UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Dottorato in Informatica

Relational Learning Models for Social Network Analysis

A dissertation presented
by
DANIELE MACCAGNOLA

Submitted in partial fulfillment of the requirements
for the degree of
DOCTOR of PHILOSOPHY

Supervisor: *Prof. Enza Messina*
Tutor: *Prof. Carla Simone*

October 2015

To my parents

Abstract

Social networks have been studied for nearly half a century by sociologists to analyze interactions between people. Nowadays, with the advent of Web 2.0, social networks have moved from being an abstract concept to actual online applications such as Facebook, Twitter and Linkedin, which are used daily by people to create and maintain relationships with friends, co-workers and other acquaintances. However, online social networks allow their users to do more than just maintain friendships: people can generally create and share content in various forms, from simple textual messages (called posts) to photos, videos, audios and much more.

Several approaches in *Social Network Analysis* have been proposed in the years to extract knowledge from social networks, addressing tasks that ranges from understanding how users create and modify their relationships, to finding the most influential people in a group, to understanding the ideas and opinions expressed by people in their posts. Many techniques from the field of *Machine Learning* have been used to address these problems. While some of them exploit the relationships among users, others focus on the content generated by the users, typically by analyzing the textual content written in the posts. These approaches, however, are generally unable to exploit both, in this way ignoring a consistent part of information available in social networks. The field of *Relational Learning* tries to overcome this limitation, by extending traditional approaches in order to use both sources of information, and thus achieve better performances.

In this thesis, I propose new Relational Learning approaches that address two tasks in Social Network Analysis. The first task is *Community Discovery*, which objective is to detect groups of users that share strong connections (e.g. working in the same company, attended the same school, etc.) or sharing the same interests. While this task is generally addressed by considering only the network structure, adding the user content can allow to increase the performance. The second task is *Opinion Detection*, which objective is to infer the opinion of users about a specific topic (politics, likeness of a brand). This task is typically addressed using user textual content, but the relationships can provide additional insights that allow to improve the inference of users' opinions. The experimental investigations reveal that network structure and user-generated content provide complementary information, and that using both sources of data can improve the performance of algorithms in both community discovery (a structure-based task) and opinion detection (a content-based task).

Publications

Journals

Maccagnola, D., Pozzi, F. A., Fersini, E. and Messina, E. (2015). *INCA: an Interest-based Clustering Algorithm for Community Detection in Social Networks*. Data Mining and Knowledge Discovery, Springer (*submitted*).

Conferences

Maccagnola, D., Fersini, E., Djennadi, R., and Messina, E. (2015). *Overlapping Kernel-Based Community Detection with Node Attributes*. 7th International Conference on Knowledge Discovery and Information Retrieval. Lisbon, November 2015.

Nozza, D., **Maccagnola, D.**, Guigue, V., Messina, E., and Gallinari, P. (2014). *A Latent Representation Model for Sentiment Analysis in Heterogeneous Social Networks*. In Software Engineering and Formal Methods (pp. 201-213). Springer International Publishing.

Pozzi, F. A., **Maccagnola, D.**, Fersini, E., and Messina, E. (2013). *Enhance user-level sentiment analysis on microblogs with approval relations.* In AI*IA 2013: Advances in Artificial Intelligence (pp. 133-144). Springer International Publishing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interactions between people have been studied for nearly half a century by sociologists, who were (and still are) interested in determining and predicting their behavior. From these studies, the need arose to develop a general concept to represent these interactions: J.A. Barnes [1] formalized it for the first time, naming it *Social Network*. This definition allowed to represent people and different types of interactions between them, from friendship to marriages, co-working and so on.

More recently, with the advent of Internet first and Web 2.0 later, social networks are not just an abstract concept anymore. Several platforms have been developed, called *Online Social Networks* or *Social Network Services*, that allow people to create and maintain their personal relationships. Famous examples of these platform are Facebook, Twitter, Linkedin and Google Plus, but many other have been developed in the last years. Nowadays, the term ”Social Network” is generally used to refer to these platforms, rather than the network of people underlying them.

These platforms allow the people using them (generally referred to as ”users”) to search for their friends and acquaintances, and to add them to their personal network. However, (online) social networks are not limited to relationships. The main strength of platforms such as Facebook and Twitter is the possibility to produce and share content with friends. From simple text to images, videos and music, huge amount of content is generated and shared every day by social network users.

Many scientists, from sociology first and other disciplines (such as mathematics, statistics and computer science) later, studied the data available from social networks with the general objective of gaining knowledge about its users. This field of study, called *Social Network Analysis*, was originally develop to understand and predict the relationships between users. Most of its basic tasks are related to the structure of the network. For example, we might want to know how

the relationships between two people change over time, or what does a group of people represent. To answer these questions, most existing approaches focus uniquely on the relationships between people. However, the content generated by users may reveal additional knowledge: for instance, two people may be interested in creating a new relationships not just because they have common friends, but also because they are both interested in the same sport.

On the other hand, some tasks in Social Network Analysis are focused on knowledge that can be drawn from the user-generated content. For example, we might want to know which are the main interests of a user, or whether he likes a certain kind of movies, or even whether would he vote for a particular politician. Approaches for addressing these tasks are generally based on techniques from text mining, and try to gain knowledge about the users by analyzing what he writes on his personal page (typically called ”posts”). Many methods, however, focus only on the text and disregard the fact that social networks users are connected to friends and acquaintances that can influence their interests and opinions. If I have friends that share posts about adventure movies, I might interested in them as well, or I might become interested in the future.

There is a need for approaches that can deal with both user content (usually in form of text) and relational structure (generally represented using graph structures). Most approaches to address social network analysis problems are drawn from the fields of Graph Theory and Machine Learning. While the former lack the capability of dealing with content, the latter lack methods to deal with the graph structure.

Recent research [2–4] have proposed to extend traditional Machine Learning methods with a relational representation, leading to a new field of study called *Relational Learning*. While many approaches have been proposed in literature, very few have been actually applied to Social Network Analysis.

1.1 Thesis Contributions

In this thesis, we propose to extend Machine Learning approaches for Social Network Analysis based either only on relational structure or user-generated content, in such a way that they can exploit both sources of information.

We consider two different Social Network Analysis tasks: *Community Discovery* and *Opinion Detection*.

The first task has the objective of detecting groups of users in a social network that form a community, in the sense that they share strong connections (they all belong to the same workplace, or same school) or they all share the same interests. While this task is generally addressed by

considering only the relational structure of the network, we argue that user-generated content can provide additional information that can improve the performance of community detection algorithms.

The second task has the objective of inferring the opinion of social network users about a specific topic, for example in order to understand if they hold a positive or negative opinion over a certain brand or politician. This task is generally addressed by traditional Machine Learning methods that focus on the text generated by the users. While the textual content is fundamental to determine the opinion of a user, we argue that relationships have an important role in influencing and thus determining the ideas and opinions of people in social networks.

For this reason, this thesis presents four contributions to address these tasks:

- In the first contribution, we extend an unsupervised approach for Community Detection based on graph-theory in such a way that it can exploit the additional information provided by user-generated content.
- The second contribution provides a new definition of community: while the one used in the first contribution was based only on relationships, this new definition is based on the interests of the users. Afterwards we introduce an unsupervised approach that extends a textual clustering approach with relational structure to detect interest-based communities.
- The third contribution introduce a semi-supervised approach for detecting the opinion of social network users. This model is based on two components, one determined by the textual content of the user's posts, and one determined the the opinion of neighboring users (who are more likely to influence the user's opinion).
- The fourth contribution introduces a model that, differently from the previous one, is able to predict the opinion of both posts and users at the same time. The model exploits a latent representation that allow to compare the two different kinds of entities.

