 2011a; Whittaker et al., 2010]	VC	✓	x	✓	✓
	[Xu et al., 2013]	NVB	✓	x	x	✓
	[Bilge et al., 2009]	NVB	✓	x	x	x
	[Antonio et al., 2012; Bosma et al., 2012; Kolari et al., 2006; Mishne et al., 2005; Nagamalai et al., 2010; Yardi et al., 2009]	VC	✓	x	✓	x
Multiple identities	[Solorio et al., 2013a,b; Zheng et al., 2011a]	VC	✓	x	✓	x
	[Yang et al., 2014]	NVB	✓	x	x	x
	[Kumar et al., 2017; Tsikerdekis and Zeadally, 2014a]	NVB	✓	x	✓	x
Grouping	[Beutel et al., 2013; Cao et al., 2012, 2014; Danezis and Mittal, 2009; Yu et al., 2008a,b]	NVB	x	✓	✓	✓
	[Bu et al., 2013; Liu et al., 2016]	VC	x	✓	✓	x
	[Stringhini et al., 2015]	NVB	x	✓	x	✓

Table 3.3 – Comparison between different literature methods based on selected criteria. The abbreviations used to present the criteria are: AP : Analysis approaches; VC : Verbal communication; NVB : Non-verbal behavior; MO : Method Objective; SMA : Single malicious accounts; GMI : Grouping of multiple identities; PI : Public information; G : Generalization.

social medias.

In this chapter, we have seen that verbal and non-verbal communications are interesting axis, but that the overall sockpuppet detection rate was not sufficient for automatic discovery of such behavior. In fact, the manipulator can skip quite easily from the detection of deception through verbal communication because (1) he can change the text content and (2) the automatic detection of similar content through different verbal expressions is not reliable enough.

We conclude also that the detection of malicious accounts without knowing the associated accounts created for the same objective is not accurate, because if a fake account is blocked, the sockpuppeteer can create another one and continue his manipulation. However, there does not exist a single method in the literature that both detect and group sockpuppets.

Based on this review of the literature, we deduce that the best criterias that should be used to obtained a reasonable accuracy are in a **general** method built on well selected **public features** of **non-verbal behavior** to **detect** and **group** the malicious accounts of different social medias.

In the next chapters we present our proposed process *SocksCatch* which is composed of three phases. The first phase prepares the process and describes what type of data should be collected. The second phase detects the sockpuppet accounts using non-verbal features applied on machine learning algorithms. Finally, the third phase groups the detected sockpuppets using community detection algorithms.

CHAPTER 4

THE SOCKSCATCH PROCESS

As we have seen in the previous chapters, social medias are a privileged target for manipulators. In order to tackle this problem, researchers proposed several solutions to detect them. Part of these solutions only detects malicious accounts, but after the detection the manipulators can continue their attack rapidly by using another (fake) account. The other part of these solutions groups the malicious accounts by creating a graph containing all the platform accounts types (legitimate and fake). They split up this graph in *trust* and *attackers* zones. The trust zone is where the legitimate accounts are grouped together. The attackers zone is where the fake accounts are grouped together. The negative point of these methods is that they group all the accounts in the same graph, thus a legitimate account might be added in the attackers zone, and a fake account might be added to the trust zone.

In order to improve the methods presented in the literature, we propose a general process that can be applied on several social media types with little adaptation. This process detects sockpuppets and groups together fake accounts created by a same sock-

puppeteer, in order to stop all the attacks performed by a same user using different sockpuppet accounts.

In this chapter we present a global overview of the *SocksCatch* process, that aims at detecting and grouping sockpuppets in social medias. First, we introduce the overall process of *SocksCatch*. Then we propose a model of online social medias, in order to represent their common elements. Finally, we introduce an application of the *SocksCatch* process on the most significant collaborative project: english Wikipedia.

4.1 SocksCatch: Overall Process

Currently, as seen in section 2.3, many social medias have partial solutions for the detection and/or grouping of malicious accounts, but most of them need a manual validation. That means that the administrators of these sites have to check the accounts and their histories in order to take a decision to ban an account or not, thus wasting a lot of time and effort [Tsikerdekis and Zeadally, 2014a]. In addition, a manipulator may attack many victims before being banned.

To the best of our knowledge, there does not exist an automatic method that can at the same time detect malicious accounts and group together the sockpuppets created by a same sockpuppeteer.

Building on these conclusions we introduce *SocksCatch*, a complete process to detect and group sockpuppets automatically (shown in Fig. 4.1). This process is composed of three main phases. The first phase objective is the selection and pre-processing of data. The second phase objective is the detection of sockpuppet accounts using a machine learning algorithm. The third phase objective is the grouping of sockpuppet accounts created by a same user using a community detection algorithm.

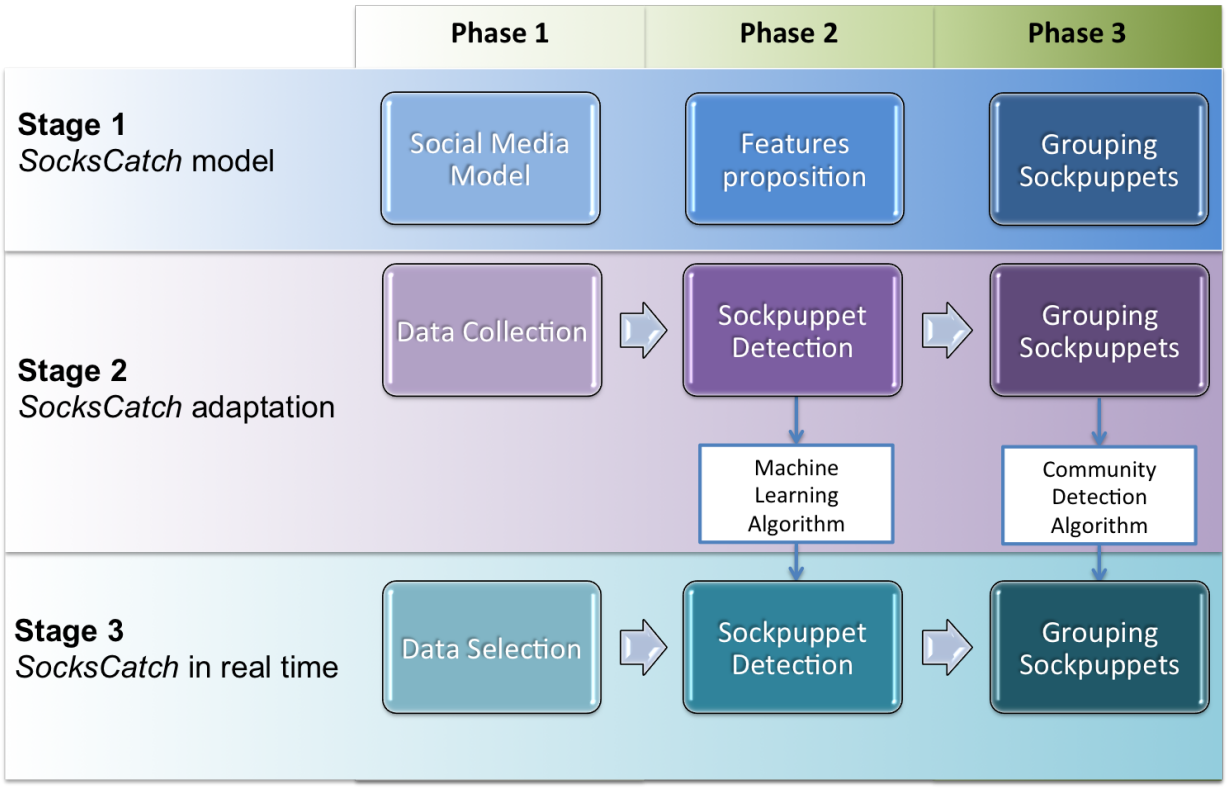

Figure 4.1 – SocksCatch's main stages and phases.

: Data Flow; : Algorithm provided as output/input.

As shown in figure 4.1, *SocksCatch* distinguishes three stages:

- First stage is the *SocksCatch* model, where we propose a general model of social media dedicated to the detection and grouping of sockpuppets. This stage is the generic part of sockscatch, *i.e.* it proposes features and models that exist in all social media although they may be instatiated differently.

- Second stage is *SocksCatch* adaptation, where we instantiate and evaluate *SocksCatch* model on a selected social media, in order to find the best machine learning algorithm for the second phase, and the best community detection algorithm for the third phase. This stage relies on a specific data collection to train and select the

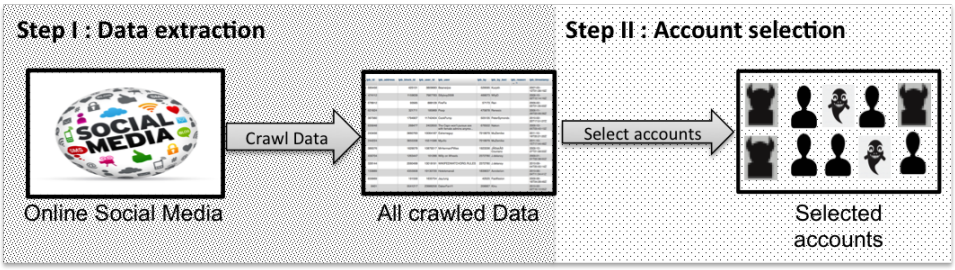

Figure 4.2 – SocksCatch's first phase's steps.

machine learning and community detection algorithms. This stage has to be done independently for each social media when the feature instantiation differs.

- Third stage is the deployment of *SocksCatch* in real time, where *SocksCatch* is applied online on a selected social media using the best learned machine learning algorithm and the best community detection algorithm founded on the evaluation of the second stage. In this stage, the data comes from the site activity, and the detection and grouping of sockpuppets is automated.

Let us note that in this thesis the first stage is proposed and the second stages is instantiated on a social media. The third stage (deployment) can only be done by social medias administrators to apply the findings of the two previous stages.

The first phase of *SocksCatch* is dedicated to the pre-processing of the data used in each stage. In the model stage, we represent our social medias model that will be used in the following phases in order to support *SocksCatch*. In adaptation stage, the first phase comprises two steps: data extraction from the site's history (I) and accounts selection (II) (Fig. 4.2). Their objective is to crawl a dataset with a specific number of accounts from a social media site in order to evaluate *SocksCatch* on real accounts and to check its accuracy. In real time stage the data is retrieved directly from the database of the selected social media directly.

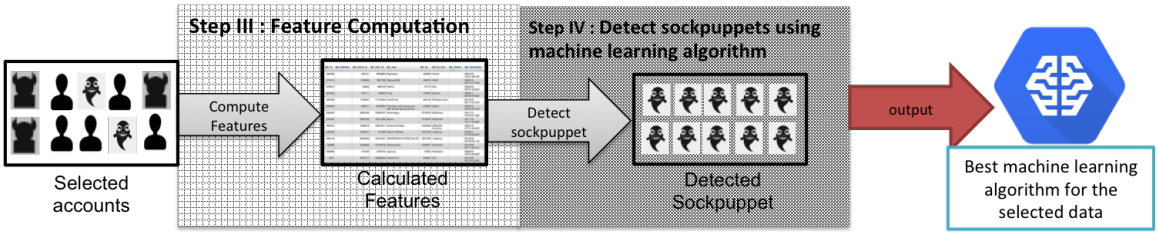

Figure 4.3 – SocksCatch's second phase's steps.

The second phase objective is the detection of sockpuppets. In model stage of this phase, we propose a set of general features that can be applied on different social media sites. These features should differentiate the behavior of sockpuppet from that of legitimate accounts. The adaptation stage of this phase comprises two steps: feature adaptation and computation (III) and detection of sockpuppet accounts using machine learning algorithms (IV) (Fig. 4.3). In the step III, we instantiate the proposed features on the dataset of a selected social media, then we compute them for each selected account. In the step IV, the computed features of each account are evaluated on several trained and tested machine learning algorithms in order to detect sockpuppets. The selection of the best machine learning algorithm for a particular site depends on the results of the evaluation applied to the dataset of the selected social media in the first phase.

The real time stage uses this trained algorithm to detect the sockpuppets.

The last phase objective is the grouping of sockpuppets. In model stage of this phase we propose to create a set of graphs to be used by community detection algorithms to group the sockpuppets created by the same user, and a set of features to represent the accounts behaviors. The adaptation stage of this phase comprises two steps: grouping based on actions in same pages (step V) and grouping based on accounts features (step VI) (Fig. 4.4). In the step V, a graph is created by relating each detected sockpuppet account with the pages where it performed actions. From this first graph another graph

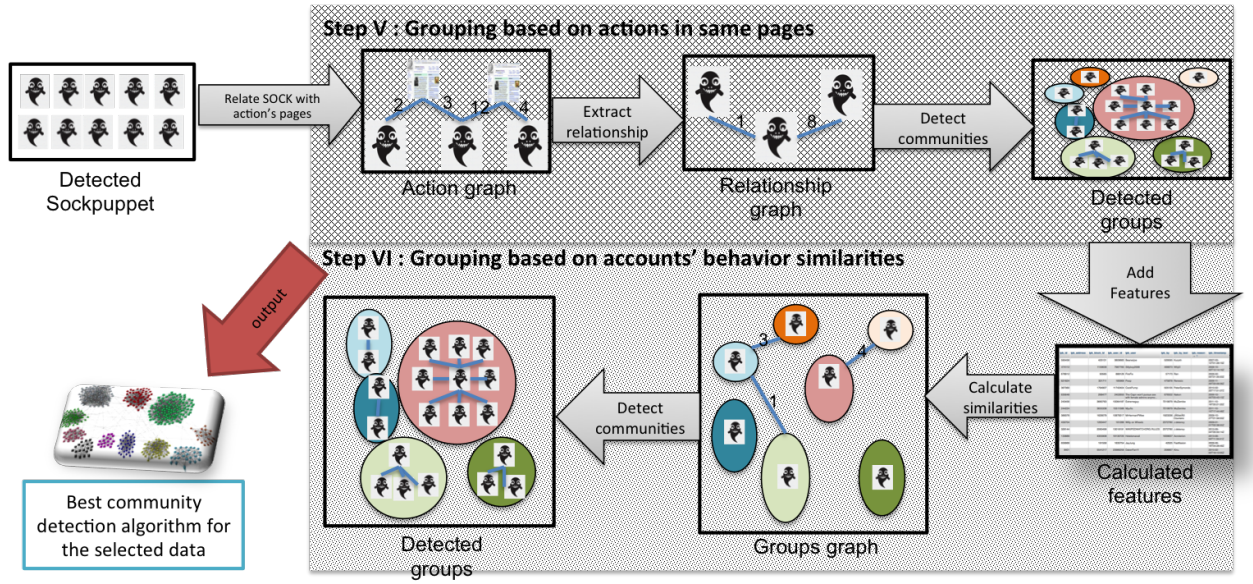

Figure 4.4 – SocksCatch's third phase's steps.

that links the accounts to each other is extracted. Then, sockpuppets are grouped from this extracted graph using a community detection algorithm. In the last step (VI), we improve the detected groups by relating the remaining single sockpuppets (non-grouped) to the detected groups based on their behavior similarities.

As in the case of the selection of machine learning algorithm, the best community detection algorithm is identified from the results of the experiment of *SocksCatch*.

The real time stage uses this algorithm and features to group the sockpuppets.

In the next section we define formally the model of social medias used to support the *SocksCatch* process in stage 1.

4.2 A model of social media (Stage 1, Phase 1)

We define an online social media as a tuple of the main elements that characterize the users and their interactions with the medium. Those were selected as abstractions of features existing in the majority of social media .

Definition [Online Social Media]: An Online Social Media OSM is a tuple $\{U, P, C, A\}$ such as :

- U is the set of user accounts.
- P is the set of pages.
- C is the set of categories.
- A is the set of actions.

In the following, we detail each of these sets:

Accounts: the set of existing user accounts is noted $U = \{u_1, ..., u_{|U|}\}$. Features are used to describe accounts. The set of all defined features is F, where $F = \{f_1, ..., f_{|F|}\}$.

The relation between accounts and features is represented by the two following functions.

Definition [Relation between accounts and features]: An account has a list of features. Therefore, each feature represents a part of the behavior of the account. The $uFeatures$ function gives the features value of an account.

$$uFeatures : UxF \mapsto \mathbb{R}$$

An account u_i is a virtual representation of a user in a particular OSM. Let us note that a single user can create several accounts.

Each set of accounts created by the same user forms a group. Thus, a group contains either one account or multiple accounts. We define the set of groups as GRP where $GRP = \{grp_1, ..., grp_{|GRP|}\}$.In the same way than single accounts, each group has a set of features, calculated from the features of its members.

The relation between accounts, features and groups is represented by the following functions.

Definition [Relation between accounts, features and groups]: An account belongs to one and only one group. Therefore, each group includes one or more accounts. The $uGroup$ function gives the group of an account.

$$uGroup : U \mapsto GRP$$

The list of accounts corresponding to a given group is obtained by the $grpAccounts$ function.

$$grpAccounts : GRP \mapsto 2^U$$

A group has a list of features that represent the behavior of its members. The $grpFeatures$ function gives the features of a group.

$$grpFeatures : GRPxF \mapsto \mathbb{R}$$

Example

If $uGroup(u_1) = grp_1$, $uGroup(u_2) = grp_1$ and $uGroup(u_3) = grp_2$, then the accounts u_1 and u_2 belong to a same group grp_1 and u_3 belong to the group grp_2.

Inversely, $grpAccounts(grp_1) = \{u_1, u_2\}$ and $grpAccounts(grp_2) = \{u_3\}$.

Pages: Pages are represented by $P = \{p_1, ..., p_{|P|}\}$.The pages correspond to the concrete spaces in OSM where all public information is exchanged in, *e.g.* a wall of an account

or of a specific group in Facebook, an article or a user discussion page in Wikipedia, image pages in Instagram, or profile pages in Twitter. Each page belongs to a specific category.