1.2 Thesis Outline

The thesis is organized as follows. In Chapter 2, we introduce the field of Social Network Analysis and its most prominent tasks. In this chapter we also introduce important concept of Graph Theory and the basis of Relational Learning, that are needed in the following discussion. In Chapter 3 we introduce the contributions relative to the task of Community Detection, together with a discussion relative to the existing approaches to address this task. In Chapter 4 we present the contributions relative to Opinion Detection. Finally, in Chapter 5, conclusions are drawn.

Chapter 2

Social Network Data Analytics

Networks of people have been studied over several decades with the general objective of analyzing their relationships, aiming to discover informative patterns among these interactions. The idea of ”social network” has been used to describe complex relationships between members of social systems at different levels. The term was formalized by J.A. Barnes [1] to represent patterns of ties among people, for example groups like tribes or families, or social categories such as gender or ethnicity.

Analysis and studies on such social networks, performed by sociologists interested in determining interactions and behaviors of people, have traditionally been conducted manually using time expensive and difficult methods. A classic example is the *six-degrees-of-separation* experiment performed by Milgram [5], who studied a network of people connected by postal mail, in order to test whether two arbitrary people could be connected by a limited number of edges (specifically, up to six edges). Of course, these experiments required a lot of effort and had a very low response rate (think about the time required to send that many postal mails). They were also biased by the limited availability of people who agreed to participate to these trials, and by the scarce reliability of their response rates.

Only recently, with the explosion of internet and the so called ”Web 2.0”, social networks have moved from being a metaphor to actual internet applications, generally called *online social networks*. Examples of these networks are Facebook, LinkedIn and Twitter. Online social networks have known a sudden increase in popularity in the last years: people who register to these services can get in touch with their friends and acquaintances to discuss, or follow public pages belonging to famous people or organizations, forming bond and interactions regardless of geographical limitations. In addition, other means of online communications, such as Skype and Google Hangout, and several websites for sharing media content, such as Flickr and Youtube, can also be considered as a form of social network (some of them have been effectively integrated into actual online social network services).

Social networks are a valuable source of data about users, providing a lot of information regarding their habits, their behaviour, their interests and preferences. First of all, social networks provide data about relationships among users, which can give insights about the people a user share interests with, both in real life or chatting online. These relationships can be used to discover how communities of users are formed or what they represent, e.g. groups of friends, co-workers, people attending the same school, etc.

But social networks are not only about relationships. Online systems like Facebook and Twitter have struggled to provide their users new ways to *generate content*. The most simple content available is of course *text*: people write the most disparate variety of things on their favourite social network page, commenting the latest news they have read on a newspaper, or discussing about famous people they have seen on TV, or they just write something personal, like what they are doing or expressing personal emotions.

While noisy and huge, this textual content is a rich source of information about users, and it can be exploited for social and marketing research in several ways. It can be used to tell whether a person likes a certain brand of beverage, what he thinks about that new smartphone that will be shortly released on the market, or even who will he likely vote at the coming political elections.

Social networks, of course, allow the users to add more than just textual content, letting them to upload various media types such as images, audio tracks and videos. Multimedia content is often linked on social networks from other websites (like Youtube for videos, Spotify and other music providers for audio tracks, and many others for images). The analysis of multimedia content is generally much harder than textual content. However, the simple fact that a user is listening to a certain song, or is watching a certain video on Youtube may be indicative of his behavior, or be used to find other people who share similar interests.

All the content published and generated by users is, however, not intended to be only read and seen by other users - social networks allow people to do more than just that. In general, a user is usually allowed to add comments to another user's post. This comment can be plain text, but sometimes can contain multimedia like a normal post - this is allowed, for instance, both in Facebook and in Twitter.

Other actions are also available to social network users. They can approve another user's published content, for example by using the "like" button in Facebook, or the "+1" command in Google Plus. Some social networks also allow to dislike a content. They can also share that content with their own friends and followers - this action is called "share" in Facebook and "retweet" on Twitter. These actions can be seen as another form of relationship between users (e.g. user "Alice" shared 5 posts generated by user "Bob"), between a user and a post (e.g. user "Alice" approved post "Go Pats!"), or even between posts (e.g. post "No way" is a reply to post "Obama will win!").

Collecting the actions performed by users can complement the knowledge obtained from their friendship relations and published content.

The amount of information available in social network from relationships, generated content and users' behavior can be a valuable source of knowledge. *Social network analysis* is the field of study that deals with extracting this knowledge, using techniques and methods from graph theory, statistics and machine learning.

Social network analysis is a wide field, which addresses a number of different tasks that arise from the information provided by social networks. These tasks can be roughly divided into two macro-groups: those where the focus is the topological structure of the network itself, and those where the focus is the content generated by the users.

- **Tasks based on structure** In this groups fall problems that deal with the analysis of the network connections, in order to determine important nodes, communities, links, and evolving regions of the network.

  These include methods to perform a statistical analysis of the network itself, looking for properties that a "typical" social network should display. The distribution of connections in the network, for instance, can reveal how the users communicate among them, whether through a few "hubs" or via a more balanced distribution of connections. Important discoveries have been done by studying the structure of real networks, such as the "small world phenomenon" [5, 6], which indicates that people are generally divided by up to six degrees of separation. More recently, by studying how networks evolve over time, researchers have discovered that network diameters tend to shrink, as the number of connections get more dense [7].

  Other tasks based on the network structure concern with the ranking of nodes by their importance, usually to understand which users are important in the network. Many methods developed for this task are commonly used in practice. Starting from the well known PageRank algorithm that was developed for ranking of web pages, many others have been studied to rank the importance of users and pages in social networks.

  One of the most investigated tasks is *Community Detection*, which objective is to identify tightly-connected regions of the social network, which likely represent communities of people that share some sort of social relationship. This task is related to the more general problem called *graph partitioning*, where a graph is split into subparts based on the density of its links. Many approaches have been applied directly from graph partitioning to community detection, with mixed results. Social networks present several specific characteristics, namely the behaviour of its users, the dynamics of their connections, the additional content they produce, that traditional graph partitioning methods cannot easily manage.

Another important task based on structure relates with the detection of experts, also called opinion leaders. They are important actors in the network that generally gather a large number of connections, and whose behavior is thought to influence that of other users. Many approaches developed for detecting opinion leaders are based on the level of connections between the users, from a simple count of their direct links, to more complex measures of their reachability in the whole network.

Finally, much research have been devoted to the task of link prediction, which focuses on inferring the presence of links that are not directly observable, or the prediction of new links that may be are likely to be generated in the future (or can be suggested to a user by a recommender system).

- **Tasks based on content** A second group is composed by tasks primarily focused on content-based knowledge extraction from social network data. Most of the huge amount of content generated by social network users is text, usually in form of relatively short messages (called *posts*). Data mining techniques (and, more specifically, text mining approaches) are often used to extract knowledge from these short messages.

  Many textual content-based tasks are concerned with the inference of characteristics related to single users or single posts. In many applications, some of the nodes may be labeled with information relative to the single user. Among those, *opinion detection* aims at determining whether a post expresses a positive or negative opinion (or sentiment) about a certain topic. For instance, this may be used for understanding the political opinion of social network users before an election, usually based on the post he wrote or the opinion of his acquaintances. Many marketing applications can take advantage from these approaches to study the opinion of users on specific products before or after they are launched on the marked.

  These approaches are also used to infer other characteristics of the users, whether they are real users or fake ones (spam bots). Some works have also proposed content-based methods to infer personal data such as gender or age, using the text a user posts on his web page.

  Finally, the multimedia content shared by social network users has been studied as well to address many problems usually connected to the placement of tags, which are short descriptions that are attached by users to various media like images and videos. These brief descriptions are not only useful to understand the actual content of these media, but they also give insights on the actual interests of the users who placed them, modified or even only visualized them.

In order to address these tasks, many approaches have been proposed in literature, drawn from fields like machine learning, probabilistic reasoning and graph theory. A wide amount of the

existing approaches, however, suffer of a major limitation: they exploit the information provided either by the topology of the network, or by the user-generated content, but not both. Most of the structure-based tasks are addressed by considering only the topology of the network. However, the textual content can be helpful when deciding, for instance, if a user belong to a certain community or another. On the other hand, content-based tasks are uniquely addressed by considering the textual content generated by the users. However, using topological information (i.e. with whom a user is connected) can be very informative when inferring, for example, which are the interests of a user.

Traditional machine learning methods are generally unable to address these tasks using both content and structure data. Tasks such as community detection are approached by techniques drawn from graph theory, which are developed to exploit the topological structure of the network, but are not thought to be applied to textual content data. On the other hand, tasks such as opinion detection are addressed by techniques which were conceived for plain text data - documents, books, news pieces, etc. - which do not show any kind of relationship, tie or dependency between them. Traditional machine learning methods for text mining are not fit to analyze social network content, not without incurring in wrong assumptions of independency.

For these reasons there is a need of new machine learning approaches that can overcome these limitations. These new methods must be able to combine the different types of data provided by the two sources, user-generated content (typically textual data) and topological structure (typically lists of connections between objects).

In recent years, a new line of research approached this problem by combining machine learning with other techniques from different fields, such as probabilistic reasoning and relational representation. This field, generally called *Relational Machine Learning*, encompasses a large number of different approaches.

In this thesis the focus will be the application of relational learning approaches to two of the most important tasks available in social network analysis: the structure-based task called *community discovery* and the content-based task called *opinion detection*. More details are given in Chapter 3 and 4.

The rest of the chapter is organized as follows: first, in Sec.2.1 we introduce concepts of graph theory that will be used in all the discussed applications of social network analysis. In Sec. 2.2 we give a brief introduction to the field of machine learning and its major techniques. Subsequently, we explain how the relational information can be included into traditional methods.

2.1 Graph Theory

Graphs are mathematical structures that are widely used in many fields of science to represent pairwise bonds between objects.

Social network analysis leverages many concepts of graph theory to analyze the relationships among people. First of all, a social network is usually represented by a graph, which is defined as follows:

Definition 1. *A graph is a set $G = (V,E))$, where V is the set of nodes, and E is the set of edges with $E = \{(x,y)\,|x,y \in V\}$.*

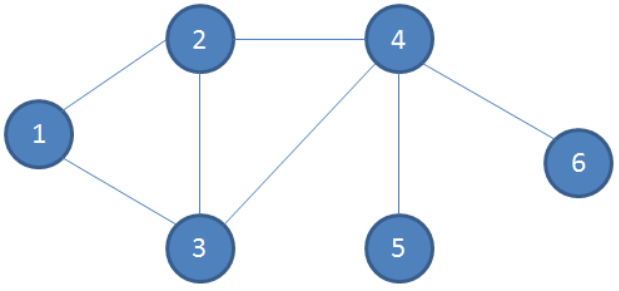
For example, if we want to represent a social network as a graph, we can use the nodes to represent users and the edges to represent relationships among those users.

A graph can be also represented by an *adjacency matrix*, defined as:

Definition 2. *Given a graph $G = (V,E))$, the relative adjacency matrix is a binary matrix A of size $n \times n$, with $n = |V|$. The entry A_{ij} of the matrix has value 1 if there exists an edge $E(i,j)$, 0 otherwise.*

Many variants of graphs have been used to represent the many aspects of social networks.

For instance, relationships among people can be either one-way or mutual. In the same way, a graph may be *directed* or *undirected*. If the graph is undirected, the edges $E_{1-2} = (V_1,V_2)$ and $E_{2-1} = (V_2,V_1)$ are the same. This usually happens on social networks like Facebook, where a friendship relationship is usually mutual and thus it is represented by an undirected edge. If the graph is directed, instead, edges E_{1-2} and E_{2-1} are two distinct entities, which can both exist separately. This is the case with social networks like Twitter, where the following relationship can be one-sided (user one follows user two, but not viceversa).



(a) Graph

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	1	0	0
3	1	1	0	1	0	0
4	0	1	1	0	1	1
5	0	0	0	1	0	0
6	0	0	0	1	0	0

(b) Adjacency Matrix

FIGURE 2.1: Example of graph (a) and relative adjacency matrix (b).

Another distinction between types of graph is related to the importance given to its edges. If every edge has the same importance , the graph is considered *unweighted*. Under these circumstances, the edges can be viewed as binary variable - either an edge exists, or it does not exist. Unweighted graphs can be used to represent situations where the edge does not assume a particular importance compared to others, e.g. a friendship network in Facebook.

In other cases, each relationship may have a different relevance. For example, there may be multiple edges occurring between two nodes (e.g. number of links between two web pages), or the connection between them may contain additional information (e.g. monetary transactions between two entities). In the field of social networks, this can be useful to describe many kind of relationships among users. For instance, we may want to measure the number of posts written by a user explicitly mentioning another user, or the number of times a user has approved another user's post.

In order to describe this information, we need a *weighted graph*, which is defined as follows:

Definition 3. *A weighted graph is a set $G^{w_e} = (V,E,W))$, where V is the set of nodes, E is the set of edges with $E = \{(x,y)\,|x,y \in V\}$, and W is the weight function $W : E \to \mathbb{R}$.*

Weights $w(i,j)$ with $(i,j) \in E$, also denoted as w_{ij}, are generally real and positive numbers, but there are also cases where the weight takes only integer values.

In literature, weighted graphs are often also called *edge-weighted graph*, indicating that the weight is relative to the graph's edges. However, there are situations when also the nodes of the graph contain important numerical information, e.g. the content of posts written by a social network user. Therefore, we require a broader definition of weighted graph that encompasses both edge and node weights.

Definition 4. *A full weighted graph is a set $G^w = \left(V,E,W^V,W^E\right))$, where V is the set of nodes, E is the set of edges with $E = \{(x,y)\,|x,y \in V\}$, W^V is the node weight function $W^V : V \to \mathbb{R}$ that assigns a numerical value to each node, and W^E is the edge weight function $W^E : E \to \mathbb{R}$ that assigns a numerical value to every edge.*

Finally, graphs can be *homogeneous* or *heterogeneous*. The majority of social networks studied in literature are homogeneous, meaning that the nodes are all of one kind - network of friends, web pages connected by hyperlinks, papers in a citation network, etc.

However, this is not always the case. Several social networks are inherently structured as a heterogeneous network, meaning that they are composed of different types of nodes - take for example the movie-actor graph of IMDB, or the user-image network of Instagram and Flickr. Sometimes, even social networks which are usually represented by homogeneous graphs can benefit by a heterogeneous representation. For instance, a Twitter network, which is usually

composed of nodes representing its user, can be improved by adding nodes representing each user-generated post.

2.2 Machine Learning

Machine learning is a field of computer science that studies the use of algorithms to learn how to perform tasks or make predictions. Algorithms are sequences of instructions that are executed to transform an input into a specific output, developed to solve a broad range of problems. For example, to sort a sequence of numbers, it is possible to develop an algorithm that takes these numbers as input and returns the ordered list as output. Several algorithms have been devised to solve this specific problems, thus the main issue is to choosing the most efficient one, in terms of time and/or memory.

However, some tasks still cannot be solved by a specific algorithm. Take, for example, the detection of spam in emails: the input is the sequence of characters forming the email, while the output is a simple answer saying ”the mail is spam” or ”the mail is not spam”. However, we do not know how to transform these characters into the wanted answer.

While we lack the knowledge required to devise such an algorithm, we have plenty of data regarding emails: we can easily collect a dataset where each email is labeled as ”spam” or ”not spam”. What we need is the computer to ”learn” to discriminate among them, that is, we want the machine to automatically generate an algorithm to solve this task.