Categories: Categories are represented by $C = \{c_1, ..., c_{|C|}\}$.Each category c_i corresponds to a semantic category of which nature depends on the kind of OSM, *e.g.* set of walls or set of groups in Facebook, set of articles or set users discussions in Wikipedia, set of images in Instagram or set of users profiles in Twitter. The basic hypothesis is that different categories carry different weight in social medias, in terms of use and visibility.

The relation between pages and categories is represented by the two following functions.

Definition [Relation between pages and categories]: A page belongs to one and only one specific category. Therefore, each category includes several pages. The $pCategory$ function gives the category of a page.

$$pCategory : P \mapsto C$$

The list of pages corresponding to a given category is obtained by the $cPages$ function.

$$cPages : C \mapsto 2^P$$

Example

If $pCategory(p_1) = c_1$, $pCategory(p_2) = c_1$ and $pCategory(p_3) = c_5$, then the pages p_1 and p_2 belong to a same category c_1 and p_3 belongs to the category c_5. Inversely, $cPages(c_1) = \{p_1, p_2\}$ and $cPages(c_5) = \{p_3\}$

Actions: actions refer to the activity of accounts in social media pages such as writing/reporting a post or a comment in Facebook, a tweet in Twitter, or adding/deleting a photo in Instagram. The set of actions A is composed by the set of activities performed by each account on pages and might depend on the characteristics of each social media.

$$A = \bigcup_{i=1,j=1}^{|U|,|P|} A_{i,j}$$

$A_{i,j}$ is the set of actions produced by the account u_i on page p_j. Each of these sets is represented by $A_{i,j} = \{a_{i,j}^1,...,a_{i,j}^{|A_{i,j}|}\}$, an action $a_{i,j}^k$ being the k^{-th} action performed by account u_i on page p_j.

We complete these basic definitions with several functions that give the basic properties of the accounts and of the actions.

We include in the general model the time of events such as the creation of an account, the time of the first action and when an action occurs because they provide a temporal information about the actions. They are obtained by the 3 following functions (with $\mathcal{T}$ being the set of timestamps):

Definition [Account creation time]: The creation time of an account is obtained by the function

$$uRegTime : U \mapsto \mathcal{T}$$

Definition [Account first action time]: The time of the first action of an account is obtained by the function

$$uFirstActTime : U \mapsto \mathcal{T}$$

Definition [Action occurrence time]: The occurrence time of an action is obtained by the function

$$aTime : A \mapsto \mathcal{T}$$

In addition, the set of pages edited by an account (*i.e.* all the pages for which there exists an action performed by this account is defined as follows:

Definition [Edited pages by an account]: The set of edited pages by an account is obtained by the function

$$uPages : U \mapsto 2^P$$

Additionally, each action is described by a set of properties. Among the general properties of interest to characterize an account behavior, we represent the number of bytes added or removed by an action thanks to the function $aBytes$.

A second function, $aDel$, is used to indicate that an action has been reported after its public release. A *report* action is a general term to catch the possibility for users to notify an action as being malicious. As a matter of fact, this possibility exists in most of existing OSM using different forms (revert on Wikipedia, delete post on Facebook, report tweet on Twitter...).

Definition [Action properties]: The number of bytes added or removed by an action is given by:

$$aBytes : A_{ij} \mapsto \mathbb{Z}$$

The $aDel$ function returns 1 if an action has been reported, 0 otherwise.

$$aDel : A_{ij} \mapsto \{0, 1\}$$

The social media model represents users, accounts, pages, categories and their interactions. In the next sections, we build *SocksCatch* graphs based on this model that will be used in the phase 3 of stage 1.

4.2.1 Action graph

Using the previous definitions, we can represent a social media as a graph that links accounts and pages through actions.

A graph is broadly defined as $G = (V, E)$ where V is the set of nodes (also called vertices) and E is a set of edges that connect nodes. In our model, the nodes are the accounts and pages, and the edges represent the actions between these nodes. Edges are weighted, that is, a value is assigned to them. A strong weight then indicates a high intensity relation.

In order to represent the global activity of accounts on pages, we define a weighted bipartite graph called **action graph**. In terms of notation, the action graph is thus defined as $G^{act} = (V^{act}, E^{act})$ where G^{act} stands for the whole platform, V^{act} stands for the set of nodes, and E^{act} for the set of all edges. The edges of this action graph are the interactions between an account node u_i and a "page" node p_j. Hence, $V^{act} = U \cup P$ and $U \cap P = \emptyset$. Therefore the edges represent relations between nodes in sets U and P. We note $e^{act}_{i,j} \in E^{act}$ an edge that connects the account u_i with the page p_j. The edges are extracted from the actions A. In order to represent the period when an account has been active on a page, the weight $w^{act}_{i,j}$ of $e^{act}_{i,j}$ is calculated as the average of the timestamps of each action of u_i on p_j.

$$w^{act}_{i,j} = \frac{\sum_{a \in A_{i,j}} aTime(a)}{|A_{i,j}|}$$

Algorithm 1 shows the construction of this graph. For all accounts (line 3), and for each page visited by this account (line 4), the attributes related to the actions done by the user (number of actions, time of these actions) is computed. Then, the algorithm sets the average of time of actions made by account in the same page as the weight and

creates the edges between the users and the pages.

Algorithm 1 Create Action graph

```
1:  function CREATEACTIONGRAPH(U)
2:      G^act = newBipartiteGraph ();
3:      for each u_i ∈ U do
4:          for each p_j ∈ uPages (u_i) do
5:              sumActionTime [p_j] = 0;
6:              for each a_k ∈ A_i,j do
7:                  sumActionTime [p_j] += aTime (a_k);
8:              end for
9:              w_i,j = sumActionTime [p_j] / length (A_i,j);
10:             G^act.addEdge (u_i, p_j, w_i,j);
11:         end for
12:     end for
13:     return G^act;
14: end function
```

Fig. 4.5 represents an example of an action graph formed by five sockpuppets (u_1, u_2, u_3, u_4 and u_5) and six pages (p_1, p_2, p_3, p_4, p_5 and p_6) where these sockpuppets performed actions.

Example

Let us assume the following:

u_1 performed two actions in p_1 at "2005 – 07 – 23 22 : 42 : 15" and "2005 – 07 – 23 22 : 42 : 01".

u_2 performed one action in p_1 at "2005 – 07 – 24 18 : 58 : 47" and another one in p_2 at "2005 – 07 – 24 18 : 58 : 22".

u_3 performed one action in p_3 at "2005 – 08 – 26 12 : 34 : 56".

u_4 performed one action in p_4 at "2005 – 11 – 08 11 : 56 : 45".

u_5 performed two actions in p_5 at "2005 – 06 – 16 20 : 15 : 11" and "2005 – 06 – 16 20 : 03 : 17" and two actions in p_6 at "2005 – 06 – 15 16 : 29 : 21" and

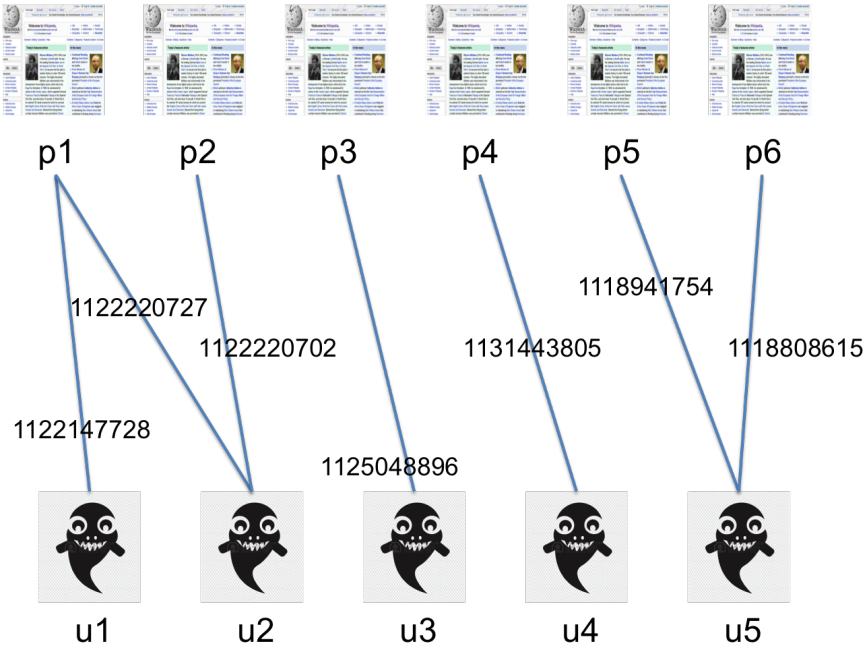

Figure 4.5 – An example of action graph with relations between sockpuppets and pages where they performed actions.

"2005 – 06 – 14 21 : 51 : 09".

As described above, the algorithm adds an edge between the account and the page where he performed action and it adds the average of timestamp of actions performed in this page as weight for the edge, for example u_1 performed two actions in the same page p_1 at "2005 – 07 – 23 22 : 42 : 15" and "2005 – 07 – 23 22 : 42 : 01". Hence, the average time is 2005 – 07 – 23 22 : 42 : 08 and the weight in seconds between u_1 and p_1 is:

$$w_{1,1}^{act} = \frac{1122147735 + 1122147721}{2} = 1122147728$$

4.2.2 Relationship Graph

From the action graph, we extract another weighted undirected graph to be used in community detection algorithms called **relationship graph**. In terms of notation, the relationship graph is thus defined as $G^{rel} = (V^{rel}, E^{rel})$ where G^{rel} stands for the whole graph, V^{rel} stands for the set of nodes, and E^{rel} for the set of all edges. This graph is used to group the accounts using a community detection algorithm. It represents the relationship between accounts based on the actions they performed in the same pages.

On the contrary to the action graphs, an edge is a symetric relation between two nodes of same nature. In this graph, the edges link only the accounts (U) between each others: $V^{rel} = U$. An edge $e^{rel}_{i,j}$ is created between $u_i \in U$ and $u_j \in U$ if there exists at least one page in which both accounts have contributed:

$$\forall e^{rel}_{i,j} \in E_{rel} \,/\, \exists p_k \in P \,/\, |A_{i,k}| > 0 \;\wedge\; |A_{j,k}| > 0 \tag{4.1}$$

The weight of each edge is obtained from a calculation on actions attributes to represent the degree of relation between the accounts u_i and u_j. This weight is the inversed euclidean distance between the normalized average of the actions' time of u_i and u_j. Hence, the edge between two accounts is "strong" if they made their actions in a short time window.

The euclidean distance is calculated as follows:

$$d(u_i, u_j) = \sqrt{\sum_{p_k \in uPages(u_i) \cap uPages(u_j)} (wn^{act}_{i,k} - wn^{act}_{j,k})^2} \tag{4.2}$$

where $wn^{act}_{i,k}$ and $wn^{act}_{j,k}$ are respectively the normalized weight of edges between the account u_i with page p_k and the account u_j with page p_k in the **action graph**, and they are calculated as follows:

$$wn_{i,k}^{act} = \frac{w_{i,k}^{act} - \min(W^{act})}{\max(W^{act}) - \min(W^{act})} \tag{4.3}$$

where $W^{act} = (w_{1,1}^{act}, ..., w_{N_u,N_p}^{act})$.

The inverted euclidean distance is then calculated as follows in order to make the smallest distance as the highest weight:

$$w_{i,j}^{rel} = \frac{1}{1 + d(u_i, u_j)} \tag{4.4}$$

Example

Fig. 4.6 represents the example of the extracted relationship graph between the five sockpuppets mentioned above where only u_1 and u_2 performed actions in the same page. In order to compute the weight of the edge between u_1 and u_2, we calculateb the following values :

The normalized weight of action's graph edges, where $\max(W^{act})$ = 1131443805 and $\min(W^{act})$ = 1118808615, is:

$$wn_{1,1}^{act} = \frac{1122147728 - 1118808615}{1131443805 - 1118808615)} = 0.264$$

$$wn_{2,1}^{act} = \frac{1122220727 - 1118808615}{1131443805 - 1118808615)} = 0.270$$

The euclidean distance between the normalized weights of u_1 and u_2 is:

$$d(1,2) = \sqrt{(0.264 - 0.270)^2} = 0.006$$

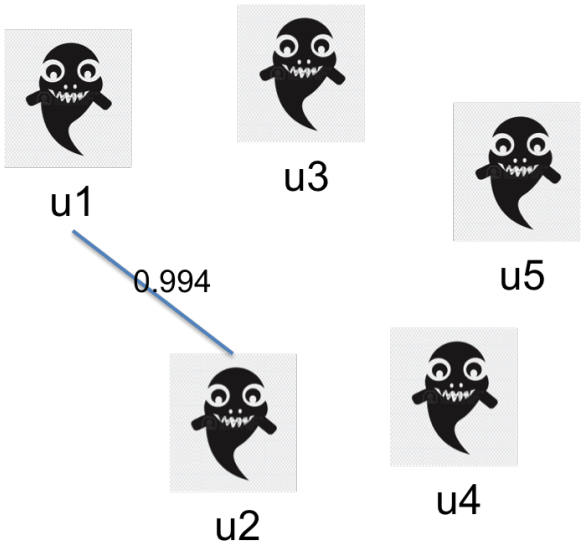

Figure 4.6 – An example of relationship graph between five sockpuppets based on their actions made in the same pages.

The inverted euclidean distance of u_1 and u_2 is:

$$w_{1,2}^{rel} = \frac{1}{1+0.006} = 0.994$$

4.2.3 Groups Graph

By giving the relationship graph to the community detection algorithms, the core groups detected from this relationship graph are detected. Using these groups we create another weighted undirected graph that relate the groups based on features similarities called **groups graph**. In terms of notation, the groups graph is thus defined as $G^{gr} = (V^{gr}, E^{gr})$ where G^{gr} stands for the whole graph, V^{gr} stands for the set of nodes, and E^{gr} for the set of all edges. This graph represents the relationship between groups based on the similarity between the features of each group.

In groups graph, an edge is a relation between two nodes of the same nature. In this graph, the edges link the groups (GRP) that have one member (account) with other

singleton groups or with groups that have multiple members. That means that no edge may be created between groups that already contain multiple members: $V^{gr} = GRP$. An edge $e^{gr}_{i,j}$ is created between $grp_i \in GRP$,and $grp_j \in GRP$ if there exists at least one group with only one account:

$$\forall e^{gr}_{i,j} \in E^{gr} \ / \ \exists \ grp_i \in GRP \wedge grp_j \in GRP \tag{4.5}$$
$$if(|grpAccounts(grp_i)| = 1 \ \wedge \ |grpAccounts(grp_j)| \geq 1)$$
$$\vee \ if(|grpAccounts(grp_i)| \geq 1 \ \wedge \ |grpAccounts(grp_j)| = 1)$$

The weight of each edge is obtained from a calculation on groups' features to represent the degree of similarity between grp_i and grp_j. This weight is the inversed euclidean distance between the features of grp_i and grp_j. Hence, the values of features of each group are the average of the features of the group's members.

The euclidean distance is calculated as follows:

$$dg(grp_i, grp_j) = \sqrt{\sum_{k=1}^{N_f} (grpFeatures(i,k) - grpFeatures(j,k))^2} \tag{4.6}$$

where $grpFeatures(i,k)$ and $grpFeatures(j,k)$ are respectively the k^{th} feature of the group grp_i and the group grp_j

The inverted euclidean distance is then calculated as follows in order to obtain the smallest distance as the highest weight:

$$w^{grp}_{i,j} = \frac{1}{1 + dg(grp_i, grp_j)} \tag{4.7}$$

Fig. 4.7 represents the example of a groups graph. It contains four groups where only grp_2 and grp_4 are related with an edge of 0.026 as weight.

The action graph and the relationship graph are two different points of view on the OSM. The first characterizes the behavior of the different accounts in the social media,

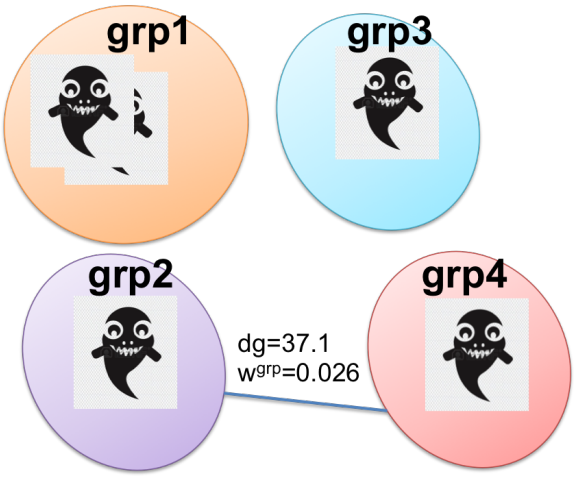

Figure 4.7 – Groups graph containing four groups where only grp_2 and grp_4 are related with an edge of 0.026 as weight.

while the second reifies how this behavior links the accounts with each other. The groups graph shows if there is a similarity in the behavior of different groups.

In the next section, we present the most used collaborative project, english Wikipedia. We then show how we mapped our proposed social media model on Wikipedia, in order to use its data for adaptation stage where we will instantiate and evaluate Sockscatch.

4.3 SocksCatch's application on Wikipedia's sockpuppets (Stage 2, Phase 1)

As seen in chapter 2, collaborative projects present several advantages to test and apply our approach, such as the big number of collaborators and manipulators at the same time. The collaborative project with the highest number of collaborators is English Wikipedia. Furthermore, Wikipedia offers the ability to collect all logs and history of interactions of all the accounts publicly.

Another important feature is that it have common features with other social medias

like commenting on other user's page and writing in a personal way.

Wikipedia contributors are expected to edit the pages using only one account. Hence, the improper use of multiple Wikipedia user accounts is not allowed based on Wikipedia rules.