While we do not need a machine to learn how to sort numbers (we already know how to do that), for tasks like spam detection we do not have algorithms, but we can *”learn by examples”*. Underlying the example data, we believe there is a process that explains why some emails are spam and some are not. While we do not know the details of this process, we know it is not completely random, but shows some kind of regular behavior.

We can therefore build a *good and useful approximation* of the underlying process. While this approximation can explain only a part of the data, we can identify patterns that can be exploited to create the wanted algorithm. *Machine learning* is the field of study which deals with detecting such patterns, generate models that can explain the data, and use them to make predictions.

Machine learning finds applications in several fields, where large amount of data are processed to construct simple models, which can provide additional useful knowledge. Application areas are several: in sales, machine learning methods can be used to analyze and predict the behavior of the customers; in finance, models can be generated to predict trends in the stock market, or to detect frauds in the transactions; in medicine, they can be used to improve diagnosis or to model

the effectiveness of drugs; in other science fields, like biology or physics, huge amount of data can be analyzed to find hidden patterns.

Machine learning approaches tackle a wide range of different tasks, of which classification, regression, clustering are among the better known ones. These tasks can be grouped into two macro-categories: *supervised learning* and *unsupervised learning*. These two categories will be detailed in the following.

2.2.1 Supervised Machine Learning

In the spam detection problem, a (often very large) number of emails has to be analyzed to understand if each of them contains a legitimate message or it just includes unwanted spam content (e.g. advertisements or scams). The objective is to distinguish between legitimate emails from spam using their characteristics (or *features*) like the name or address of the sender, or the words used in the message. Starting from a set of emails manually annotated by humans, a machine learning method can fit a model to the labeled data to be able to recognize whether a new email is spam or not.

Spam detection is a *classification* problem, where there are two *classes*: spam and not spam. The content of the email form the *input* of the classifier, whose task is to assign the input to one of the two classes. Classification is one the most common examples of supervised learning task. It is called *supervised* because the machine learning method trains with data which has been previously *labeled*, thus following the "supervision" of some prior knowledge.

More formally, given a set of n examples of the form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, such as x_i is the *feature vector* of the i-th object, and y_i is its label (or class), the objective of classification algorithm is to find a function $g : X \to Y$ that maps the input space X to the output space Y.

In the spam detection example, x_i may be the vector which shows how often particular words are used in an email, and y_i is the label that indicates if the email is spam or legitimate.

The set of examples is usually split into two subsets: a *training set* that is used to train the model, and a *test set* that is used to validate the capability of the model to apply to objects it has never seen.

By training with the labeled data, a classifier fits a model to that data. The model can then assume several forms, for example it can be composed of a set of rules (like in decision trees [8]) or be a probability function (like in Naive Bayes classifiers [9]). The generated model can then be used to predict the class of a new instance based on its characteristics.

2.2.2 Unsupervised Machine Learning

Supervised learning makes use of a labeled dataset to train the model. However, in many situations, labeled data is not readily available, often because there is no supervisor or expert that can label the data. We still have a set of n examples of the form $\{x_1, \ldots, x_n\}$. However, we do not have any observed label y_i to train on.

Even without supervision, however, the input data can still be analyzed to find patterns and regular structures. One method to find such regularities is called *clustering*. Its aim is to find groups of data objects, called *clusters*, where objects belonging to the same cluster are "similar" among them (for some definition of similarity), while objects in different clusters are "not similar".

The process of finding the best clusters is generally described as an optimization function, where the objective is to minimize the distance between objects belonging to the same cluster, or between those objects and a representative of the cluster called "centroid" (which is usually computed as the average of all the objects belonging to the cluster).

Formally, this optimization function can be written as:

$$\min \sum_{k=1}^{K} \sum_{x_i \in C_k} d(x_i, c_k) \tag{2.1}$$

where $k = 1, \ldots, K$ is the number of clusters, C_k is the k-th cluster and c_k its centroid. The distance $d(\cdot, \cdot)$ can be computed in several ways - the most common is arguably the Euclidean distance, but many others are used.

An example is given by document clustering, where the objective is to group similar documents based on their topic. In case of news articles, they can be subdivided into groups such as politics, sports, crime news, fashion, etc. Differently from a classification task, we *do not* know how which topics are present in the dataset, and often we do not even know the number. Each document is represented by a vector of features, which usually depend on the words appearing in it. The most common representation is the *bag of words*, where a vector of numbers is used to indicate the words appearing in the document. Similar documents will have a similar representation in terms of this vector, that is, documents in the same cluster will likely contain the same words. Of course, which words have to be considered for the bag of words, and how to compute the similarity between documents are critical choices for obtaining meaningful clusters.

Clustering is in general affected by the many choice a user has to take: how many clusters are required, which distance measure to use, which features of each object are relevant for discriminating the groups.

2.3 Dealing with relational data

Traditional machine learning deals with input data that takes the form of elements described by a fixed number of attributes, which take a specific range of values (binary, integer, real, or even categorical values). This setting is sometimes addressed as *Attribute-Value learning* [4], because the learning is performed on the attributes and values of each element of the dataset.

When considering attribute-value learning, the data is assumed to be *independent and identically distributed* (i.i.d.). That means that every element in the dataset assumes values independently from each other.

While for several domains this description provides a sufficiently valid model for the data, this is not always true. The elements of a dataset may be connected by relationships, violating the assumption of independence - the attributes of an element may be dependent on the attributes of another element.

Consider for example the task of web-pages classification. This task shares many similarities with the task of document classification - both documents and web pages are described by the set of words appearing in them, and in both cases we want to find the topic of discussion. However, there is a very important difference: web-pages are connected by means of *hyperlinks*, and the topic of a page is likely dependent on the topic of the pages to which it links.

If a machine learning method that assume i.i.d. data is applied to this task, its learning performance will probably be negatively affected [10]. On the other hand, relationship provide additional knowledge that can be exploited by a suitable machine learning algorithm to improve its performance.

2.3.1 Mapping relational data to propositional data

Several real-world datasets show a certain degree of relations within them. Relational databases, for instance, are often formed by a number of tables with some degree of dependency.

For sake of simplicity, we introduce the following example, detailed in table 2.1. A class of students attends several courses, each of which is held by a professor. Each student may or may not have passed the exam of a certain course, and we want to model the likelihood of a student succeeding a given exam.

A machine learning algorithm could take the characteristics related to each student (their age, iq, etc.) to estimate their success in a given exam. However, a better approach would be to take into considerations the relationships between student and course, and course and professor.

TABLE 2.1: Example dataset of students, courses and teachers.

Student	Course	Grade	...
Bran Stark	Mathematics	A	...
Bran Stark	Chemistry	B	...
Jon Snow	Chemistry	F	...
Jon Snow	Literature	C	...
Margaery Tyrell	Literature	A	...

Course	Teacher	...
Chemistry	Petyr Baelish	...
Mathematics	Tywin Lannister	...
Literature	Davos Seaworth	...

TABLE 2.2: Example of propositionalization of dataset in Tab. 2.1.

Student	Course	Grade	Teacher	...
Bran Stark	Mathematics	A	Tywin Lannister	...
Bran Stark	Chemistry	B	Petyr Baelish	...
Jon Snow	Chemistry	F	Petyr Baelish	...
Jon Snow	Literature	C	Davos Seaworth	...
Margaery Tyrell	Literature	A	Davos Seaworth	...

For example, we might want to add a characteristic which indicates the how many years of experience the professor of that course has, or how many students usually pass the exam.

This approach, which consists of generating attribute-value data from relational data, is generally called *propositionalization*. Several propositionalization methods have been proposed in the past few years, most of them directly transforming the relational structure into an attribute-value one.

The most common is the *table-based approach*. It consists of generating a single table containing all the relevant characteristics related to the element (in a database, that usually requires a JOIN of all the relevant tables). Continuing from the example given above, we can generate a table where the student is associated to each course and each professor (see Table 2.2).

One advantage of this approach is that all the traditional machine learning algorithms can be applied to the propositionalized problem. However, the main disadvantage is that this problem might be incomplete - some information might be lost in the transformation. For instance, in a propositionalized problem it is not possible to describe how neighbor elements affect each other.

While many attempts have been done to deal with relational data using traditional attribute-value approaches, there is no general way to map the relational information into a finite number of attributes - thus enabling traditional machine learning - without losing information.

For this reason, relational learning approaches that avoid propositionalization have been proposed, as detailed in the following section.

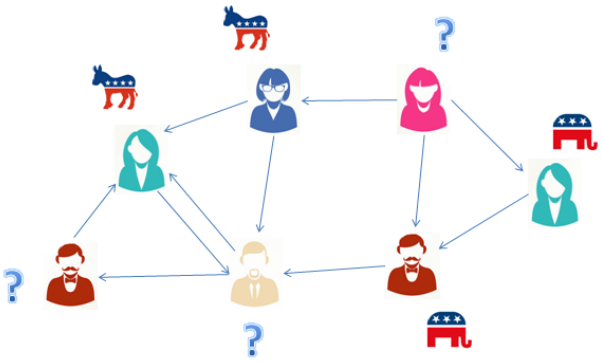


FIGURE 2.2: Example of collective classification task. Differently from traditional classification, some of the unknown instances depend on at least another unknown instance.

2.3.2 Relational Learning

Most machine learning methods can deal with data in the attribute-value format. When dealing with relational data, transforming it to attribute-value data cannot be done without losing information. Therefore, we can choose one of these two options: either we apply this transformation, accepting the information loss, and be able to apply traditional machine learning approaches; or we use an algorithm that can handle relational data directly, without propositionalization.

Main Tasks in Relational Learning Dealing directly with relational data makes it possible to consider new tasks that are direct extensions of the tasks in traditional machine learning. While several of them exist, the most important (and the ones that will be considered in this thesis) are *collective classification* and *link-based clustering*.

Collective classification refers to the combined classification of elements connected together by some relationship. The class label of an element is assumed to be correlated not only to the element's attributes, but also on the (often unseen) class labels of its neighbors.

Collective classification (see Figure 2.2) is a combinatorial problem defined as follows. Given a graph $G = (V, E)$, each node $v \in V$ can also be seen as a random variable that can take a value from an appropriate domain. V is further divided into two set of nodes, a first set of nodes V_{known} for which we know the correct values (observed variables) and a set of nodes V_{unknown} whose value is unknown and must be determined. Note that V_{known} and V_{unknown} can be considered like training and test set in traditional classification. The labels of a node $v_u \in V_{\text{unknown}}$ depend on its neighborhood $N(v_u) = \{v_j | (v_u, v_j) \in E\}$. The task is to assign a label to these nodes, considering that nodes in $N(v_u)$ can belong both to V_{known} and V_{unknown}. Not all neighbors are observed variables, therefore the problem of collective classification falls in the category of *semi-supervised learning* problems.

Several studies shown that performing collective classification in a relational setting lead to better results compared to independent (traditional) classification [2, 11–14].

The second objective in relational learning derives directly from unsupervised learning, and specifically from clustering. While clustering aims at finding groups of objects based on their characteristics, *link-based clustering* takes into consideration the relationships among these objects. This task is closely related to graph clustering and community detection, which focus on the detection of subgraphs (or set of nodes) with a high degree of connectivity.

In general, the goal of link-based clustering is to divide the data set into clusters such that the elements assigned to a particular cluster are similar or connected in some predefined sense.

Several methods have been proposed to address this task. Given that most of these approaches have been developed using graph clustering techniques, they tend to focus uniquely on the relationships among objects - disregarding in part, or completely, the contribution of objects characteristics.

Many approaches have been developed to address relational learning problems. In the following, we give a brief description of the main classes of techniques used to model and analyze relationships in a machine learning environment.

Relational learning approaches in the state of the art The approaches presented in literature to tackle the problem of relational learning are several, and they can be distinguished in several classes. Here we report the most important categories (for a full review, see [4]).

- **Inductive Logic Programming** In Inductive Logic Programming (ILP) [15], input and outputs are described by first-order predicate logic. The relationships between elements of the dataset, and even the connection between an element and its attributes, are defined using logic rules.

  For example, we may describe a student-course relationship by using a set of facts (in a logic language such as Prolog), as follows:

  ```
  student(s1, 'Jon Snow').

  course(c1, 'Math 101').

  course(c2, 'Chemistry 101').

  attends(c1,s1).

  grades(s1, c2, 'C').
  ```

  which means that student Jon Snow attends the basic course on Math, and he has already completed the basic course on Chemistry obtaining a grade of 'C'.

Starting from the background knowledge defined by these rules, ILP aims at detecting patterns (”concept learning”) that can be used for several tasks. This concept can be tied to a specific n-ary relation or predicate (e.g. to define a classification rule), or without a specific predicate (e.g. to discover frequent patterns in the dataset).

- **Graph Mining**

  Graphs are widely used in many fields of science to represent pairwise bonds between objects, where nodes connected by edges are used to represent many different things.

  Many relational learning methods have based their approach on graphs [16, 17], which serve as a comprehensive representation of the relationships among elements. For example, in a web mining context, nodes of a graph can represent web pages and the edges the hypertext links between them.

  Graph-based learning systems can be of two types: in the first case, approaches where the whole dataset can be represented by a single graph, where each node is an instance of the dataset connected by a relation to other instances. In the second case, each instance is represented by a graph, and the goal is usually to compare characteristics of these graphs to find common patterns (find frequent substructures, etc.).

  Generally, however, when we talk about relational learning we refer to the first case, where we aim at analyzing instances for which the assumption of independence does not hold. In this case, graph-based learning systems aim at predicting properties of existing nodes or edges in the graph (”node/edge classification”), or the existence of edges not yet observed (”link discovery”).

  Graph-based methods generally differ from other relational learning methods because they focus mostly on the relational structure of the graph, often disregarding properties of the single nodes (instances of the dataset). They might consider node and edge labels (usually the goal is to predict them), but they often ignore other attributes.

- **Multi-relational Data Mining**

  Multi-relational data mining methods approach the relational learning task from a database perspective. While attribute-value methods learn from the data of a single table, *multi-relational* methods use information spread over different tables of a database. In literature, mainly rule-based learners and decision tree classifiers have been proposed [18]. These methods are usually thought to be integrated with (large) relational databases, hence they focus most of their effort on efficiency and scalability.

  However, these methods often reduce the data spread over multiple tables into a single tuple. As explained for propositionalization, this approach is usually not possible without loss of information.

- **Statistical Relational Learning**

  The above approaches are generally able, in different ways, to deal with the relational structure of a dataset. However, they are not designed to deal with *uncertainty*, a phenomena that naturally occurs in any application of machine learning, both traditional and relational. Inference and learning in these models are generally not based on probability theory.

  In order to overcome this limitation, models have been proposed to address relational learning using statistical methods, such as Bayesian networks [19] or Markov networks [20]. This field of study, referred to as *Statistical Relational Learning* (SRL), is a subdiscipline of artificial intelligence and machine learning that focuses on the development of models used in domains characterized by uncertainty (which is addressed by statistical methods) and relational structure. SRL takes ideas from probabilistic reasoning, relational representation and machine learning.

  Typically, the methods developed in SRL makes use of probabilistic graphical models or probabilistic logic to describe the relational structure of data. In the last decade, several SRL approaches have been proposed to address relational learning from different points of view. Many approaches seem to differ only in their syntax, but some methods allow to express certain knowledge more easily than others.