Currently, in Wikipedia, any user that finds a malicious contribution and considers an account as being a sockpuppet can open an investigation page and notify the 'check user', a privileged set of users who have access to the IP-address of all accounts and contributions, in order to discuss with others whether this account is malicious or not. The page's administrators and 'check users' will then check manually if this account is a real sockpuppet or not by studying the account's behavior and by checking the IP-address for some cases like sleeper[1] accounts. If the account is considered as being malicious, the administrator mentions in the ban commentary the main account of the user (sockpuppeteer).

All this data with the reason of ban is publicly available[2]. This makes a verifiable use case to test algorithms/features to detect this kind of manipulation in OSMs.

In order to evaluate our proposed model of *SocksCatch* we instantiate it on English Wikipedia data.

The tuple {U, P, C, A} of social medias described in section 4.2 is mapped on Wikipedia as following :

- Accounts U are the registered accounts and the IP address of unregistered accounts.
- Categories C are namespaces such as set of articles and set of discussion pages (table 2.2 : page 16).

[1] Accounts that are created as a backup for a malicious user.

[2] through `https://en.Wikipedia.org/w/index.php?title=Wikipedia:sockpuppet_investigations/Cases/Overview&offset=&limit=500&action=history`

- Pages P are articles and discussion pages.

- Actions A are called contributions such as a modification in an article and a comment in a discussion page.

As mentioned above, the data collection phase of *SocksCatch* is created for the adaptation stage, because the collected data will be used to evaluate the process.

This phase is composed of two steps, where in the data extraction step we collect all the required data for our experiment from English Wikipedia, then in the accounts selection step we select from the collected data the accounts used to evaluate *SocksCatch*.

4.3.1 Data extraction

Until April 2015, the total amount of data contained in English Wikipedia was about 10 TB uncompressed[3], so that it is not efficient to retrieve the whole data and process it. Hence, we start by only crawling the accounts (blocked and legitimate), without their activities history, from February 2004 to April 2015.

There are several reasons to block an account (such as spam, vandalism, or threats), and each reason implies a limited or unlimited time of blocking. For our model, we decide to study the accounts that are blocked for sockpuppet reason and for infinite time. The total number of accounts blocked for this reason amounts to 118 414 accounts. When an account is blocked because of sockpuppetry, the administrators state to which original account the blocked one was linked. This allows to create the real clustering Cl' (groups) of accounts created by the same user.

[3] 100 GB compressed using 7-Zip

4.3.2 Account selection

For this study, the 118 414 sockpuppets are grouped and referred to their sockpuppeteer, the first malicious account created by the user. 12 088 groups, are obtained ranging from 2 to 557 members. Among those groups, 5 000 sockpuppet accounts that are in a group with more than 3 sockpuppets are randomly selected, in order to be sure that these accounts are for users who tried many times to manipulate. These 5 000 sockpuppets are filtered to keep only the sockpuppets that have at least a contribution in one namespace. The final number of sockpuppets used for this phase is 4 869 sockpuppets and they are members of 1 402 groups.

Example

An example of such groups is that of the owner of the account *"YCWebCrawler"*, who has created 4 supplementary accounts (*Fixthemistakes*, *Wordslayer*, *Wikidirt* and *PageOneEditor*) to manipulate on Wikipedia. All these accounts are blocked as sockpuppet accounts in English Wikipedia because they were created solely for the deletion of material on the *Dennis L. Montgomery* page. All these accounts use similar language about protecting family members in summaries.

After selecting sockpuppet accounts, we need to select also active accounts, in order to train the machine learning algorithms on the acceptable behaviors. We randomly select 5 000 active accounts, which fulfill two criteria: they have been active for more than one year and they have made at least one contribution on Wikipedia. These criteria were selected in order to have a high probability that these accounts are not sockpuppets undiscovered yet.

Once these accounts were found, we crawl all related data in the thirty namespaces mentioned in table 2.2 page 16. Furthermore, we also retrieved all information about the namespaces' pages, in order to know the behavior of other accounts towards the account's modifications. This allows for example to automatically detect whether the modifications done by a specific account are kept by other users or reverted.

The final database used in this study thus contains 71 GB of information.

The list of selected accounts with their status (Sockpuppet or Active) is available for researchers [4].

4.4 Conclusion

We have presented in this chapter the three stages and phases of *SocksCatch*. Then we defined a model for social medias. Using this model, we describe how to build a bipartite graph that represents the activities of accounts on social media. In addition, we presented a relationship graph that relates accounts based on their activities performed in the same pages. Furthermore, the groups graph that relate the group of accounts based on their behavior similarity is detailed. Finally, the use case and our collected data that is used to evaluate *SocksCatch* is introduced.

In the next chapter, we develop the sockpuppet detection phase. In this phase several features are proposed in order to differentiate between a sockpuppet and an active account, then we show how these features can be adapted on English Wikipedia. Finally, the adapted features on English Wikipedia are evaluated using different machine learning algorithms.

[4] `http://pagesperso.litislab.fr/~jsaunier/EnWikipedia_users_list.txt`

CHAPTER 5

SOCKPUPPET DETECTION

As we have seen in the previous chapter, *SocksCatch* is composed of three phases: preparation phase, sockpuppets detection phase and sockpuppets grouping phase. In addition, *SocksCatch* distinguishes three stages: model stage, adaptation stage and real time stage.

In this chapter, we develop the second phase of *SocksCatch* : detection of sockpuppets using machine learning algorithms.

In the model stage, we propose a set of general features, in order to differentiate the sockpuppet and legitimate account's behaviors on social medias. These features can be adapted and applied on different OSMs.

In the adaptation stage, we adapt and instantiate the proposed features on English Wikipedia. Then, we introduce the experimental part: we present the metrics that evaluate the computed features on six trained and tested machine learning algorithms.

The machine learning algorithm with the highest accuracy is selected in order to be used in the real time stage of sockpuppets detection phase.

5.1 Sockpuppet Detection's Features (Stage 1, Phase 2)

In order to differentiate between a sockpuppet and a legitimate account, we propose several features F^{detect}, where $F^{detect} \subset F$ and F is the set of features (definition 4.2, page 73). These features focus on three main axes: action behavior, other accounts behavior toward an account's actions, and account characteristics.

Concerning the action behavior, we assume that the goal of a sockpuppet is to manipulate contributions on a specific topic or category of an OSM. Thus, the number of actions performed by an account in specific categories is interesting. We have also observed that some fake accounts try to remove specific pieces of information or to add misinformation. The average of bytes added and/or removed in each action and the average of the actions of each account in the same page are then promising features.

The behavior of the other accounts towards the studied account is also an indication. The number of reported actions of an account is another important feature because some other users can notify malicious contributions.

Finally, the account characteristics can be a sign of malicious behaviour. While in general, legitimate users register and contribute directly (the registration being done in order to contribute), some manipulators create many accounts at the same time and leave most of them sleeping until the active account is blocked. In order to detect these cases, we propose a feature that subtracts the time of the first action performed and the time of account's registration.

These features will be instantiated on a social media platform, in order to be computed for each account. Then the list of accounts with their calculated features will be used to

train and test the machine learning algorithms.

The proposed features are detailed below:

Number of account's actions in different OSM categories:

This is a set of features to represent the activity of an account in each of the OSM categories to detect the focus of the selected account. Hence, each category correspond to a feature, using the number of actions performed in these categories.

If we take Facebook as an example, the number of added comments in Facebook pages as a feature, and the number of added comments in friends posts as another feature is taken into account.

The features $ActionNb(u_i, c_k)$ are calculated as follows:

$$\forall c_k \in C, \forall u_i \in U,$$

$$\texttt{ActionNb}(\texttt{u}_\texttt{i}, \texttt{c}_\texttt{k}) = |A_{u_i,c_k}| \; with$$

$$A_{u_i,c_k} = \{a_{i,j} | a_{i,j} \in A \text{ and pCategory}(p_j) = c_k\}$$

Average of bytes added or removed in each action:

The goal of a fake account may also be detected through its addition/removal behavior. Hence, we propose two features to compute the average of bytes added by writing actions or/and the average of the bytes removed by writing action.

Let $A^+_{u_i}$ be the set of actions consisting in adding bytes and $A^-_{u_i}$ be the set of actions consisting in removing bytes, they are defined as:

$$\forall u_i \in U,$$

$$A^+_{u_i} = \{a_{i,j} | a_{i,j} \in A_{u_i} \text{ and aBytes}(a_{i,j}) > 0\}$$

$$A^-_{u_i} = \{a_{i,j} | a_{i,j} \in A_{u_i} \text{ and aBytes}(a_{i,j}) < 0\}$$

The feature $Addition(u_i)$ and $Removal(u_i)$ are then:

- if $A^{+}_{u_i} = \emptyset$, $Addition(u_i) = 0$.
- if $A^{+}_{u_i} \neq \emptyset$,

$$Addition(u_i) = \frac{\sum_{a \in A^{+}_{u_i}} aBytes(a)}{|A^{+}_{u_i}|} \tag{5.1}$$

- if $A^{-}_{u_i} = \emptyset$, $Removal(u_i) = 0$.
- if $A^{-}_{u_i} \neq \emptyset$,

$$Removal(u_i) = \frac{\sum_{a \in A^{-}_{u_i}} aBytes(a)}{|A^{-}_{u_i}|} \tag{5.2}$$

Focus of an account's actions:

Some fake accounts try many times to change or to add the same (mis)information in the same page. They focus on the same pages, because they have a specific objective to manipulate an information or idea. This feature is calculated as a ratio between the number of actions and the number of pages modified by an account.

The feature $Focus(u_i)$ is defined as:

$\forall u_i \in U$,

- if $|uPages(u_i)| = \emptyset$, $Focus(u_i) = 0$.
- if $|uPages(u_i)| \neq \emptyset$,

$$Focus(u_i) = \frac{|A_{u_i}|}{|uPages(u_i)|} \tag{5.3}$$

Number of reported actions of an account: As mentioned in chapter 2, most of OSM platforms give the ability for the accounts to report or to delete other accounts action if they are considered as malicious or inappropriate. Using this reporting system the OSM platforms benefit from the contribution of their users to detect the malicious actions,

which is very helpful because most of platforms have a huge number of users, and in addition the human detection is reliable. In this feature, we compute the number of reports performed by other accounts toward the selected account.

Let A_{u_i} be the set of actions of an account and $A_{u_i}^{rep} \subset A_{u_i}$ the set of its reported actions i.e,

$$\begin{aligned} &\forall u_i \in U, \\ &\quad A_{u_i} = \bigcup_{j=1}^{N_p} A_{i,j} \\ &A_{u_i}^{rep} = \{a_{i,j} | a_{i,j} \in A_{u_i} \text{ and } \mathrm{aDel}(a_{i,j}) = 1\} \end{aligned}$$

The feature $Reported(u_i)$ is then:

$$Reported(u_i) = |A_{u_i}^{rep}| \tag{5.4}$$

Delay between account's creation and first action: The policies of most OSM platforms do not allow the creation of multiple accounts by the same user [Elizabeth, 2015; Twitter, 2017], but at the same time they can be created technically. The manipulators benefit from this technical issue to create many accounts at the same time, and they leave most of them sleeping until their active account is blocked, where they begin using a backup one. We propose this feature that calculates the difference between the creation time of an account and its first action, in order to detect the sleeping accounts.

The feature $Delay(u_i)$ is defined as:

$$Delay(u_i) = uFirstActTime(u_i) - uRegTime(u_i) \tag{5.5}$$

In the next section, we present the adaptation stage of sockpuppet detection phase, where we start by the adaptation of the proposed features on Wikipedia environment, then we compute and evaluate the adapted features using machine learning algorithms.

5.2 Sockpuppets Detection Adaptation (Stage 2, Phase 2)

The objective of this section is to evaluate the efficiency of the proposed features in the detection of sockpuppets on english Wikipedia.

This phase of *SocksCatch* adaptation is composed of two main steps. In first step, we adapt and compute the proposed features in *SocksCatch* model stage on english Wikipedia. Then, in second step, we evaluate these computed features using machine learning algorithms, in order to know if the proposed features are efficient in the case of wikipedia and which is (are) the best algorithm(s) that should be used in *SocksCatch*'s real-time stage.

5.2.1 Features Adaptation on Wikipedia

As mentioned in the previous chapter, we applied *SocksCatch* on english Wikipedia. Hence, the set of features F^{detect} proposed in second phase of *SocksCatch*'s model stage is instantiated in order to be adapted and computed on Wikipedia data.

The adapted features that detect the behavior of the accounts on Wikipedia are described below.

Number of account's contributions by namespaces:

These features are instantiated from "Number of account's actions in different OSM categories". Six features are created based on the following namespaces: article, article discussion, user page, user discussion page, project namespace and other (all other namespaces, mentioned in table 2.2 page 16, are combined under one feature, because they are rarely used).

Average of bytes added and removed in all the contributions:

As mentioned in the section 5.1 and more precisely in the description of the feature "Average of bytes added or removed in each action", the goal of a manipulator may be detected sometimes from its addition/removal behavior.

In Wikipedia, we can apply the two original features, where $A^+_{u_i}$ is the set of bytes added in each contribution and $A^-_{u_i}$ is the set of bytes removed in each contribution.

Example

For example, let say that an account u_3 performed 5 contributions in different articles, where it added respectively 400, 600, 100 bytes of information in contributions a_1, a_2, a_5 and it removed respectively 200, 600 bytes of information in contributions a_3, a_4. The values of features $Addition(u_3)$ and $Removal(u_3)$ are then:

$$Addition(u_3) = \frac{400+600+100}{3} = 550\ bytes$$

$$Removal(u_3) = \frac{200+600}{2} = 400\ bytes$$

Average of contribution in the same articles:

In this feature, the "focus of an account's actions" is instantiated to one feature where the average number of contributions in the same article for each account is calculated.

Example

For example, let say that an account u_{12} contributes 5 times in article p_1 and 2 times in article p_2. The value of feature $Focus(u_{12})$ is then:

$$Focus(u_{12}) = \frac{5+2}{2} = 3.5$$

Number of revert after each contribution in the articles:

In collaborative projects, each page or article is managed by many contributors. It is the reason why we consider that some malicious contributions will be reverted directly by another contributor.

Since in Wikipedia the articles are the main namespace, we decided to instantiate the feature "number of reported actions of an account", described in section 5.1, to the number of direct revert after each contribution in an **article** without taking into consideration the revert of contributions in other namespaces because the manipulation that should be deleted occurs on articles and not on other namespaces.

Example

For example, an account u_4 contributes 20 times in Wikipedia pages, but had 10 out of 20 contributions reverted ($|A_{u_4}^{rep}|$). The value of feature $Reported(u_4)$ is then:

$$Reported(u_4) = 10$$

Delay between the time of account's registration and his first contribution:

This feature is the same as proposed in the process, where it can detect the accounts created but not used yet.

The 11 obtained features are summarized in table 5.1.

In the next section, we present the evaluations metrics used in order to evaluate the second phase of *SocksCatch*'s adaptation stage.

5.2.2 Evaluation Metrics

The second phase of *SocksCatch* uses machine learning algorithms. Thus before evaluating this phase at adaptation stage, we present the metrics used to evaluate their results in order to select the best machine learning algorithm that should be used in real time stage.

We first show the model accuracy n and then use the confusion matrix shown in Table 5.2. This matrix is used to visualize the performance of the different algorithms using the following metrics:

$$TruePositiveRate(TPR) = \frac{TP}{TP + FN} \tag{5.6}$$

Table 5.1 – a brief description of the instantiate features adapted on Wikipedia data

Feature's title	Adapted features F^{detect} for account u_i
Number of account's actions in different OSM categories	$ActionNb(u_i, ns0)$: number of contributions in the articles $ActionNb(u_i, ns1)$: number of contributions in the discussion of articles $ActionNb(u_i, ns2)$: number of contributions in the user page $ActionNb(u_i, ns3)$: number of contributions in the discussion of user page $ActionNb(u_i, ns4)$: number of contributions in the page that contains informations about Wikipedia $ActionNb(u_i, nsOthers)$: number of contributions in all the rest of namespaces
Average of bytes added or removed in each action	$Addition(u_i)$: average of bytes added in all the contributions $Removal(u_i)$: average of bytes removed in all the contributions
Focus of an account's actions	$Focus(u_i)$: average of contribution in the same articles
Number of reported actions of an account	$Reported(u_i)$: number of revert after each contribution in the articles
Delay between account's creation time and first action	$Delay(u_i)$: time in seconds between the time of the registration in Wikipedia and the time of the first contribution

This metric indicates the rate of truly detected sockpuppet.

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \tag{5.7}$$

This metric indicates the rate of falsely detected sockpuppet.

$$Precision = \frac{TP}{TP + FP} \tag{5.8}$$

This metric indicates the fraction of returned cases that are valid sockpuppet cases.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.9}$$

Table 5.2 – Table representing the confusion matrix used to evaluate the efficiency of our proposed method.

	Predicted sockpuppet	Predicted Legitimate Accounts
Sockpuppet	True Positive (TP)	False Negative (FN)
Legitimate accounts	False Positive (FP)	True Negative (TN)

This metric indicates the fraction of true positives and true negatives returned over the total number of cases.

$$F - measure = \frac{2 * Precision * TPR}{Precision + TPR} \tag{5.10}$$

This metric indicates the test of a model's accuracy that combines TPR and precision.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.11}$$

Matthews correlation coefficient (MCC) metric indicates a balanced measure which can be used even if the classes are of very different sizes.

$$\text{Area under ROC curve (AUC)} \quad = \frac{TPR - FPR + 1}{2} \tag{5.12}$$

Area under a ROC curve (AUC) estimates the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance.

In the next section, we present the experiment realized on the computed features in order to evaluate the accuracy of sockpuppet's detection using machine learning algorithms.

5.2.3 Sockpuppet's detection: experiment and results

This experiment uses the dataset that comprises 10 000 accounts, half blocked accounts and half legitimate accounts.

Php scripts that work on raw pages have been developed in order to calculate all the instantiated features values for each account, to pre-process the data, to use them in training and testing of machine learning algorithms.