  In the following, we report the best known approaches in SRL. They can be roughly divided into two categories: the Probabilistic Relational Models, which rely mostly on graphical models like Bayesian Networks or Markov Networks, and Probabilistic Logic, which relies on probabilistic extensions of first-order logics. This distinction, however, is not sharp, as graphical models and logic are often used together.

  Probabilistic relational models (PRMs) [3, 21] extend Bayesian networks with the concept of objects, which are characterized by properties and relations between them. A PRM defines a template for a probability distribution over a database: this template includes a relational component to describe the relational schema of the database, and a probabilistic component that describes the probabilistic dependencies that hold in the domain. A PRM, together with a database of objects, defines a probability distribution over the attributes of these objects.

  Similar formalisms are provided by Relational Bayesian Networks [22], which were developed independently from PRMs, and by Entity-Relationship Probabilistic Model [23], which generalize the first model.

  Another approach in SRL is given by Probabilistic Logic. The integration of first order logic reasoning with probabilistic inference is a challenging task that has been widely researched over the last years. The progress obtained in this field has not, however, lead to a good convergence of the practical formalisms proposed in literature.

The strength of these approaches relies in the strong theoretical foundations, because logic and probabilities has been definitely studied more than the expressiveness of PRMs and probabilistic ER models.

Several probabilistic logic methods have emerged in literature. Bayesian Logic Programs (BLPs) [24] combine Bayesian networks with first-order logic reasoning to improve the inference power of both methods. Logical Bayesian networks [25] are a variant of BLPs. Their main difference is that the structure of the network is defined in a deterministic way, while in LBNs the structure can be inferred by a probabilistic process.

Finally, another probabilistic logic framework is given by *Markov Logic Networks* [13], that extend Markov networks using first order logic.

While the expressive power of these approaches is very high, so is the computational cost required to perform learning and inference on them.

In particular for the field of social network analysis, which often deals with very large amount of data, this is a key feature. Therefore, we will need to find a good trade-off between the computational complexity and the expressive power of these methods.

In this thesis, in order to improve relational learning for social network analysis, we will take inspiration from some of the existing methods, in particular graph-based methods and probabilistic relational models.

Chapter 3

Unsupervised Relational Machine Learning for Community Discovery

3.1 Introduction

Methods for network structure analysis have been extensively studied in order to reveal patterns that can improve the knowledge relative to network users. Fortunato et al. [26] argue that graphs representing real social networks are neither completely ordered (like lattices), nor completely disordered (like random graphs [27]), thus suggesting that the distribution of edges in real networks can be informative of the behavior of its participants. The distribution of edges in networks is often inhomogeneous, both on a global and on a local scale. Groups of vertices concentrate a high amount of edges, while low concentrations of edges are present between these groups.

This inhomogeneous organization of networks is called *community structure* [28] and it reflects the complex organization of social relationships, representing families, groups of co-workers or friends, or even virtual groups of people with similar interests.

Communities, also called clusters or modules, are ”groups of vertices which probably share common properties and/or play similar roles within the graph” [26].

Detecting such communities is one of the most relevant problems in the context of social network analysis. This task, called community detection or *community discovery*, has been addressed by many researchers in various fields, from sociology to biology, but it still has not been adequately solved.

The identification of communities is not an easy task. One of the most prominent reasons is the lack of an universally accepted definition of community. Informally, a community can be

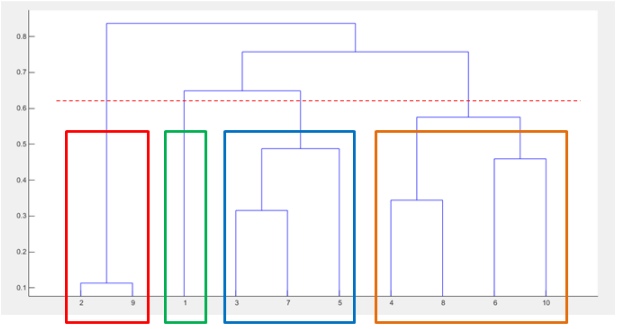


FIGURE 3.1: Example of dendrogram. The dashed line indicates the cut, while the four colored rectangles indicates the resulting four clusters.

defined as a set of nodes which are densely connected among them and sparsely connected to other nodes. However, several definitions have been proposed in literature, as this general criterion can be satisfied in many ways. In social networks, such a community corresponds to a set of users that shows some structural properties based on the underlying network (e.g. friendships in Facebook and Google+, following/follower in Twitter).

Many definitions describe communities as separate subgroups of the networks that cover the entire network (every node is assigned to one and only one community).

However, not all definitions are designed to represent communities as clean partitions of the network. For example, communities might have a hierarchical order, where nodes are organized in small sets, which in turn form larger groups, and so on. The hierarchical organization is usually represented as a tree structure called *dendrogram*. Single nodes are placed at the bottom, and by moving up the tree the nodes are aggregated into increasingly larger groups. The height at which two groups of nodes are aggregated is an indicator of how similar they are (lower height indicates a higher similarity). A dendrogram can also be cut at a given height to obtain a partition of the network (see Fig. 3.1).

Another problem is that vertices placed at the boundary of a group of tightly connected notes are often difficult to assign to a community or another. A way to solve this issue is to allow nodes to belong to more than one community, thus creating *overlapping communities*. This is an important feature for a community detection algorithm, as it is common for users in a social network to participate to multiple communities.

Moreover, it is not straighforward to decide the best partition (or hierarchy). Finding a good measure for comparing communities is still a challenge. A first intuitive approach would be to evaluate the density of edges within and between communities, by counting the number of edges between nodes belonging to the community (*intracommunity edges* or *intra-density*) and

the number of edges between a node in the community and nodes outside the community (*intercommunity edges* or *inter-density*). A good community would have a high intra-density and a low inter-density.

However, many other definitions have been introduced in literature to refine this coarse principle. Fortunato et al. [29] classify them into three categories:

- *Local Definitions.* Each community is evaluated by considering only its nodes (*self-referring approach*) or at most their immediate neighbors (*comparative approach*). One notable example of self-referring definition is the *clique*, which is defined as a maximal subgraph where each node is adjacent to all the others. The simplest instance of a clique is a triangle, which is commonly found in real networks and therefore the most used as indicator of community. A relaxed definition is *n-clique*, which limits the distance of each pair of nodes in the subgraph to n. Two other definitions are *k-core* and *k-plex*: in the former, each node must be adjacent at least to k other vertex of the subgraph; in the latter, each node must be adjacent to all others, except at most k of them. Two comparative definitions are that of *strong community* and *weak community*: the first defines a community as a subgraph where every node has more neighbors inside the subgraph than on the inside; the second is equivalent to the definition above based on intra- and inter-density.

- *Global Definitions.* Communities are evaluated as part of the whole graph. Global definitions are usually based on concept of *null model*, that is, a graph with similar topological features as the original one (e.g. the number of nodes and edges) but without a community structure. One simple example is the null model proposed by Girvan and Newman [28], which defines another graph with the same number of nodes and edges as the original one, but where the edges are rearranged in a random way. Girvan and Newman use this null model to define a measure called *Modularity* to evaluate the quality of the communities (more details will be given in Sec.3.2).

- *Definitions based on vertex similarity.* While the previous definitions have as objective the detection of communities based only on the network structure, a community can be also determined by the similarity of its nodes, evaluated through a quantitative measure. The similarity can be computed in several ways, depending on the attributes associated to each vertex. In a social network, a node representing a user can be associated to several types of features: for instance, they may represent specific words written by the user in his posts. The notion of similarity would in this case represent users who employ the same words.

Deciding the definition of community is a necessary step before choosing the most appropriate approach to be used. In general, definitions based on the topology of the network are used (local

or global definitions), and therefore use algorithms that are based purely on graph theory, or otherwise based on the network structure only.

A limited number of approaches have focused on definitions that are linked to the content generated by the network users (vertex similarity definitions), requiring traditional data mining methods to address the problem of content analysis. A lot of valuable information is encoded in the contents shared by the users. In fact, contents provide very specific information about the nature of the relationships of connected users: sharing similar contents is an indication of affiliation to the same group.

Recently, however, some methods have been proposed to exploit both these features and the topology of the network to improve results. A definition of community that includes both network structure and user content, however, still has not be defined.

After communities in a network have been identified, it is necessary to evaluate how accurate the task of detection has been performed. This task is similar to the evaluation of clusters in classic data mining, as it presents the same challenge as ground truth might not be always available. Therefore, we have to consider two possible scenarios: when ground truth is available and when it is not.

In the first case, we have a some knowledge of how communities should be formed. In general, the task here is composed of two steps: comparing the number of detected communities with the number provided by the ground truth, and compare the actual content of the detected communities with the ground truth communities. This task is generally performed using measures from the field of Information Retrieval: *Precision, Recall and F1-Score*. A more detailed descriptions of these measures will be provided in Sec. 3.3.3.

In the second case, when ground truth is not available, we can still evaluate the quality of the inferred communities. One possibility is to analyze the characteristics and attributes of the nodes belonging to a community to see if community members show some degree of coherency. In case of social networks, we might compare the users in a given community by looking at their posts, profile information and generated content. This comparison is generally performed by human subjects who compare the word frequency of each community and determine whether these keywords represent a coherent topic. A more focused and single-topic set of keywords represents a coherent community. Often, tag clouds are used to demonstrate these topics (see Fig. 3.2.

Semantic analysis of the communities is a qualitative way to evaluate communities. A quantitative approach is also possible, using clustering quality measures. This method also allows to compare the performance of two or more community detection algorithms and determine the preferred one. A detailed example of both clustering quality measures and semantic evaluation of the communities will be shown in Sec. 3.4.3.

FIGURE 3.2: Example of word cloud.

In the rest of the chapter we provide a more detailed overview of the existing approaches for community detection. Then, we propose two new approaches to improve the state of the art: in the first contribution of this thesis, we partially address the limitation of the existing approaches by extending a graph-based approach by adding user-generated content to the model. This approach is based on a traditional concept of community (based on the relationship between users), allowing us to use existing datasets for comparing the performance.

In Sec.3.4 we introduce INCA, the second contribution of this thesis, a clustering-based approach that aims at discovering interest-based communities by combining structure and content into a single model. This approach focuses on a definition of communities connected to user interests, for which no ground truth dataset is available. For this reason, we will introduce new evaluation measures to allow performance evaluation.

3.2 Related Work

Over the years, many approaches have been proposed to address the task of community detection. As discussed above, the method strongly depends on the definition of community itself: most definitions are based on the structure of the network - either considering the entire structure or only the neighborhood of each node; some, instead, are based on the concept of ”similarity” between nodes, determined by the content produced by the users.

Three macro-categories of community detection algorithms are therefore available: methods based only on structure, based only on content, and based on both data sources. In the following, we give an overview of the most important algorithms for each category.

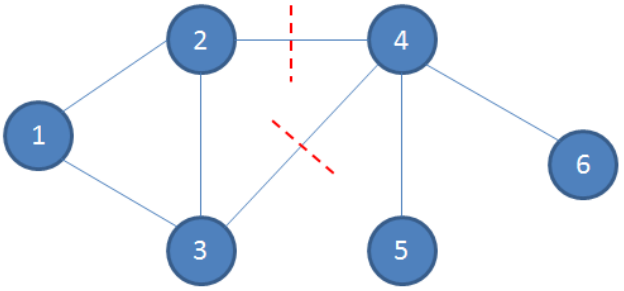


FIGURE 3.3: Example of graph cut. By removing the edges (2,4) and (3,4), the two communities are isolated, thus their cut size is 2.

3.2.1 Methods based on structure

Community detection has been mostly addressed by considering only the topology of the network, using concepts from the field of graph partitioning. Several methods have been proposed, based on very different approaches and different concepts of community. Among the most important, we can mention the methods based on quality measures (such as Cut and Modularity), methods based on edge removal, and spectral clustering methods.

The problem of graph partitioning has been initially addressed by partitioning methods that consider a "cost" the removal of edges from the network. Their objective is to remove those edges that lie between groups of nodes considered "dense", but minimizing the cost. The number of edges that must be removed to completely separate two groups of nodes is called *cut size* (see Fig. 3.3).

A graph partitioning algorithm would then separate the nodes into n groups such that the cut size between those groups is minimum, i.e. minimizing the number of edges to "*cut*".

Two quality measures are often used to evaluate the quality of a cut: *Normalized Cut* [30] and *Conductance* [31]. The Normalized Cut of a group of nodes $V' \subset V$ is computed as the sum of weights of the edges that connect V' to the rest of the graph, normalized by the total edge weight of V' and the total of the rest of the graph $\overline{V'} = \{V - V'\}$.

Given the adjacency matrix A, the Normalized Cut is defined as:

$$\text{Ncut}(V') = \frac{\sum_{i \in V', j \in \overline{V'}} A(i,j)}{\sum_{i \in V'} \text{degree}(i)} + \frac{\sum_{i \in V', j \in \overline{V'}} A(i,j)}{\sum_{j \in \overline{V'}} \text{degree}(j)} \tag{3.1}$$

Conductance is a closely related measure, defined as follows:

$$\text{Conductance}(V') = \frac{\sum_{i \in V', j \in \overline{V'}} A(i,j)}{\min\left(\sum_{i \in V'} \text{degree}(i), \sum_{j \in \overline{V'}} \text{degree}(j)\right)} \tag{3.2}$$

The Normalized Cut or Conductance of a graph partition is then given by the sum of the measures for each community.

It has been shown that optimizing these objective functions is NP-hard, so heuristics are usually applied [30, 32, 33]. One of the most frequently used algorithms for graph partitioning is the Kernigan-Lin (KL), which represents the difference between the number of edges inside a community and the number of edges between the communities, under the constraint that all communities have the same size. The objective function is defined as:

$$\text{KL}(C_1, \ldots, C_k) = \sum_{i \neq j} A(C_i, C_j) \quad \text{subject to } |C_1| = |C_2| = \cdots = |C_k| \tag{3.3}$$

where $A(C_i, C_j) = \sum_{u \in C_i, v \in C_j} A(u,v)$ is the number of edges between two communities C_i and C_j.

The algorithm starts by splitting the graph into two partitions of the same size, either randomly or following some prior knowledge. At each iteration, the algorithm swaps the same number of nodes from one partition to the other, aiming at increasing the value of KL. The choice of the nodes to move is taken using a greedy approach. In order to avoid being trapped in local maxima, the algorithm allow sometimes to decrease KL. The final partition will be the one with the highest value of KL obtained during the iterations. The Kernigan-Lin algorithm is fast (with a computational time of $O(|V|^2)$ in the worst-case), but the resulting partitions heavily depend on the starting configuration (this still makes it a good approach for refining partitions obtained from other methods).

Another method for finding communities has been proposed by Newman and Girvan. They introduce an efficient measure for the goodness of a graph partition, called *Modularity*, which rapidly gained popularity. This measure has the advantage of being independent of the number of communities. The basic idea behind modularity is that a good community should have a distribution of the edges is different from the distribution of the null model. In particular, modularity is the number of the edges that fall into a community, minus the expected number if edges were distributed at random. More formally, modularity is usually referred to as Q, and defined as:

$$Q = \sum_{i,j=1}^{k} \left[\frac{A(C_i, C_j)}{|E|} - \left(\frac{\text{degree}(C_i)}{2|E|} \right)^2 \right] \tag{3.4}$$

where C_i is the i-th community, degree(C_i) is the total degree of the i-th community, and m is the total number of edges in the network.