Supervised learning is used in order to detect sockpuppet accounts. The relevance of the selected features is evaluated through several machine learning algorithms in order to prove the accuracy and robustness of the proposed method.

The following machine learning algorithms are used to evaluate the efficiency of the detection of sockpuppets phase:

- Support Vector Machine (SVM) [Cortes and Vapnik, 1995].
- Random Forest (RF) [Breiman, 2001].
- Naïve Bayesian (NB) [Robertson and Jones, 1976].
- K-Nearest Neighbor (KNN) [Altman, 1992].
- Bayesian Network (BN) [Pearl, 1985].
- Adaptive Boosting (ADA) [Freund and Schapire, 1995].

Let us note that all these algorithms are considered ideal for binary classification [Witten et al., 2016], as it was the case in our study.

The dataset was randomly split to 2/3 (6 666 accounts) for learning and validation and 1/3 (3 334 accounts) for testing. We should note that the 6 666 accounts are members of groups different from the groups of 3 334 accounts. We separate the data in this way, in order to be sure that while testing our dataset, the machine learning algorithm will not benefit from a similar behavior learned from accounts in same group.

In order to detect the best parameters of the machine learning algorithms, the 2/3 of the dataset was validated using ten-fold cross-validation for each algorithm. Then, the best detected parameters' values of each algorithm are set for the testing phase.

During validation, the 6 666 accounts are split randomly into ten folds where each fold contains 667 accounts; nine folds (6003 accounts) of them are used to build a model while one fold (663 accounts) is used for testing the model. Then we record the values of the metrics described in section 5.2.2. This process is repeated ten times and each time we used a different seed for splitting the dataset. The average of the ten recorded values of each metric is calculated at the end.

We used ten-fold cross-validation because it has been previously proven to produce highly accurate estimates in terms of how models (and as a result our overall method) would perform in previously unseen data [Hall et al., 2011].

Weka[1] toolbox is used to experiment the features, where the metrics described in the section 5.2.2 are used to quantify the quality of prediction.

The obtained results are shown in the table 5.4, table 5.5 and Fig. 5.1. These results are discussed in the following paragraphs.

Machine Learning Algorithms optimization

All the results are obtained after the optimization of the parameters of each classifier on Weka using ten-fold cross-validation method.

Table 5.3 presents the parameters' values that give the best accuracy of the selected machine learning algorithm while the validation of our model.

The best accuracy is obtained for Random Forest when we set the number of trees to be generated as 100, the maximum depth of the tree as 0 (for unlimited) and we

[1] `http://weka.wikispaces.com`

Table 5.3 – Table presenting the parameters' values that give the best accuracy of the selected machine learning algorithm while the validation of our model.

Machine Learning Algorithm	Weka Parameters
SVM	Cost = 5 Kernel = Poly Kernel
Random Forest	# of Trees = 100 Maximum depth = unlimited
Naïve Bayesian	useSupervisedDiscretization = True
K-Nearest Neighbor	K=1 Distance Weighting = Weight by 1-distance
Bayesian Network	All default values
Adaptive Boosting	# of iterations = 30

keep the default value of the number of randomly chosen attributes as 0 which mean $int(log_2(\#predictors)+1)$ is used. For the remaining classifiers, the best accuracy is obtained when the number of iterations to be performed is 30 for Adaptive Boosting, the number of neighbors to use (K) is 1 and the distance weighting method used is $Weight\ by\ 1-distance$ for KNN, the cost parameter C is 5 for SVM with Poly kernel. We set $useSupervisedDiscretization$ as $True$, to convert numeric attributes to nominal ones for Naïve Bayesian and we keep the default values of the parameters of Bayesian Network.

5.2.3.1 Machine Learning Algorithms comparison

Table 5.4 compares the precision of our proposed model over different machine learning algorithms, It shows that an accuracy over 81% is obtained with five algorithms : Random Forest 91%, Adaptive Boosting 86%, Bayesian Network 82.1%, Naïve Bayesian 82%, and K-Nearest Neighbor 81.5%.

The results of the remaining algorithm are also good results, with SVM at 73.7%.

Table 5.5 and Fig. 5.1 compares the different values of our proposed model metrics between many machine learning algorithms. It shows that TP Rate, Precision, Recall

Table 5.4 – Table comparing the accuracy percentage of our proposed process over different machine learning algorithms.

Machine Learning Algorithm	Sockpuppet's Detection Phase Accuracy
SVM	73.7%
Random Forest	91%
Naïve Bayesian	82%
K-Nearest Neighbor	81.5%
Bayesian Network	82.1%
Adaptive Boosting	86%

and F-Measure using Random Forest algorithm are 0.911, and that the MCC and AUC are respectively 0.824 and 0.961. The AUC using Naïve Bayesian give 0.887, 0.801 using KNN and 0.939 using Adaptive Boosting.

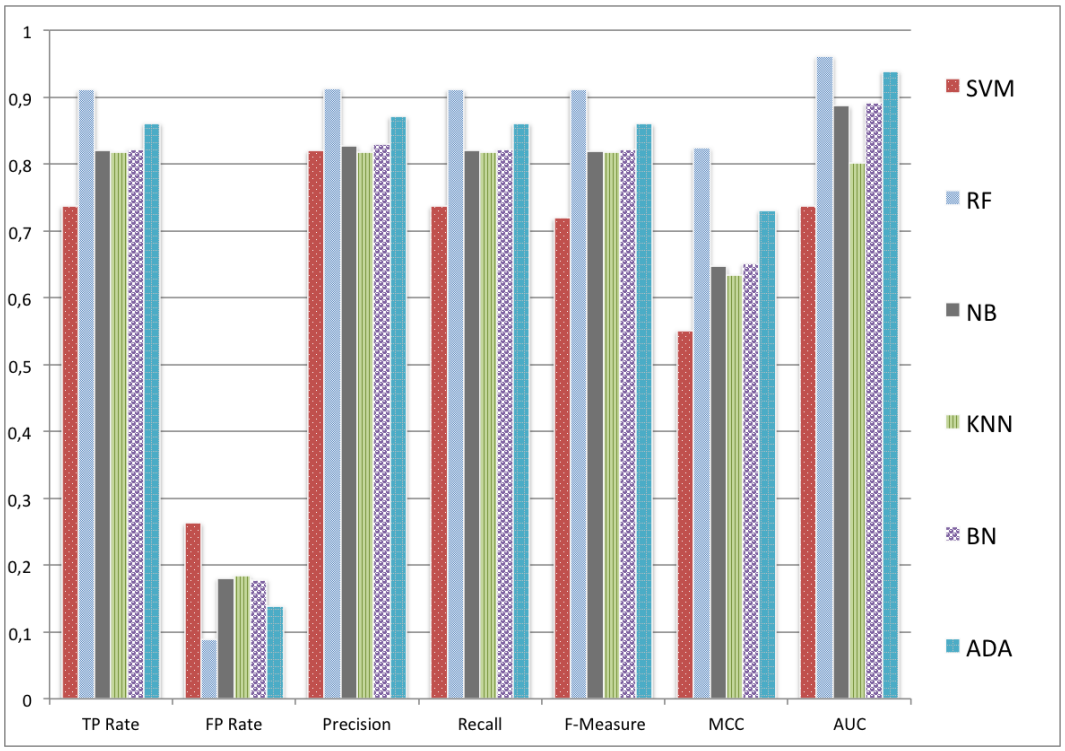

Figure 5.1 – Schematic compression of performances' results between different machine learning algorithms.

Figures 5.2 and 5.3 show respectively the ROC curve for Random Forest (highest accuracy) and SVM (lowest accuracy).

Table 5.5 – Table comparing the performances' results between different machine learning algorithms.

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	AUC
SVM	0.737	0.263	0.820	0.737	0.719	0.551	0.737
Random Forest	0.911	0.089	0.913	0.911	0.911	0.824	0.961
Naïve Bayesian	0.820	0.180	0.827	0.820	0.819	0.647	0.887
K-Nearest Neighbor	0.817	0.184	0.818	0.817	0.817	0.634	0.801
Bayesian Network	0.822	0.178	0.829	0.822	0.821	0.651	0.891
Adaptive Boosting	0.861	0.139	0.871	0.861	0.860	0.731	0.939

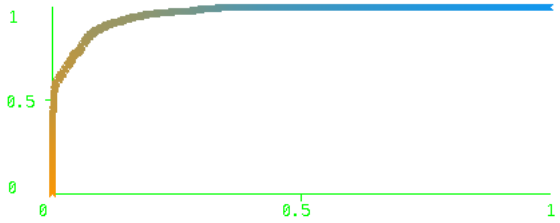

Figure 5.2 – ROC curve for Random Forest.

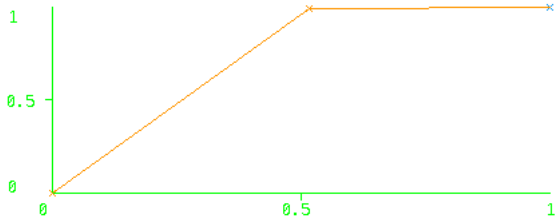

Figure 5.3 – ROC curve for SVM.

The results in tables 5.4 and 5.5 show that the compared algorithms Random forest, SVM, Adaptive Boosting, K-Nearest Neighbor, Naïve Bayesian and Bayesian Network are suitable using our features for automatic suppression of sockpuppets accounts, and they also show decent results.

5.2.3.2 Early detection of sockpuppet

In order to check the accuracy of of the early sockpuppet's detection of *SocksCatch*, we identify the differences in accuracy as time progresses for the sockpuppet's detection phase, we calculated several models (Mt) for t = 1 day of activity after registration (8 253

Table 5.6 – Table comparing the accuracy percentage of our proposed process over time.

Model	Random Forest's Accuracy
M_1	67.2%
M_{10}	69.1%
M_{20}	70.4%
M_{30}	71.9%

accounts), t = 10 days of activity after registration (8 945 accounts), t = 20 days of activity after registration (9 132 accounts), and t = 30 days of activity after registration (9 233 accounts). Time-dependent variables are likely to affect our model depending on the time t we would set when testing their performance.

Table 5.6 compares these temporal models using Random Forest; the algorithm with the best accuracy percentage on the full dataset. In this table we can see that the accuracy is increasing with time where *SocksCatch* can predict 67.2% of sockpuppets after 1 day of activity and arrive to 71.9% after 1 month of activity. However, the accuracy is increasing slowly because many sockpuppets accounts are created as sleeping accounts, and do not start their activities directly after creation, where some of them rest sleeping for many months before they start their first contribution.

The features of M_1 model are evaluated using correlation attribute evaluation method and it shows that the best three features are the number of contributions in the articles, then the number of revert after each contribution in the articles and the third is the number of contributions in the discussion of user page. From this evaluation, we can conclude that the malicious contribution of most sockpuppets are early detected and reverted by legitimate users. In addition, the number of contributions of sockpuppets in main areas are significantly different from legitimate users.

Based on these results, we can say that the early sockpuppet's detection is efficient if it is used to report suspect accounts so that administrators can keep a close eye on or

restrict certain features for suspect accounts for a time period.

5.2.3.3 Literature Comparison

In order to compare our results to previous literature, table 5.7 presents a comparison between the results of our sockpuppet's detection phase and Tsikerdekis [Tsikerdekis and Zeadally, 2014a] method. This method is the closest to our work and it presents the best accuracy between the literature's methods.

We implemented this method and compute its features on our dataset, then we train, validate and test them in the same way as presented in section 5.2.3. The results obtained for this method are the same as those published by the original authors.

Table 5.7 – Table comparing the results between our sockpuppet's detection phase and Tsikerdekis method [Tsikerdekis and Zeadally, 2014a]

Machine Learning Algorithm	Tsikerdekis method Accuracy	Our Sockpuppet's Detection Phase Accuracy
SVM	60.9%	73.7%
Random Forest	73%	91%
Naïve Bayesian	67.4%	82%
K-Nearest Neighbor	64.5%	81.5%
Bayesian Network	67.4%	82.1%
Adaptive Boosting	71%	86%

In table 5.7, we can observe that our sockpuppet's detection phase gives better accuracy percentage than previous work using all presented machine learning algorithms. The accuracy of Tsikerdekis's method is the highest using Random Forest algorithm with 73% while it is 91% using our process.

Another significant difference of 17% between the two methods arises when using K-Nearest Neighbor algorithm, where we arrive to 81.5% and Tsikerdekis's method to 64.8%.

Comparing the two studies, only the feature selection is significantly different and explains the results variation.

5.2.3.4 Discussion of results

The results presented in the previous sections show that the Random Forest algorithm has the best accuracy between all other algorithms, and that the other algorithms are nearly as efficient. The fact that five different algorithms show accuracy results over 81% is an indication of the robustness of our features selection.

Since we arrive to an accuracy of 91% we can consider that the selected features are robust and accurate and they can be used to detect sockpuppet accounts. This accuracy is due to the big number of data used in the learning algorithms and to the selected features that can differentiate between a sockpuppet and a legitimate account. The force of these features is that they cover three axes of social media's account: account's action behavior, other accounts behavior toward an account's actions, and account characteristics.

We evaluate our list of features using correlation attribute evaluation method and we found that the best three features are the delay between the time of account's registration and his first contribution, then the number of revert after each contribution in the articles and the third is the average of added bytes.

The comparison with literature shows that our features improve the accuracy of the detection of sockpuppets with 18% using Random Forest algorithm.

One of the reason that our process can be considered as effective in the detection of sockpuppet accounts is that the manipulators are not necessarily aware of the non-verbal indicators, and it is difficult to manipulate all the selected features, for example the revert number after each revision is a very good indicator because the manipulator cannot

control this feature to try to hide from detection. The non-trivial data preprocessing may also create a difficulty for the manipulators to be aware that there is a method that detects the time difference between the registration and the first contribution.

Finally, it is very difficult for a manipulator to be aware that he should change the writing behavior from time to time in order to skip from the detection.

5.3 Conclusion

In this chapter, we presented the phase of sockpuppets' detection, where we detailed all our proposed features that should differentiate between the behavior of sockpuppet and legitimate accounts. In addition, we show how to adapt and instantiate these proposed features on Wikipedia case. Furthermore, we presented the metrics used in order to evaluate the sockpuppets' detection phase.

In the evaluation of the second phase we detect sockpuppet accounts with an accuracy over 81% in 5 machine learning algorithms where the remaining algorithm detect over 73.7% of sockpuppets. This success is due to a good selection of the features that are the inputs to the algorithms, because they differentiate very well between a sockpuppet account and a legitimate account. These machine learning algorithms were trained and evaluated using 10 fold cross validation method using 10 000 cases as dataset.

We can conclude that in *SocksCatch* real time stage, Random Forest is the best machine learning algorithm that should be used in order to detect the sockpuppets on Wikipedia.

In the next chapter, we detail the third phase of *SocksCatch* process. This phase is the one that groups the detected sockpuppet created by the same sockpuppeteer in one group.

CHAPTER 6

GROUPING SOCKPUPPETS

The previous chapter describes the second phase of *SocksCatch*, where we propose in the model stage a set of features that should detect the behavior of accounts. Then, during *SocksCatch* adaptation stage, we instantiate and evaluate these features on English Wikipedia using a machine learning algorithm, where we conclude, for the dataset that have been used, that Random Forest is the algorithm that must be used in real time stage in order to detect the sockpuppet accounts on Wikipedia.

However, in social medias, manipulators attack using many sockpuppet accounts for increasingly complex criminal endeavors. They are the source of several types of manipulation such as those created to praise, defend or support a person or an organization [Stone and Richtel, 2007] or to manipulate public opinion [Elsner, 2013]. For this reason, each sockpuppet should be grouped with other sockpuppets that have the same behavior.

In this chapter, we detail *SocksCatch*'s third phase, that aims at grouping together the sockpuppets created by the same manipulator.

In this phase of *SocksCatch* model stage, we propose a process that comprises two steps: the grouping of sockpuppets based on the actions performed in the same pages then the grouping based on accounts' behavior. In *SocksCatch* adaptation stage, we describe how we adapt this process on English Wikipedia. Then, we evaluate it using different community detection algorithm, in order to find the best community detection algorithm that should be used during the real time stage.

6.1 Grouping Sockpuppets Process (Stage 1, Phase 3)

6.1.1 Grouping sockpuppets based on actions in the same pages

This step of *SocksCatch* consists in creating a relation between the sockpuppets that performed actions in the same pages. In order to identify the strength of the relationship between these sockpuppets, the process studies the approximate time between the sockpuppets' actions in each page. Then, it groups together the sockpuppets that performed actions in the same pages in a short time span.

Algorithm 2, resumes the step of grouping accounts using the actions performed in OSM pages.

Algorithm 2 Grouping accounts based on actions performed in same pages

1: $G^{act} = createActionGraph(U)$;
2: $G^{rel} = G^{act}.bipartite\_projection('accounts','euclidean\_distance')$;
3: $groupedAccounts = communityDetection(G^{rel}, communityDetectionAlgorithm)$;

Graphs construction

First, the action graph (detailed in section 4.2.1) is created by relating each sockpuppet to the pages where it performs actions (algorithm 1, page 79). If an account performed several actions in the same page, the average time of these actions is calculated and added as weight for the relation. Otherwise, the time of action itself is considered as weight of relation.

From this action graph (example in fig. 4.5, page 80), the relationship graph (detailed in section 4.2.2, page 81) that represents the relations between sockpuppets based on their actions performed in the same pages is projected (algorithm 2, line 2), using the inversed euclidean distance as weight for edges (equation 4.3, page 82).

In this first graph, there are three types of accounts:

1. Accounts without any actions - that means that they do not have any edge in the relationship graph with any other account.
2. Accounts with actions performed in common pages with other accounts - that means that these accounts have at least one edge with other accounts, based on the pages of actions, such as u1 and u2 in fig. 4.5, page 80.
3. Accounts with actions but in pages where no other accounts performed action in the same page- that means that they also do not have any edge in the relationship graph with any other account, such as u3, u4 and u5 in fig. 4.5, page 80.

Our process cannot detect the groups of the type (1) because these accounts do not have actions in the OSM so we do not have data for them[1]. Hence, these accounts are excluded from the detection phase, because we cannot detect this type of sockpuppets based on their behavior.

[1]This means they were detected originally through private data such as IP addresses

In order to group the two remaining types of accounts, we group type (2) in the next section then type (3) in the section 6.1.3.

Grouping sockpuppets

After the creation of the relationship graph (example in fig. 4.6, page 83), we use a community detection algorithm in order to produce the first set of sockpuppet groups (algorithm 2, line 3).

Fig. 6.1 represents an example of the grouped sockpuppets using *Fast Greedy* community detection algorithm.

Example

In this example four groups are created grp1, grp2, grp3 and grp4, where grp1 has two members u1 and u2, and the others groups grp2, grp3, and grp4 have respectively one member u3, u5 and u4. Let us note that in this example the account of groups with one member are the accounts that performed actions but not in the same pages where others accounts performed, because, as mentioned above, the first type of accounts is excluded.

In order to tackle this issue, we propose a second step to improve the grouping.

6.1.2 Grouping sockpuppets based on accounts' behavior

This step of *SocksCatch* improves the first groups sets, by creating additional edge between groups and sockpuppets with close behavior.

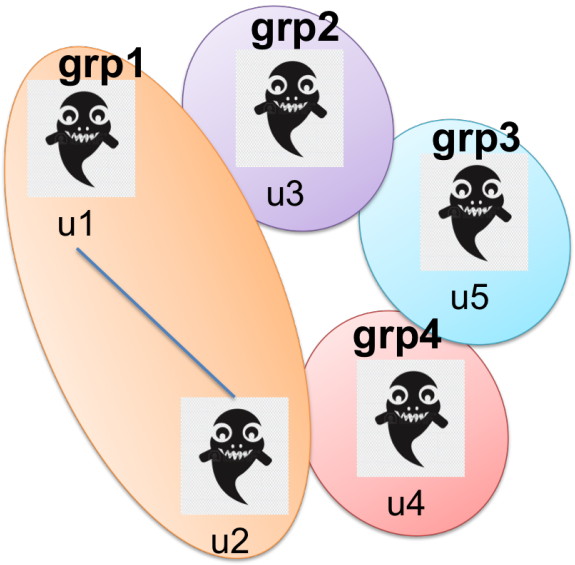

Figure 6.1 – An example of the created groups of sockpuppets based on the pages of actions and using *Fast Greedy* community detection algorithm.

The previous step produces a set of sockpuppets groups. However, this set is composed from two types of groups:

- Groups with multiple members, such as grp1 in fig. 6.1.
- Groups with one member (non-grouped sockpuppets), such as grp2 in fig. 6.1.

In order to group the non-grouped sockpuppets, we propose to add relations between groups and sockpuppets based on their behavior's similarity.

Algorithm 3 summarizes the steps used to group the sockpuppets using behavior's similarities. First, we add features to all the detected groups (line 1). This step is detailed in algorithm 4. Then we create the groups graph using these features (line 2), where this step is detailed in algorithm 5. Finally, we detect and draw the final set of groups using a community detection algorithm (lines 3, 4 and 5).

Algorithm 3 Grouping accounts after adding features

Input: $groupedAccounts$

1: $groupsWithFeatures = addFeaturesToGroups(GRP)$;
2: $G^{grp} = createGroupsGraph(groupsWithFeatures)$;
3: $Groupedgroups = communityDetection(G^{grp}, communityDetectionAlgorithm)$;

4: $finalAccountsGroups = extractGroups(Groupedgroups, groupedAccounts)$;
5: $drawGraph(finalAccountsGroups)$;

6.1.3 Account's Features

We start by defining a set of features F^{grp} that will be computed for each account, where $F^{grp} \subset F$. These features represent the behavior of the account from his actions in order to detect if there are many accounts that try to remove or add the same information, or concentrate on a specific subject with a close number of actions.

We re-use a part of the features from phase 2 (section 5.1) such as "number of account's actions in different OSM categories" and "average of bytes added or removed in each action".

In addition, other features are proposed and described below:

Total number of actions performed

Some manipulators perform lot of actions in OSM and others perform only few actions. This features will lead to connect the sockpuppets that performed actions in the same range.

$\forall u_i \in U$,

$$\texttt{TotalActionNb}(u_i) = \sum_{k=1}^{|A|} \texttt{ActionNb}(u_i, c_k) \tag{6.1}$$

Maximum number of actions performed in the same pages

We consider that some sockpuppets in the same group will focus their manipulation on the same page and others will not. This feature will help to know the type of concentration

of a group.

$\forall u_i \in U$,

$$\texttt{MaxActionInPage}(u_i) = \max \sum_{j=1}^{|uPages(u_i)|} |A_{ij}| \tag{6.2}$$

Groups graph construction

In order to get the final set of sockpuppets' groups, we create a groups graph (detailed in section 4.2.3, page 83), then we re-use a community detection algorithm to group the sockpuppets' groups.

There are two types of nodes in this new graph: *Group Nodes* representing the groups with multiple members, and *Singleton Nodes* representing the groups with one member.

Each node has the same list of features F^{grp} mentioned in section 6.1.3. However, the values of these features are computed in the following way:

- *Singleton Node* : We add the same features values of the group's member.
- *Group Node* : Each feature takes the average value of the values of the group members.

Fig. 6.2 represents an example of the detected groups in the previous step, in addition to the computed features of each account.

Example

In this example, we assume that their are four categories in the social media (c1, c2, c3 and c4). Each account has a list of eight features, where F^{grp} = Ac-

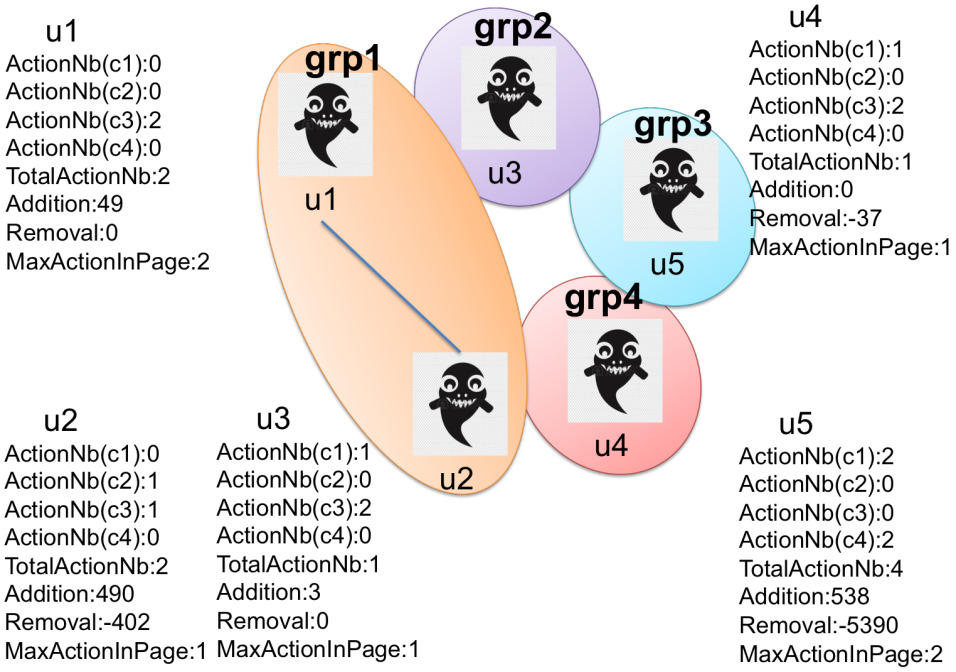

Figure 6.2 – An example of the detected groups in addition to the computed features of each account.

tionNb(c1), ActionNb(c2), ActionNb(c3), ActionNb(c4), TotalActionNb, Addition, Removal MaxActionInPage. These features are calculated for the five users.

We compute the features similarities between each *Singleton Node* and between all others nodes (*Group Node* and other *Singleton Node*) using the euclidean distance (equation 4.6, page 84). Then, a new edge is added between the *Singleton Node* and all the others nodes. The weight of each edge is the inverted euclidean distance and it is calculated using the equation 4.7, page 84.

Fig. 6.3 represents an example of the groups graph in addition to the computed features of each group and to the distance and weight between the groups.

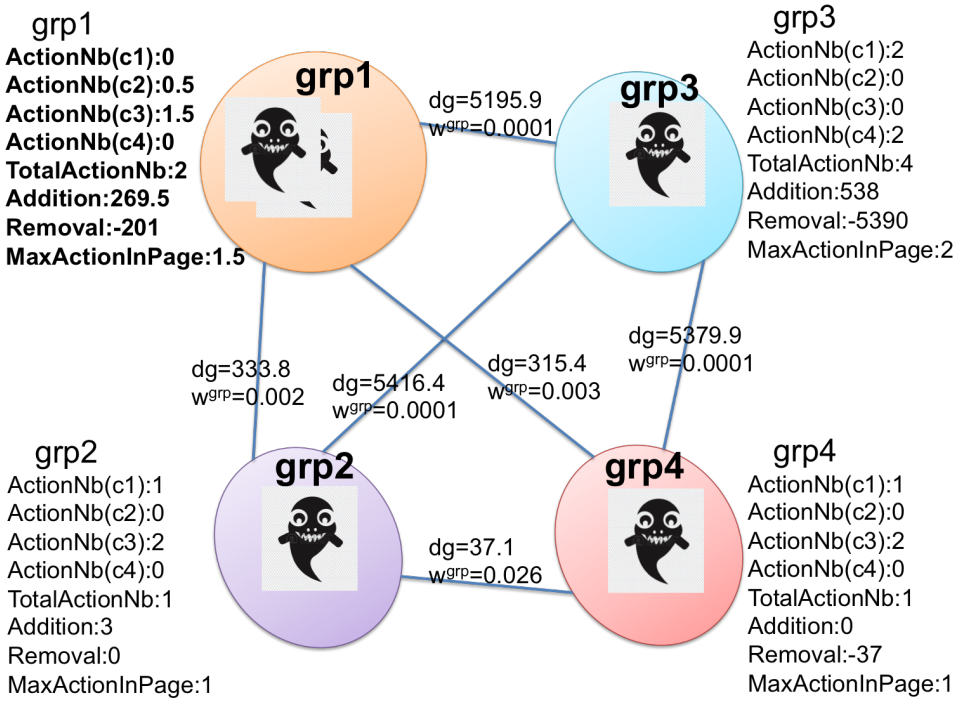

Figure 6.3 – An example of the groups graph in addition to the computed features of each group and to the distance and weight between the groups.

Example

In this figure grp2, grp3 and grp4 are *Singleton Node* , grp1 is a *Group Node*, and each node (grp1, grp2, grp3 and grp4) has a list of eight features where the features values of the group node (grp1) are the average value of the values of group's members. The edges between the nodes are weighted by inversing the euclidean distance of the features values of each node.

In order to relate the most similar groups only, we keep the edge with the highest weight between each singleton node and the others nodes. In addition, we delete all the remaining edges that have a low weight under a defined threshold.

In this way, the groups graph that relates the groups based on the most similar

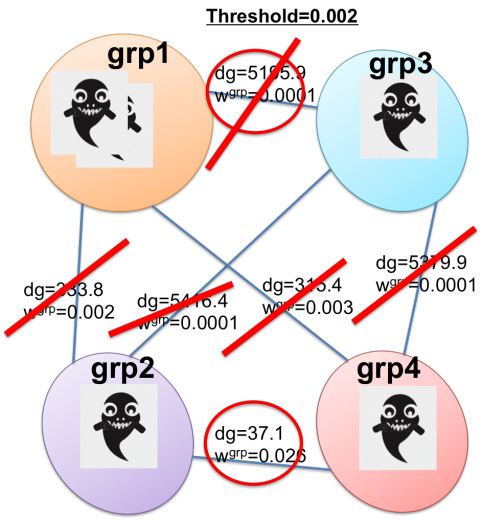

Figure 6.4 – An example of the groups graph in addition to the cuts of the edges that have a low weight.

features is built.

The algorithm 5 details how the groups graph are constructed.

Fig. 6.4 represents an example of the groups graph in addition to the cuts of the edges that have a low weight.

Algorithm 4 Add groups features

```
1: function addFeaturesToGroups(GRP)
2:     sumFeatures, setGroupFeatue = new array();
3:     for each grp_i ∈ GRP do
4:         for each f_k ∈ F^grp do
5:             for each u_j ∈ grpAccounts (grp_i) do
6:                 sumFeatures [i][k] += uFeatures(u_j, f_k);
7:             end for
8:         end for
9:         for l = 1 to |F^grp| do
10:            grp_i.setGroupFeature (l, sumFeatures [i][l]/|grpAccounts(grp_i)|);
11:        end for
12:    end for
13:    return GRP;
14: end function
```

Algorithm 5 Create groups graph

```
1: function createGroupsGraph(GRP)
2:     G^grp = newGroupsGraph ();
3:     minDistance = MAX_INTEGER;
4:     for each grp_i ∈ GRP do
5:         for each grp_j ∈ GRP do
6:             if |grpAccounts(grp_i)| = 1 or |grpAccounts(grp_j)| = 1 then
7:                 dg = euclidean_distance(grpFeatures(grp_i), grpFeatures(grp_j));
8:                 weight = 1/(1 + dg);
9:                 if (dg < minDistance) and (weight > SIMILARITY_THRESHOLD) then
10:                    minDistance = dg;
11:                    minJ = j;
12:                    w^grp = weight;
13:                end if
14:            end if
15:        end for
16:        G^grp.addEdge(i, minJ, w^grp);
17:    end for
18:    return G^grp;
19: end function
```

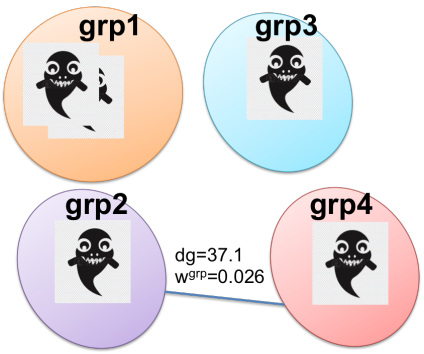

Figure 6.5 – An example of the groups graph with the edges that relate the most similar groups.

Example

In this figure grp2, grp3 and grp4 are *Singleton Node* and grp1 is *Group Node*, and each node (grp1, grp2, grp3 and grp4) has a list of eight features where the features values of the group node (grp1) are the average value of the values of group's members. The edges between the nodes are weighted by inversing the euclidean distance of the features values of each node.

Grouping sockpuppets

The groups graph with the edges that relate the most similar groups (example fig.6.5), is used as input to a community detection algorithm. The final set of sockpuppets' groups is formed from the *Singleton Nodes* that are not grouped in the final step, and the "Group Nodes" that contains the accounts grouped through both steps.

Fig. 6.6 represents an example of the final set of the grouped sockpuppets using *Fast Greedy* community detection algorithm.

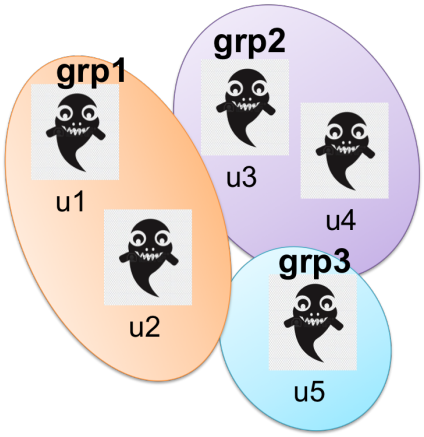

Figure 6.6 – An example of the created groups of sockpuppets after using features similarities.

Example

At the end of this example, three groups of sockpuppets are created grp1, grp2 and grp3, where grp1 has always two members u1 and u2. Additionally, u3 and u4 are grouped together after using the features similarities in grp2. However, u5 is still alone in grp3, because its behavior is not similar to others account.

6.2 Sockpuppets Grouping Adaptation (Stage 2, Phase 3)

The objective of this section, is to evaluate the efficiency of the proposed process in the grouping of sockpuppets created by the same sockpuppeteer on English Wikipedia.

This phase of *SocksCatch*'s adaptation is composed of two main steps. In first step, we group together the sockpuppets that contribute in the same pages using a community detection algorithm. Then, in second step, we adapt and compute the proposed features

in *SocksCatch* model stage on English Wikipedia, in order to create relations between the sockpuppets that have a close behavior. Furthermore, we re-group the sockpuppets with strong relation using a community detection algorithm.

At the end of the evaluation of this phase we will know if the proposed process is efficient on the case of Wikipedia and which are the two best community detection algorithms that will be used in the two steps of *SocksCatch*'s real time stage.

In the following we start by presenting the metrics that are used in order to evaluate the final set of detected groups. These metrics are used to compare the similarity between the detected groups during *SocksCatch* adaptation stage and the real groups detected manually by the administrators of English Wikipedia. Furthermore, we present how we adapt and instantiate the proposed features in the second step on Wikipedia case. Finally, we experiment and evaluate both steps of *SockCatch*'s grouping phase.

6.2.1 Evaluation Metrics : Measures to compare two clusters

In our experiment, we used modularity (described in section 3.5.1.1, page 51) and different measures to compare two clusters in order to evaluate the grouping phase of *SocksCatch* adaptation stage. We compare the output clustering Cl (groups) of *SocksCatch*'s grouping process with the real clustering Cl' using different metrics that are most used in this field.

The used metrics are the variation of information (VI) [Meilă, 2007], normalized mutual information (NMI) [Strehl and Ghosh, 2002], split-join distance [Dongen, 2000], Rand index (RI) [Rand, 1971] and adjusted Rand index (ARI) [Hubert and Arabie, 1985]. These metrics are chosen because they measure the similarities between the clusters in different ways, so that we can compare the results of similarities from different

perspective.

These metrics can be divided in three types where each type measure the similarity in a way: VI and NMI use the information-theoretical mutual information, $Split-Join$ uses the summation of set overlaps, and RI and ARI use the counting of pairs of elements.

6.2.1.1 Variation of Information

Meila proposed a measure called variation of information (VI) between two clusterings. It corresponds to the amount of information lost and gained in changing from clustering Cl to clustering Cl'. The VI score range between 0 (identical clusterings) and $\log N$ where N is the total number of elements in the clustering. Kraskov *et al.* detail a formulation to normalize VI (NVI) where NVI range from 0 and 1 (identical clusterings) [Kraskov et al., 2005].

6.2.1.2 Normalized Mutual Information

Normalized mutual information (NMI) introduced by Strehl *et al.* measures the similarity of the two sets of clusterings, ignoring permutations. An NMI score of 1 implies that the labels correlate perfectly, and an NMI score of zero indicates that the labels have no correlation with each other.

6.2.1.3 Split-Join

Dongen proposed Split-Join metric which is the number of changes (splits and joins) required to transform one clustering into the other, and in that sense it ranges from 0 (identical clustering) to $2 * N$ where N is the total number of elements in the clustering.

6.2.1.4 Rand Index

The principal of Rand Index (RI), proposed by Rand, is to check for each pair of elements in Cl, if they are classified is the same way (together or separately) in Cl'. The value of RI depends on the number of clusters and the number of elements. It ranges from 0 (no pair classified in the same way under both clusterings) to 1 (identical clusterings). Morey and Agresti showed that the Rand Index is highly dependent upon the number of clusters [Morey and Agresti, 1984].

6.2.1.5 Adjusted Rand Index

Hubert *et al.* introduced the Adjustment Rand Index (ARI) to correct the problem with Rand Index of two random clustering that does not take a constant index (e.g. zero). Thus, Adjusted Rand Index is the (normalized) difference of the Rand Index. It ranges between -1 and 1 (identical clustering).

6.2.2 Features adaptation on Wikipedia

As in the sockpuppet's detection phase, the set of features proposed in the *SocksCatch* model stage for the second grouping of sockpuppets is adapted on Wikipedia data. These features are proposed in order to relate together the sockpuppet's group (with one member and multiple members) detected in the first step of this third phase.

The adapted features that relate the sockpuppets' groups with close behavior are described below:

"Number of account's contributions by namespaces" and "Average of bytes added and removed from each contribution"

These are the features described in section 5.2.1, page 96 instantiated respectively

from “Number of account’s actions in different OSM categories” and “Average of bytes added or removed in each action”.

Total number of contributions in all the namespaces

This feature is instantiated from “Total number of actions performed”.

Maximum number of contribution in the same articles

This feature, “Maximum number of actions made in the same pages” is instantiated to one feature where the maximum number of contributions in the same article for each account is calculated.

Example

For example, let say that an account u_6 contributes 3 times in article p_{10}, 12 times in article p_3 and 8 times in article p_6.

The value of feature $MaxActionInPage^{u_6}$ is then:

$$MaxActionInPage^{u_6} = \max(3, 12, 8) = 12$$

In the following, we experiment the efficiency of the grouping sockpuppets phase proposed in *SocksCatch* model stage.

6.2.3 Grouping sockpuppets: experiments and results

In order to evaluate the third phase of *SocksCatch* adaptation stage we use the same sockpuppets detected in the second phase.

As mentioned above, the final number of sockpuppets used is 4 869 sockpuppets and they are members of 1 402 groups.

We use the API of python-igraph[2] to evaluate this phase of *SocksCatch* model, so after selecting the sockpuppet accounts, we create the action graph then we extract from it the first relationship graph, and create the groups of sockpuppets based on the contributions of accounts in the same pages using a community detection algorithm.

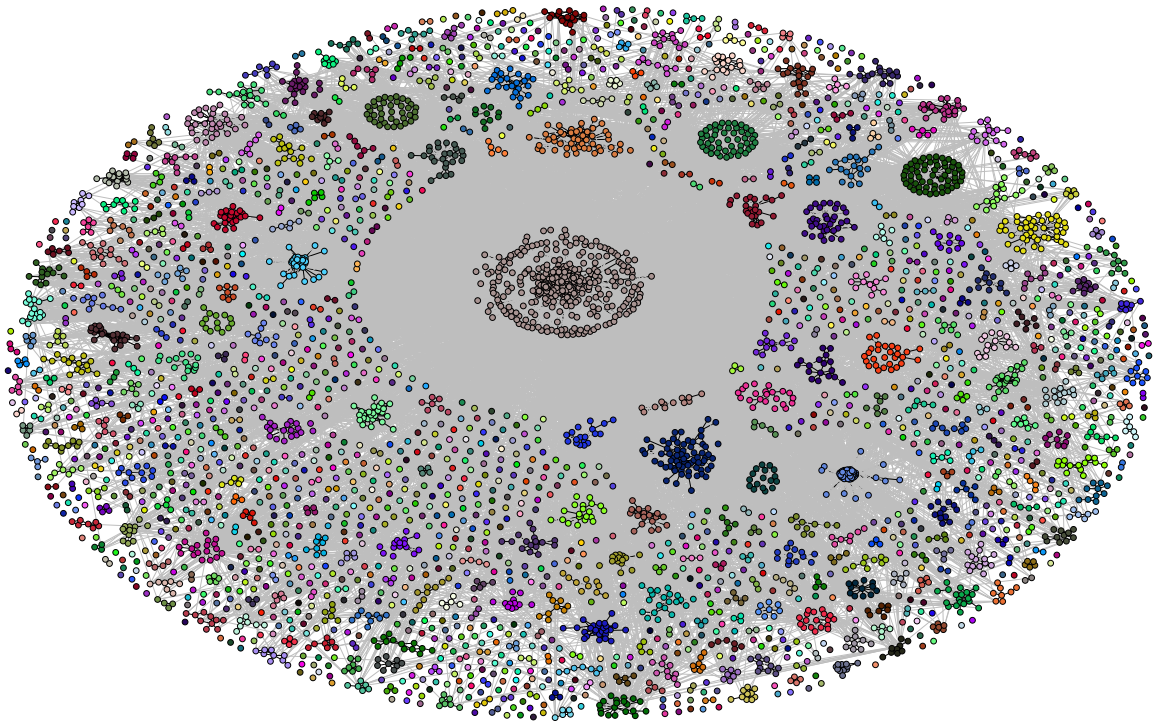

Figure 6.7 – First set of sockpuppets' groups with the relations between accounts detected through *InfoMap* algorithm.

Fig. 6.7 represents the graph of the created groups using the *InfoMap* algorithm. In this graph each color represents a group and we can observe that there are a lot of singleton nodes (group with one sockpuppet). From this figure we present in table 6.1 some statistics about the number of groups in relation to the number of their members. In this table we can see that the number of singleton nodes is 1 402 groups.

After grouping the sockpuppets, we compute the features mentioned in section 6.2.2 for each sockpuppet. We use 0.00022 as threshold of similarity, based on a manual

[2] `http://igraph.org/python`

Table 6.1 – Number of first set of groups in relation to the number of their members.

Number of groups' members	Number of groups
Singleton member	1402
Two members	171
between 3 and 10 members	225
between 11 and 20 members	49
More than 20 members	22

Table 6.2 – Number of final set of groups in relation to their number of members.

Number of groups' members	Number of groups
Singleton member	795
Two members	205
between 3 and 10 members	334
between 11 and 20 members	52
More than 20 members	23

calibration to find the best threshold. Then, we create the groups graph that contains the new relations between the singleton nodes and the most similar other nodes.

Finally, we group sockpuppets from the groups graph using the same community detection algorithm used in the previous step. We use the same community in both steps, because it is very time consuming to test with all the combinations of algorithms, but it will be interesting in the future to test this phase with more combinations other than the same algorithms in both steps.

Fig. 6.8 represents the graph of the final created groups using the *InfoMap* algorithm. In this graph each color represents a group and we can observe that most accounts are in a group with at least two members. From this figure we present in table 6.2 some statistics about the number of groups in relation to the number of their members. We can conclude from this table, that the number of singleton nodes has decreased after the second grouping from 1 402 to 795. In addition, the number of groups that have between 3 and 10 members has increased from 225 to 334 groups.

Since our objective in this section is to evaluate our model, we test this phase using the following community detection algorithms:

- $FastGreedy$ [Clauset et al., 2004].
- $InfoMap$ [Rosvall et al., 2009; Rosvall and Bergstrom, 2007].
- $LeadingEigenVector$ [Newman, 2006].
- $LabelPropagation$ [Raghavan et al., 2007].
- $WalkTrap$ [Pons and Latapy, 2006].

These algorithms are those that have received the most interest from the scientific community [Fortunato, 2010]. In addition, these approaches illustrate the diversity of methods and give an overview of the techniques classified according to their methodological principles [Danon et al., 2005; Lancichinetti and Fortunato, 2009; Papadopoulos et al., 2012].

Let us note that the grouping can be done through different methods such as community detection algorithms or graph partitioning. However, we decided to solve the problem of the grouping using the community detection algorithms, because the graph partitioning consists in grouping the nodes of a graph into a generally predetermined number of homogeneous subgroups by minimizing the number of links between the different groups. On the other hand, the community detection algorithms perform the same operation with or without requiring the knowledge of the number of communities, which is the case in this study.

Table 6.3 – Number of created groups and modularity over different community detection algorithms.

Community detection algorithm	First detected groups #	Modularity 1	Final detected groups #	Modularity 2
Fast Greedy	1658	0.379	1207	0.998
InfoMap	1869	0.366	1409	0.998
Label Propagation	1677	0.177	1235	0.981
Leading Eigen Vector	1559	0.341	1107	0.998
Walk Trap	2918	0.288	2015	0.999

6.2.3.1 Community detection algorithms comparison

The metrics described in the sections 3.5.1.1, page 51 and 6.2.1, page 124 are used to quantify the quality of the predicted groups.

Table 6.3 compares the results of different community detection algorithms used by our process. It represents the number of the groups created by *SocksCatch* in the two steps (using the contributions in same pages then using the behavior similarity). The grouping based on contributions in same pages and the number of created groups is very high in comparison with the real number which is 1 402, by example using *Fast Greedy* algorithm, the step 1 creates 1 658 groups. In fact, this high number is due to the random dataset selection, that leads to a big numbers of singleton node that contributes in some pages where no other account from the same dataset contributes in them. We can observe that the values of the modularity 1 are lower than the values of the modularity 2, because in first step of this phase the detection of the communities is more complicated due to the number of edges which is much higher than in second step.

We can observe that using *InfoMap* algorithm, the number of *SocksCatch* groups is the closest to the number of initial groups. But a close number of groups does not mean a

good grouping. In order to compare between the created groups and the initial groups we used the comparison metrics exposed in section 6.2.1, where Cl represents our created groups and Cl' represents the original groups .

Table 6.4 represents the different results of the metrics used to evaluate the similarity between *SocksCatch* created groups and the initial groups. From this table we can see that the values of Rand Index are over 98% for two algorithms, but in fact, Fowlkes *et al.* [Fowlkes and Mallows, 1983] show that the Rand Index converges to 1 as the number of clusters increases which is undesirable for a similarity measure, so it could be unrealistic in these cases. Also, we can see that the Rand index is greater than the adjusted Rand index. In fact, the range of values is greater for the adjusted Rand index ($[-1,+1]$ than for the Rand index ($[0,+1]$), but it is also affected by the number of clusters.

From table 6.4 we can deduce that the algorithms *InfoMap* and *Walk Trap* have very close results. In addition, they have the best results for most of the similarity metrics between all other algorithms, for example, The NMI value is 0.868 for *InfoMap* and 0.873 for *Walk Trap*, which means that the created groups are $\approx 87\%$ similar to the initial groups using NMI metric. Also we can observe that Split-Join value of *InfoMap* is the lowest which means that it is the best between all the algorithms as it measures the number of changes (splits and joins) required to transform one clustering into the other, which means that the nearest number to 0 is most similar. Also, for the VI metric where the lowest number is the best. However the NVI which is a normalized value of VI has the best value for *InfoMap* and *Walk Trap* and which is equal to 0.858, that means that for VI metric, *SocksCatch* create $\approx 86\%$ of sockpuppets' groups similar to the real groups on English Wikipedia.

Hence, we can conclude that the best algorithm that will be used in *SocksCatch*'s

Table 6.4 – Similarities between our created groups and the original groups over different community detection algorithms.

Community detection Algorithm	VI	NVI	NMI	Split-Join	RI	ARI
Fast Greedy	3.021	0.749	0.729	3537	0.882	0.021
Infomap	1.701	0.858	0.868	3041	0.987	0.126
Label propagation	3.609	0.7	0.658	3649	0.727	0.008
Leading Eigen Vector	3.741	0.689	0.635	3707	0.769	0.011
Walk Trap	1.707	0.858	0.873	3075	0.991	0.125

real time stage is *InfoMap* because it has the best results in most similarity metrics, in addition it detect the nearest groups' number (1 409) to the real groups' number (1 402), with a difference of 0.5% only in the term of the number of groups.

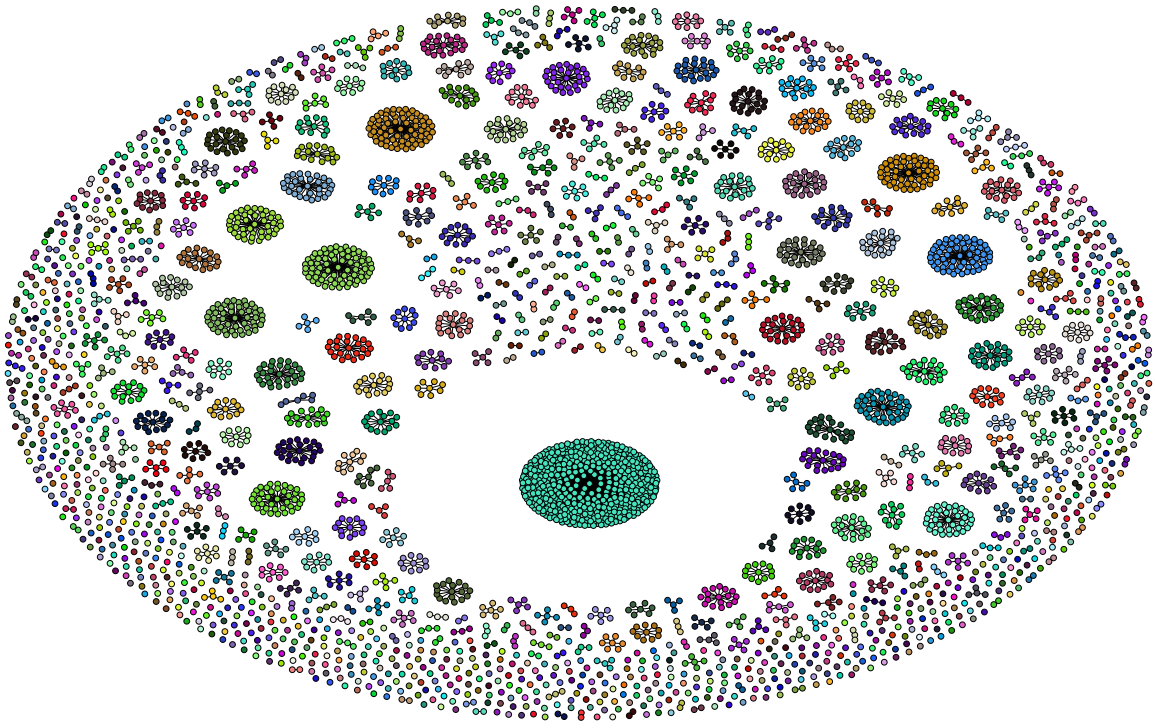

Figure 6.8 – Final groups of sockpuppets detected through *InfoMap* algorithm.

6.3 Conclusion

In this chapter, we present the phase of grouping the sockpuppets created by the same sockpuppeteer. In this phase, we propose a grouping process that is divided into two main steps. First, we group the sockpuppets that perform actions in the same pages in a close time. Then, we improve the detected groups by relating the groups with single member to others groups (single and multiple members) thanks to the comparison of their behavior using a set of proposed features. Furthermore, we presented how we adapt and evaluate the proposed process on English Wikipedia.

In the evaluation of the third phase we created a set of sockpuppets' groups with an $\approx 86\%$ of similarity with the real sockpuppet's groups (according to NMI and NVI metrics), with a difference of 0.5% in the number of total groups between the created and the real groups. In this evaluation, we experiment the two steps of this phase using the same community detection algorithms, but an interesting perspective is to evaluate them using different combination to check if they give better results.

From the results of our experiment, we can conclude that during the application of *SocksCatch* in real time on the dataset extracted from Wikipedia, InfoMap in the best community detection that must be used in both steps, with a threshold equal to 0.00022 in the second step, in order to group together the sockpuppets create by the same manipulators.

CHAPTER 7

Conclusions and Perspectives

Sockpuppets have become an increasingly important issue in social medias. Many solutions were created in order to detect malicious behaviors on social medias. Two main approaches are used: verbal communication analysis and non-verbal behavior analysis.

Detection through verbal communication relies on the analysis of textual content, such as studying the links in post on Facebook, detecting sentiments from tweets on Twitter, or checking repeated words from Reddit's comments.

Detection through non-verbal behavior does not take into account the semantics of the users interactions. Its focus is on the other aspects of the behavior of the user using descriptive values, such as the number of friends' requests on Facebook, the average of tweets every day, the average of added characters in each comment on Reddit.

In this thesis, we present *SocksCatch*, a process to detect and group sockpuppets on social medias using non-verbal behavior analysis. It is divided in three main phases: The first phase objective is the preparation of the process. The second phase objective is the detection of sockpuppet accounts using a machine learning algorithm. The third

phase objective is the grouping of sockpuppet accounts created by a same user using a community detection algorithm.

The introduction of this thesis raises six questions that have been addressed by our *SocksCatch* proposal:

Q1: How to differentiate between the behavior of sockpuppets and legitimate accounts ?

A1: The non-verbal behavior of sockpuppets are studied. Chapter 3 shows that non-verbal behavior analysis are more efficient than verbal communication because they give better results. Furthermore, users that have an objective to manipulate a precise article will find a difficulty to skip from the detection and to change their non-verbal behavior if they want to achieve their goal.

To answer this question, a set of characteristics to differentiate the behavior of sockpuppets from that of legitimate accounts is proposed such as the number of reported actions of an account, and the delay between account's creation and first action.

Q2: Can sockpuppets be automatically detected ?

A2: In the second phase of *SocksCatch*, data is collected from English Wikipedia, then our proposed characteristics are adapted on Wikipedia environment. Furthermore, these characteristics are instantiated and computed for a set of 10 000 accounts (half sockpuppets and half legitimates). Finally, the computed characteristics are evaluated using different machine learning algorithms to automatically detect sockpuppets. Our experiment shows that *SocksCatch* detects around 91% of the selected sockpuppets using Random Forest algorithm on this evaluation dataset, which proves the feasibility of the approach.

Q3: How to relate together sockpuppets that have a close behavior ?

A3: Non-verbal behaviors analysis are used in order to find out the common elements

and the differences in the behavior of sockpuppets. Furthermore, the third phase of *SocksCatch* propose to create a set of graphs that relate all sockpuppets contributing in the same areas in a close time and with a close behavior using a set non-verbal behavior characteristics.

Q4: Can the sockpuppets that correspond to a single malicious user be grouped together?

A4: The same 4 869 sockpuppets collected from English Wikipedia are used. Then, the proposed graphs are applied on these accounts, in order to group and relate together the sockpuppets created by the same manipulator. In addition, the proposed characteristics are adapted and computed on Wikipedia environment. The applied graphs are evaluated using different community detection algorithms. Our experiment shows that *SocksCatch's* grouped sockpuppets and the real sockpuppet's groups are similar around 87%, according to the cluster's comparison measures: *normalized variation of information* (NVI) and *normalized mutual information* (NMI).

Q5: What are the common characteristics between different types of online social media that can be exploited to propose a generic model for sockpuppets detection?

A5: Chapter 2 shows that OSMs are founded on common grounding because they are usually based on accounts to uniquely identify users. These accounts have some common properties between all OSMs such as usernames and the time of creation of each account. The action possibilities can also be modeled independently from the particular cases, for example all OSMs include ways to add information. Our generic model of OSMs presented in chapter 4 benefits from these common characteristics, where it shows how to present an OSM from a general point of view, without depending from the specification of this OSM.

Q6: How to adapt the solution to a specific online social media?

A6: Chapters 4, 5 and 6 presents a concrete example for the adaptation of our proposed process (*SocksCatch*) on a social media: English Wikipedia. This adaptation shows how to instantiate and apply the general model and characteristics on accounts from Wikipedia, by adapting the features of each phase and selecting the best machine learning and community detection algorithms.

As mentioned above, *SocksCatch*'s detection phase is experimented using machine learning algorithms. However, the success of a machine learning algorithm depends on the selection of features that are the inputs of the algorithm. In this way, in the evaluation of the second phase, an accuracy over 81% is achieved with five algorithms: Random Forest, Adaptive Boosting, Naïve Bayesian, Bayesian Network and K-Nearest Neighbor. We believe that these good results come from a relevant selection of characteristics, which cover well the differentiating information between a sockpuppet account and a legitimate account. These machine learning algorithms were trained, evaluated and tested using 10 fold cross validation method using 10 000 cases as dataset.

In order to check if *SocksCatch* can detect sockpuppets early before performing their attacks, it is evaluated after different time of sockpuppet's activities, where we found that our proposed process is capable to detect 67.2% of sockpuppets on Wikipedia after 1 day of activity after their registration using Random Forest algorithm. Based on this result, we can say that the detection phase is efficient if it is used to report suspect accounts so that administrators can keep a close eye on or restrict certain features for suspect accounts for a time period. Furthermore, the behavior of sockpuppets is significantly different from that of legitimate users as soon as they begin their activity

The detection phase of *SocksCatch* is also compared with the literature where the result of comparison shows that our features improve the accuracy of the detection of

sockpuppets with 18% using Random Forest algorithm.

The evaluation of the third phase shows ≈ 86% of similarity between the sockpuppets' groups detected by our algorithm and the real sockpuppet's groups (according to NMI and NVI metrics), with a difference of 0.5% in the number of total groups between the created and the real groups.

This work demonstrates that *SocksCatch* can be successfully used on English Wikipedia using Random Forest as machine learning algorithm for the detection phase, and InfoMap as community detection algorithm for the grouping phase.

Perspectives

As we have seen, *SocksCatch* shows very good results in the detection and grouping of sockpuppets. In the future, it must be evaluated on different social media data, like Facebook or Twitter. To achieve this, the different features proposed in the *SocksCatch* process have to be instantiated depending on the distinctive characteristics of each social media, and depending on which data is available.

It is possible to use or adapt the features proposed in the *SocksCatch* model for several social media. For example, table 7.1 shows an example of the characteristics that can be used on data from Twitter and from Facebook to automatically calculate features similar to those used in this thesis. The adaptation to these social medias environments may be direct (e.g. the interval between the user registration and his first post, or the number of contributions in each namespace / page type) or indirect (e.g. reverts do not exist in some social media, but a comment deletion may be assimilated to a revert). Note that some of these data are not publicly available, and thus would necessitate to have an administrator access to be put in practice. These features should be evaluated in the

Table 7.1 – Comparison between the general proposed features and the instantiated features for Twitter and Facebook data

General features	Twitter features	Facebook features
Number of account's actions in different OSM categories	Number of account's tweets or comments in different categories (home, tweets . . .)	Contributions' number by the user in the different types of Facebook pages (groups, profile, friend page . . .)
Average of bytes added or removed in each writing action	Average of bytes added in each tweet and in each comment	Average of added bytes in the personal profile and on the others wall
Diversity of pages modified by an account's actions	Average of tweets and comments for the same hashtag	Average of contributions on the hall of a friend or in a page or in a group
Number of reported actions of each account	Number of received reports on account tweet and the number of deleted comments	Number of received reports and the number of deleted comments
Delay between account's creation and first action	Delay between the time of account's registration and the time of his first action on twitter	Delay between the time of account's registration and the time of his first action on Facebook

future using the adaptation stage of the *SocksCatch* process.

Non-verbal indicators cannot be manipulated as easily as verbal communications, for example the revert number after each revision is a good indicator because the manipulator cannot control this feature to try to hide from detection. The non-trivial data pre-processing may also create a difficulty for the manipulators to be aware that there is a method that detects the time difference between the registration and the first contribution. In the future, other features that are more difficult to be known by manipulators can be added, in order to make the sockpuppets detection more accurate such as the number of reports that an account performs on other accounts, and the delay between the contributions' time of an account.

Another perspective concerns the grouping phase of *SocksCatch*. It is composed by

two steps, where the same community detection algorithm is used in the application of both steps. It would be interesting to evaluate different community detection algorithms, and test all their combinations in first and second step.

Finally, the implementation of *SocksCatch* to automatically detect and group Sockpuppets in Wikipedia would be helpful to its administrators in terms of time spent to achieve this task manually, as it is currently the case. However, we should note that the online performance of our method is not tested, where the machine learning algorithm will be trained before running the real time process. In addition, an extended solution must be made if the behavior of sockpuppets change over time, where the machine learning algorithm learn incrementally in real time about the new behaviors. Furthermore, the groups would have to be updated progressively when detecting a new sockpuppet.

BIBLIOGRAPHY

Adam, G. (2013).
Facebook and youtube contain the most spam of all social networks.
SC Media.
https://www.scmagazine.com/facebook-and-youtube-contain-the-most-spam-of-all-social-networks/article/542685/.

Alex, Y. (2017).
47 social media statistics to bookmark for 2017.
Sprout Social.
https://sproutsocial.com/insights/social-media-statistics/.

Alsaleh, M., Alarifi, A., Al-Salman, A. M., Alfayez, M., and Almuhaysin, A. (2014).
Tsd: Detecting sybil accounts in twitter.
In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 463–469. IEEE.

Altman, N. S. (1992).
An introduction to kernel and nearest-neighbor nonparametric regression.
The American Statistician, 46(3):175–185.

Ambika Choudhary, M. (2014).
The evolution of social media 2004 - 2014: The good, the bad and the ugly of it !
Daze Info.
https://dazeinfo.com/2014/12/12/evolution-social-media-2004-2014-good-bad-ugly/.

Anderson, D. S., Fleizach, C., Savage, S., and Voelker, G. M. (2007).
Spamscatter: Characterizing internet scam hosting infrastructure.
In *Usenix Security*, pages 1–14.

Andrew, H. (2017).
Top social network demographics 2017.

Social Media Today.
https://www.socialmediatoday.com/social-networks/top-social-network-demographics-2017-infographic.

Antonio, L., Cliff, E., and Reynold, X. (2012).
Feature selection and classification of spam on social networking sites.

Arenas, A., Diaz-Guilera, A., and Pérez-Vicente, C. J. (2006).
Synchronization reveals topological scales in complex networks.
Physical review letters, 96(11):114102.

Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D. (2007).
Low-resource routing attacks against tor.
In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20. ACM.

Beutel, A., Xu, W., Guruswami, V., Palow, C., and Faloutsos, C. (2013).
Copycatch: stopping group attacks by spotting lockstep behavior in social networks.
In *Proceedings of the 22nd international conference on World Wide Web*, pages 119–130. ACM.

Bilge, L., Strufe, T., Balzarotti, D., and Kirda, E. (2009).
All your contacts are belong to us: automated identity theft attacks on social networks.
In *Proceedings of the 18th international conference on World wide web*, pages 551–560. ACM.

Bisson, D. (2016).
Social engineering attacks to watch out for.
Tripwire.
https://www.tripwire.com/state-of-security/security-awareness/5-social-engineering-attacks-to-watch-out-for/.

Blizzard, E. (2014).
World of warcraft.
Wikipedia.
https://en.wikipedia.org/wiki/World_of_Warcraft.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008).
Fast unfolding of communities in large networks.
Journal of statistical mechanics: theory and experiment, 2008(10):P10008.

Boshmaf, Y., Muslukhov, I., Beznosov, K., and Ripeanu, M. (2011).
The socialbot network: when bots socialize for fame and money.
In *Proceedings of the 27th annual computer security applications conference*, pages 93–102. ACM.

Bosma, M., Meij, E., and Weerkamp, W. (2012).
A framework for unsupervised spam detection in social networking sites.
In *European Conference on Information Retrieval*, pages 364–375. Springer.

Botha, E., Farshid, M., and Pitt, L. (2011).
How sociable? an exploratory study of university brand visibility in social media.
South African Journal of Business Management, 42(2):43–51.

Breiman, L. (2001).
Random forests.
Machine learning, 45(1):5–32.

Bu, Z., Xia, Z., and Wang, J. (2013).
A sock puppet detection algorithm on virtual spaces.
Knowledge-Based Systems, 37:366–377.

Cao, Q., Sirivianos, M., Yang, X., and Pregueiro, T. (2012).
Aiding the detection of fake accounts in large scale social online services.
In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 15–15. USENIX Association.

Cao, Q., Yang, X., Yu, J., and Palow, C. (2014).
Uncovering large groups of active malicious accounts in online social networks.
In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 477–488. ACM.

Chakrabarti, D. (2004).
Autopart: Parameter-free graph partitioning and outlier detection.
In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 112–124. Springer.

Clauset, A., Newman, M. E., and Moore, C. (2004).
Finding community structure in very large networks.
Physical review E, 70(6):066111.

Cohen, W. W. et al. (1996).
Learning rules that classify e-mail.
In *AAAI spring symposium on machine learning in information access*, volume 18, page 25. California.

Cortes, C. and Vapnik, V. (1995).
Support-vector networks.
Machine learning, 20(3):273–297.

Crabb, E. S., Mishler, A., Paletz, S., Hefright, B., and Golonka, E. (2015).
Reading between the lines: a prototype model for detecting twitter sockpuppet accounts using language-agnostic processes.
In *International Conference on Human-Computer Interaction*, pages 656–661. Springer.

Daft, R. L. and Lengel, R. H. (1983).
Information richness. a new approach to managerial behavior and organization design.
Technical report, DTIC Document.

Daft, R. L. and Lengel, R. H. (1986).
Organizational information requirements, media richness and structural design.
Management science, 32(5):554–571.

Danezis, G. and Mittal, P. (2009).
Sybilinfer: Detecting sybil nodes using social networks.
In *Network and Distributed System Security (NDSS)*. San Diego, CA.

Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. (2005).
Comparing community structure identification.
Journal of Statistical Mechanics: Theory and Experiment, 2005(09):P09008.

Davoust, A., Skaf-Molli, H., Molli, P., Esfandiari, B., and Aslan, K. (2015).
Distributed wikis: a survey.
Concurrency and Computation: Practice and Experience, 27(11):2751–2777.

Dongen, S. (2000).
Performance criteria for graph clustering and markov cluster experiments.
Technical report, Centre for Mathematics and Computer Science (CWI).

Douceur, J. R. (2002).
The sybil attack.

In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer.

Drucker, H., Wu, D., and Vapnik, V. N. (1999).
Support vector machines for spam categorization.
IEEE Transactions on Neural networks, 10(5):1048–1054.

Duch, J. and Arenas, A. (2005).
Community detection in complex networks using extremal optimization.
Physical review E, 72(2):027104.

Elizabeth, K. (2015).
Facebook does not allow multiple accounts.
Tech For Luddites.
https://techforluddites.com/facebook-does-not-allow-multiple-accounts/.

Elsner, K. (2013).
China uses an army of sockpuppets to control public opinion – and the us will too.
Guardian Liberty Voice.
http://guardianlv.com/2013/11/china-uses-an-army-of-sockpuppets-to-control-public-opinion-and-the-us-will-too/.

Eric, M. (2012).
Twitter, facebook, foursquare: Tools of the modern burglar?
CNET Networks.
https://www.cnet.com/news/twitter-facebook-foursquare-tools-of-the-modern-burglar/.

Erving, G. (1959).
The presentation of self in everyday life.
Garden City, NY: Anchor, pages 1–17.

Evan, A. (2017).
How much time do people spend on social media?
Social Media Today.
https://www.socialmediatoday.com/marketing/how-much-time-do-people-spend-social-media-infographic.

Fielding, N. and Cobain, I. (2011).
Revealed: Us spy operation that manipulates social media.

The Guardian, 17.
https://www.theguardian.com/technology/2011/mar/17/us-spy-operation-social-networks.

Fortunato, S. (2010).
Community detection in graphs.
Physics reports, 486(3):75–174.

Fowlkes, E. B. and Mallows, C. L. (1983).
A method for comparing two hierarchical clusterings.
Journal of the American statistical association, 78(383):553–569.

Fred, T. (2009).
Zeus botnet targets facebook.
Digital Degenerate Blog.
https://blog.appriver.com/2009/10/zeus-botnet-targets-facebook/.

Freund, Y. and Schapire, R. E. (1995).
A desicion-theoretic generalization of on-line learning and an application to boosting.
In *Computational learning theory*, pages 23–37. Springer.

Gani, K., Hacid, H., and Skraba, R. (2012).
Towards multiple identity detection in social networks.
In *Proceedings of the 21st International Conference on World Wide Web*, pages 503–504. ACM.

Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y., and Zhao, B. Y. (2010).
Detecting and characterizing social spam campaigns.
In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 35–47.

Goga, O. (2014).
Matching user accounts across online social networks: methods and applications.
PhD thesis, LIP6-Laboratoire d'Informatique de Paris 6.

Goldberg, A. V. and Tarjan, R. E. (1988).
A new approach to the maximum-flow problem.
Journal of the ACM (JACM), 35(4):921–940.

Goolsby, R., Shanley, L., and Lovell, A. (2013).
On cybersecurity, crowdsourcing, and social cyber-attack.
Technical report, DTIC Document.

Hall, M., Witten, I., and Frank, E. (2011).
Data mining: Practical machine learning tools and techniques.
Kaufmann, Burlington.

Hastings, M. B. (2006).
Community detection as an inference problem.
Physical Review E, 74(3):035102.

Hu, X., Tang, J., Zhang, Y., and Liu, H. (2013).
Social spammer detection in microblogging.
In *IJCAI*, volume 13, pages 2633–2639. Citeseer.

Hubert, L. and Arabie, P. (1985).
Comparing partitions.
Journal of classification, 2(1):193–218.

Islam, M. R., Chowdhury, M. U., and Zhou, W. (2005).
An innovative spam filtering model based on support vector machine.
In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 348–353. IEEE.

Jain, P., Kumaraguru, P., and Joshi, A. (2013).
@ i seek'fb. me': Identifying users across multiple online social networks.
In *Proceedings of the 22nd international conference on World Wide Web*, pages 1259–1268. ACM.

Joachims, T. (1998).
Making large-scale svm learning practical.
Technical report, Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Johansson, F., Kaati, L., and Shrestha, A. (2015).
Timeprints for identifying social media users with multiple aliases.
Security Informatics, 4(1):7.

John, D. (2017).
Could the stock market ever be hacked?
CNBC.
https://www.cnbc.com/2017/07/31/could-the-stock-market-ever-be-hacked.html.

Jordan, R., Michael, R., and Andrew, W. (2016).
How to hack an election.
Bloomberg Businessweek.
https://www.bloomberg.com/features/2016-how-to-hack-an-election/.

Juola, P. (2012).
Detecting stylistic deception.
In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 91–96. Association for Computational Linguistics.

Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S. (2008).
Spamalytics: An empirical analysis of spam marketing conversion.
In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 3–14. ACM.

Kaplan, A. M. and Haenlein, M. (2010).
Users of the world, unite! the challenges and opportunities of social media.
Business horizons, 53(1):59–68.

Kaplan, A. M. and Haenlein, M. (2011).
The early bird catches the news: Nine things you should know about micro-blogging.
Business horizons, 54(2):105–113.

Kelli, G. (2015).
Identity theft victims: You might know the culprit.
CNBC.
https://www.cnbc.com/2015/07/21/identity-theft-victims-may-know-the-culprit.html.

Kelly, H. (2012).
83 million facebook accounts are fakes and dupes.
Cable News Network (CNN).
http://edition.cnn.com/2012/08/02/tech/social-media/facebook-fake-accounts/index.html.

Kim, Z. (2011).
Condé nast got hooked in $8 million spear-phishing scam.
Wired.
https://www.wired.com/2011/04/condenast-hooked-by-spear-phisher/.

Kolari, P., Java, A., Finin, T., Oates, T., and Joshi, A. (2006).
Detecting spam blogs: A machine learning approach.
In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 6, pages 1351–1356.

Kolcz, A. and Alspector, J. (2001).
Svm-based filtering of e-mail spam with content-specific misclassification costs.
In *Workshop on Text Mining (TextDM'2001)*.

Kraskov, A., Stögbauer, H., Andrzejak, R. G., and Grassberger, P. (2005).
Hierarchical clustering using mutual information.
EPL (Europhysics Letters), 70(2):278.

Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S. (2009).
Spamcraft: An inside look at spam campaign orchestration.
In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*.

Kumar, S., Cheng, J., Leskovec, J., and Subrahmanian, V. (2017).
An army of me: Sockpuppets in online discussion communities.
In *Proceedings of the 26th International World Wide Web Conference*.

Lancichinetti, A. and Fortunato, S. (2009).
Community detection algorithms: a comparative analysis.
Physical review E, 80(5):056117.

Largeron, C., Mougel, P.-N., Benyahia, O., and Zaïane, O. (2017).
Dancer: dynamic attributed networks with community structure generation.
Knowledge and Information Systems, 53(1):109–151.

Lendevrie, J., Levy, J., and Lindon, D. (2010).
La politique relationnelle et les marques face au web 2.0.
In *Mercator : Theorie et pratique du marketing*, pages 791–808. Dunod, Paris.

Lian, Q., Zhang, Z., Yang, M., Zhao, B. Y., Dai, Y., and Li, X. (2007).
An empirical study of collusion behavior in the maze p2p file-sharing system.
In *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*, pages 56–56. IEEE.

Liu, D., Wu, Q., Han, W., and Zhou, B. (2016).
Sockpuppet gang detection on social media sites.
Frontiers of Computer Science, 10(1):124–135.

Lund, H. M. (2017).
Community detection in complex networks.
Master's thesis, The University of Bergen.

Luo, F., Wang, J. Z., and Promislow, E. (2008).
Exploring local community structures in large networks.
Web Intelligence and Agent Systems: An International Journal, 6(4):387–400.

M., J. (2014).
Facebook accusé vendre des données de vos messages privés aux annonceurs.
20minutes.
http://www.20minutes.fr/web/1269387-20140103-20140103-facebook-accuse-scanner-messages-prives-vendre-donnees-annonceurs.

Madhusudan, T. (2003).
On a text-processing approach to facilitating autonomous deception detection.
In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10–pp. IEEE.

Mangold, W. G. and Faulds, D. J. (2009).
Social media: The new hybrid element of the promotion mix.
Business horizons, 52(4):357–365.

Markines, B., Cattuto, C., and Menczer, F. (2009).
Social spam detection.
In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, pages 41–48. ACM.

Mary, L. (2017).
40 essential social media marketing statistics for 2017.

Word Stream.
http://www.wordstream.com/blog/ws/2017/01/05/social-media-marketing-statistics.

Massen, C. P. and Doye, J. P. (2005).
Identifying communities within energy landscapes.
Physical Review E, 71(4):046101.

Mathew, I. (2012).
If you think twitter doesn't break news, you're living in a dream world.
Gigaom.
https://gigaom.com/2012/02/29/if-you-think-twitter-doesnt-break-news-youre-living-in-a-dream-world/.

Mayur, K. (2009).
Users of social networking websites face malware and phishing attacks.
Symantec Official Blog.
https://www.symantec.com/connect/blogs/users-social-networking-websites-face-malware-and-phishing-attacks.

Mediakix, T. (2016).
How much time do we spend on social media?
Media Kix.
http://mediakix.com/2016/12/how-much-time-is-spent-on-social-media-lifetime/.

Meilă, M. (2007).
Comparing clusterings—an information based distance.
Journal of multivariate analysis, 98(5):873–895.

Mishne, G., Carmel, D., Lempel, R., et al. (2005).
Blocking blog spam with language model disagreement.
In *AIRWeb*, volume 5, pages 1–6.

Morey, L. C. and Agresti, A. (1984).
The measurement of classification agreement: an adjustment to the rand statistic for chance agreement.
Educational and Psychological Measurement, 44(1):33–37.

Nabaa, M., Bertelle, C., Dutot, A., Olivier, D., and Mallet, P. (2010).
Communities detection algorithm to minimize risk during an evacuation.
In *Systems Conference, 2010 4th Annual IEEE*, pages 323–328. IEEE.

Nagamalai, D., Dhinakaran, B. C., and Lee, J. K. (2010).
Bayesian based comment spam defending tool.
CoRR.

Narayanan, A., Paskov, H., Gong, N. Z., Bethencourt, J., Stefanov, E., Shin, E. C. R., and Song, D. (2012).
On the feasibility of internet-scale author identification.
In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 300–314. IEEE.

Newman, M. E. (2004).
Fast algorithm for detecting community structure in networks.
Physical review E, 69(6):066133.

Newman, M. E. (2006).
Finding community structure in networks using the eigenvectors of matrices.
Physical review E, 74(3):036104.

Newman, M. E. and Girvan, M. (2004).
Finding and evaluating community structure in networks.
Physical review E, 69(2):026113.

Newsome, J., Shi, E., Song, D., and Perrig, A. (2004).
The sybil attack in sensor networks: analysis & defenses.
In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 259–268. ACM.

Norajong (2010).
Why the number of people creating fake accounts and using second identity on facebook are increasing.
Net Studies.
http://networkconference.netstudies.org/2010/05/why-the-number-of-people-creating-fake-accounts-and-using-second-identity-on-facebook-are-increasing/.

Novak, J., Raghavan, P., and Tomkins, A. (2004).
Anti-aliasing on the web.
In *Proceedings of the 13th international conference on World Wide Web*, pages 30–39. ACM.

Pantel, P. and Lin, D. (1998).

Spamcop: A spam classification and organization program. learning for text categorization-papers from the aaai workshop, madison wisconsin, 95-98.
Technical report, AAAI Technical Report WS-98-05.

Papadopoulos, S., Kompatsiaris, Y., Vakali, A., and Spyridonos, P. (2012).
Community detection in social media.
Data Mining and Knowledge Discovery, 24(3):515–554.

Patel, N., Lopez, C., Partalas, I., and Segond, F. (2017).
Une approche hybride pour la détection d'influenceurs dans les médias sociaux.
In *28es Journées francophones d'Ingénierie des Connaissances IC 2017*, pages 115–120.

Paul, P. P., Sultana, M., Matei, S. A., and Gavrilova, M. (2015).
Editing behavior to recognize authors of crowdsourced content.
In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 1676–1681. IEEE.

Pearl, J. (1985).
Bayesian networks: A model of self-activated memory for evidential reasoning.
In *Proceedings of the 7th Conference of the Cognitive Science Society, 1985*, pages 329–334.

Podhradsky, A., Dovidio, R., Engebretson, P., and Casey, C. (2013).
Xbox 360 hoaxes, social engineering, and gamertag exploits.
In *System Sciences (HICSS), 46th Hawaii International Conference*, pages 3239–3250. IEEE.

Pons, P. and Latapy, M. (2006).
Computing communities in large networks using random walks.
J. Graph Algorithms Appl, 10(2):191–218.

Post, A., Shah, V., and Mislove, A. (2011).
Bazaar: Strengthening user reputations in online marketplaces.
In *Proceedings of NSDI'11: 8th USENIX Symposium on Networked Systems Design and Implementation*, page 183.

Qian, T. and Liu, B. (2013).
Identifying multiple userids of the same author.
In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1124–1135.

Raghavan, U. N., Albert, R., and Kumara, S. (2007).
Near linear time algorithm to detect community structures in large-scale networks.
Physical review E, 76(3):036106.

Rand, W. M. (1971).
Objective criteria for the evaluation of clustering methods.
Journal of the American Statistical association, 66(336):846–850.

Richmond, R. (2010).
Stolen facebook accounts for sale.
The New York Times, 2.
http://www.nytimes.com/2010/05/03/technology/internet/ 03facebook.html.

Rip, E. (2011).
Impermium launches social spam index; finds that up to 40 percent of online ids are fake.
TechCrunch.
https://techcrunch.com/2011/08/31/impermium-launches-social-spam-index-finds-that-up-to-40-percent-of-online-ids-are-fake/.

Robertson, S. E. and Jones, K. S. (1976).
Relevance weighting of search terms.
Journal of the Association for Information Science and Technology, 27(3):129–146.

Rosvall, M., Axelsson, D., and Bergstrom, C. T. (2009).
The map equation.
The European Physical Journal Special Topics, 178(1):13–23.

Rosvall, M. and Bergstrom, C. (2007).
Maps of information flow reveal community structure in complex networks.
Technical report, Citeseer.

Rosvall, M. and Bergstrom, C. T. (2008).
Maps of random walks on complex networks reveal community structure.
Proceedings of the National Academy of Sciences, 105(4):1118–1123.

Russell, S. and Norvig, P. (1995).
A modern approach.
Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 25:27.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998).
A bayesian approach to filtering junk e-mail. learning for text categorizaton-papers from the aaai workshop.
Madison Wisconsin. AAAI Technical Report WS-98-05, pages 55–62.

Scott, S. (2017).
The fake americans russia created to influence the election.
New York Times.
https://www.nytimes.com/2017/09/07/us/politics/russia-facebook-twitter-election.html.

Sculley, D. and Wachman, G. M. (2007).
Relaxed online svms for spam filtering.
In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 415–422. ACM.

Sebastiani, F. (2002).
Machine learning in automated text categorization.
ACM computing surveys (CSUR), 34(1):1–47.

Short, J., Williams, E., and Christie, B. (1976).
The social psychology of telecommunications.
John Wiley and Sons Ltd.

Silvestri, G., Yang, J., Bozzon, A., and Tagarelli, A. (2015).
Linking accounts across social networks: the case of stackoverflow, github and twitter.
In *International Workshop on Knowledge Discovery on the WEB*, pages 41–52.

Simon, K. (2017).
Digital in 2017: Global overview.
Weare Social.
https://wearesocial.com/special-reports/digital-in-2017-global-overview.

Solorio, T., Hasan, R., and Mizan, M. (2013a).
A case study of sockpuppet detection in wikipedia.
In *Workshop on Language Analysis in Social Media (LASM) at NAACL HLT*, pages 59–68.

Solorio, T., Mizan, M., and Hasan, R. (2013b).

Sockpuppet detection in wikipedia: A corpus of real-world deceptive writing for linking identities.
arXiv preprint arXiv:1310.6772.

Statista, T. (2015).
Number of unique u.s. visitors to wikipedia.org from may 2011 to april 2015 (in millions).
Statista.
https://www.statista.com/statistics/265119/number-of-unique-us-visitors-to-wikipediaorg/.

Stein, T., Chen, E., and Mangla, K. (2011).
Facebook immune system.
In *Proceedings of the 4th Workshop on Social Network Systems*, page 8. ACM.

Stone, B. and Richtel, M. (2007).
The hand that controls the sock puppet could get slapped.
New York Times.
http://www.nytimes.com/2007/07/16/technology/16blog.html.

Strehl, A. and Ghosh, J. (2002).
Cluster ensembles—a knowledge reuse framework for combining multiple partitions.
Journal of machine learning research, 3(Dec):583–617.

Stringhini, G., Mourlanne, P., Jacob, G., Egele, M., Kruegel, C., and Vigna, G. (2015).
Evilcohort: detecting communities of malicious accounts on online services.
In *24th USENIX Security Symposium (USENIX Security 15)*, pages 563–578.

Sture, N. (2010).
Fake accounts in facebook - how to counter it.
Ezine Articles.
http://ezinearticles.com/?id=3703889.

Technology (2012).
Facebook has more than 83 million illegitimate accounts.
BBC News.
http://www.bbc.com/news/technology-19093078.

Thomas, K., Grier, C., Ma, J., Paxson, V., and Song, D. (2011a).

Design and evaluation of a real-time url spam filtering service.
In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 447–462. IEEE.

Thomas, K., Grier, C., Song, D., and Paxson, V. (2011b).
Suspended accounts in retrospect: an analysis of twitter spam.
In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 243–258. ACM.

Tran, N., Li, J., Subramanian, L., and Chow, S. S. (2011).
Optimal sybil-resilient node admission control.
In *INFOCOM, 2011 Proceedings IEEE*, pages 3218–3226. IEEE.

Tran, N., Min, B., Li, J., and Subramanian, L. (2009).
Sybil-resilient online content voting.
In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 15–28, Berkeley, CA, USA. USENIX Association.

Tsikerdekis, M. and Zeadally, S. (2014a).
Multiple account identity deception detection in social media using nonverbal behavior.
Information Forensics and Security, IEEE Transactions on, 9(8):1311–1321.

Tsikerdekis, M. and Zeadally, S. (2014b).
Online deception in social media.
Communications of the ACM, 57(9):72–80.

Twitter, R. (2017).
The twitter rules.
Twitter.
https://support.twitter.com/articles/18311.

TwitterInc. (2012).
Shutting down spammers.
Twitter.
https://blog.twitter.com/official/en_us/a/2012/shutting-down-spammers.html.

Van, D. and Stijn, M. (2001).
Graph clustering by flow simulation.
PhD thesis, University of Utrecht.

Venkatadri, G., Goga, O., Zhong, C., Viswanath, B., Gummadi, K. P., and Sastry, N. (2016).
Strengthening weak identities through inter-domain trust transfer.
In *Proceedings of the 25th International Conference on World Wide Web*, pages 1249–1259. International World Wide Web Conferences Steering Committee.

Viswanath, B., Post, A., Gummadi, K. P., and Mislove, A. (2010).
An analysis of social network-based sybil defenses.
ACM SIGCOMM Computer Communication Review, 40(4):363–374.

Vrij, A., Granhag, P. A., and Porter, S. (2010).
Pitfalls and opportunities in nonverbal and verbal lie detection.
Psychological Science in the Public Interest, 11(3):89–121.

Wang, G., Konolige, T., Wilson, C., Wang, X., Zheng, H., and Zhao, B. Y. (2013).
You are how you click: Clickstream analysis for sybil detection.
In *Usenix Security*, volume 14.

Wang, G., Mohanlal, M., Wilson, C., Wang, X., Metzger, M., Zheng, H., and Zhao, B. Y. (2012a).
Social turing tests: Crowdsourcing sybil detection.
arXiv preprint arXiv:1205.3856.

Wang, G., Wilson, C., Zhao, X., Zhu, Y., Mohanlal, M., Zheng, H., and Zhao, B. Y. (2012b).
Serf and turf: crowdturfing for fun and profit.
In *Proceedings of the 21st international conference on World Wide Web*, pages 679–688. ACM.

Weinberg, B. D. and Pehlivan, E. (2011).
Social spending: Managing the social media mix.
Business horizons, 54(3):275–282.

Whittaker, C., Ryner, B., and Nazif, M. (2010).
Large-scale automatic classification of phishing pages.
In *Network and Distributed System Security (NDSS)*, volume 10, page 2010.

Willis, L. E. (2014).
Why not privacy by default.
Berkeley Technology Law Journal, 29:61.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016).
Data Mining: Practical machine learning tools and techniques.
Morgan Kaufmann.

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I. (2008).
Spamming botnets: signatures and characteristics.
ACM SIGCOMM Computer Communication Review, 38(4):171–182.

Xu, L., Zheng, X., and Rong, C. (2013).
Trust evaluation based content filtering in social interactive data.
In *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, pages 538–542. IEEE.

Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B. Y., and Dai, Y. (2014).
Uncovering social network sybils in the wild.
ACM Transactions on Knowledge Discovery from Data (TKDD), 8(1):2.

Yardi, S., Romero, D., Schoenebeck, G., et al. (2009).
Detecting spam in a twitter network.
First Monday, 15(1).

Yu, H., Gibbons, P. B., Kaminsky, M., and Xiao, F. (2008a).
Sybillimit: A near-optimal social network defense against sybil attacks.
In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17. IEEE.

Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. D. (2008b).
Sybilguard: defending against sybil attacks via social networks.
IEEE/ACM Transactions on networking, 16(3):576–589.

Zackn, W. (2017).
Twitter has a spam bot problem — and it's getting worse.
ZD Net.
http://www.zdnet.com/article/twitter-spam-bot-problem-on-the-rise/.

Zafarani, R. and Liu, H. (2013).
Connecting users across social media sites: a behavioral-modeling approach.
In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 41–49. ACM.

Zardi, H., Romdhane, L. B., and Guessoum, Z. (2016).
Efficiently mining community structures in weighted social networks.
International Journal of Data Mining, Modelling and Management, 8(1):32–61.

Zephoria, T. (2017).
The top 20 valuable facebook statistics.
Zephoria.
https://zephoria.com/top-15-valuable-facebook-statistics/.

Zheng, X., Lai, Y., Chow, K., Hui, L. C., and Yiu, S. (2011a).
Detection of sockpuppets in online discussion forums.
Hong Kong University CS Tech Report. TR-2011-03. Available at: www. cs. hku. hk/research/techreps/document/TR-2011-03. pdf.

Zheng, X., Lai, Y. M., Chow, K.-P., Hui, L. C., and Yiu, S.-M. (2011b).
Sockpuppet detection in online discussion forums.
In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2011 Seventh International Conference on*, pages 374–377. IEEE.