Several algorithms have been proposed to get closer to the maximum value of modularity. Newman [34] proposed an heuristic for optimize modularity, using a greedy agglomerative approach. At each iteration, two groups of nodes are merged to form a larger community, provided that the modularity of the network increases after the merge. Initially, every node forms a community (singletons), and at every iteration the algorithm chooses the best two communities to merge. The best implementation of this algorithm can be performed in $O(md\log n)$ time, where d is the height of the dendrogram describing the hierarchical structure of the communities.

While this greedy approach is currently the faster method to optimize modularity score, the performance of other methods is generally higher.

One more advanced approach for optimizing modularity is Simulated Annealing, which is a probabilistic method commonly used for optimizing functions in several fields. Simulated annealing performs an heuristic search in the space of all possible solutions (in this case, representing all the possible partitions of the graph) to find the global optimum.

If agglomerative algorithms started from a number of communities equal to the number of nodes, and gradually merged them into larger groups, the class of divisive algorithms have the opposite approach. They start from the full graph as one big community, and progressively split it into smaller communities.

One good example of divisive algorithm has been proposed by Newman and Girvan [35] and it is based on the concept of *edge betwenness*. Edge betweennness evaluates the likelihood that an edge lies between two communities, by measuring how many times this edge is crossed by a shortest path between two nodes. If the edge is traversed often, its edge betweenness value will be high, indicating that this edge connects two communities. Hence, by removing these edges we might be able to separate a network into its communities. The algorithm proposed by Newman and Girvan computes the betweenness score for all edges, then chooses the one with the highest value and removes it. The scores are recomputed, and the cycle iterates until a chosen number of communities is obtained.

While edge betweenness can be computed using other measures (examples are *random-walk betweenness* and *current-flow betweenness*), Parthasarathy et al. [36] argue that this has only a slight impact on the final result.

This algorithm, however, is very expensive from a computational cost point of view, requiring $O(|V^3|)$ time. Tyler et al. [37] proposed a modified version of the algorithm that requires a lower computational time. They compute edge betweenness only for a random subset of node pairs,

obtaining an estimate of the true betweenness. By repeating the calculations many times, they are able to obtain good results while maintaining a lower computational cost.

Other algorithms propose alternative measures of centrality ([38], [39]), trying to find a tradeoff between good performance and low computational time.

Finally, another important approach is given by *spectral clustering* [40]. Spectral clustering is very commonly used not only for community discovery, but also for data clustering in many fields. These methods analyze the eigenvectors of the adjacency matrix to find the best assignment of nodes to communities.

The main matrix used in spectral clustering is the *Laplacian matrix* $\mathcal{L}$. The unnormalized Laplacian is computed as $L = D - A$, where A is the adjacency matrix and D is the degree matrix (a matrix with the degree of each node on the diagonal). The normalized Laplacian $\mathcal{L}$ is computed as $\mathcal{L} = D^{-\frac{1}{2}}(D-A)D^{-\frac{1}{2}}$. Both L and $\mathcal{L}$ are symmetric and positive definite, therefore they have real and positive eigenvalues.

Traditional spectral methods focus on the first k eigenvectors of the adjacency matrix, a low-dimensional representation which holds most of the information regarding the similarity between nodes. This representation can be then processed using classic clustering methods like K-Means.

The main disadvantage of spectral clustering is the computational complexity of calculating eigenvectors. In general, a traditional implementation requires $O(|V|^3)$ time, while faster implementations can reach a complexity of $O(|C| \cdot |V| \cdot t)$, with t representing the number of iterations.

Other algorithms have been proposed to exploit the spectral representation of the network. Capocci et al. [41] use a normalized version of the adjacency matrix. Wu and Huberman [42] propose a method which consider the network as a "resistor network", where each node is considered to be a resistance. Their algorithm computes "voltages" using approximations of the Kirchoff's equations, and the communities should be revealed by differences in voltages between nodes. The method they propose has the advantage of requiring a lower computational time, which can reach $O(|V|\log|V|)$ if a sampling is performed.

Other methods have been used for community detection. For a more detailed review of the existing methods, see Parthasarathy et al. [36] and Fortunato et al. [29].

3.2.2 Data Clustering methods

Nodes can be classified in communities not only depending on their connection in the graph, but also depending on other measures of similarity between them. We can compare nodes using a measure that depends on the features (or attributes) each node is associated to. Examples of

these features are the user birth place and the user gender or age, but many more types of features can be considered. Among the existing measures, the most frequently used in the field of social network analysis are the *euclidean distance* and the *cosine similarity*.

Euclidean distance is a measure of dissimilarity between the attributes of two nodes, computed as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{|X|} \left(X_{ik} - X_{jk}\right)^2} \tag{3.5}$$

where X is the set of attributes of a node. Two nodes will be more similar if their euclidean distance is low.

Cosine similarity is another traditional distance measure, mainly used in natural language processing. For this reason, it is often used when comparing textual features of the nodes, e.g. attributes representing the words used by the users of a social network.

Cosine similarity is defined as follows:

$$\text{cossim}_{ij} = \frac{X_i \cdot X_j}{\|X_i\|\|X_j\|} = \frac{\sum_{l=1}^{|X|} \left(X_{il} \times X_{jl}\right)}{\sqrt{\sum_{l=1}^{|X|} \left(X_{il}\right)^2} \times \sqrt{\sum_{l=1}^{|X|} \left(X_{jl}\right)^2}} \tag{3.6}$$

Cosine similarity takes value between 0 and 1, where 0 indicates that the attributes of two nodes i and j are completely different, while 1 means that the attributes are the same.

Once a distance measure has been chosen, clustering is usually performed using a *partitional* algorithm (like K-Means) or a *hierarchical* algorithm [43].

In the first case, the number of clusters k is predetermined. The nodes of the graph are embedded in a metric space which depends on their attributes. K-Means identifies k points in this space, called *centroids*, such that the distance of each node to its closest centroid is minimal. The initial choice of the centroids is usually random, and each node is associated to each closest centroid to form a cluster. At each iteration, centroids are recalculated as mean of nodes in each cluster, then the nodes are moved with respect to the new centroids. The procedure ends after a number of iterations, or when the centroids do not change anymore. It is important to note that the result of K-Means are not stable, and they heavily depends on the initial choice of the centroids.

Hierarchical algorithms do not require to know in advance the number of clusters. They produce a dendrogram. Agglomerative hierarchical clustering (the most widely used) begins with singleton clusters (clusters containing only one node), and at each iteration chooses to merge the

pair of clusters with the highest similarity (or the lowest distance). The final clusters are chosen by ”cutting” the dendrogram at a certain height.

While hierarchical clustering does not require to know in advance the number of clusters, it still requires to choose the ”cutting height”, which can be equally difficult. Moreover, it computational cost is usually higher than K-Means.

3.2.3 Methods based on both structure and attributes

The approaches detailed up to this point either address the problem of discovering communities by focusing on the network topology, or by comparing node features using similarity measures.

Several hybrid approaches have been proposed in the literature to address community detection exploiting both contents and structural relationships. These works can be divided into two categories: approaches based on generative probabilistic models, and approaches based on partitional or discriminative models.

In the first group we find approaches that are based on joint models mostly based on Latent Dirichlet Allocation (LDA) [44] or Probabilistic Latent Semantic Analysis (PLSA) [45]. While LDA and PLSA have been designed to model words in documents (mainly for topic modeling), these approaches focus on joint modeling of both content and structure of the network. These generative probabilistic models consider both contents and links as being dependent on one or more latent variables, and then estimate the conditional distributions to find community assignments.

Hu et al. [46], for instance, propose a probabilistic Social Topic model that classifies the users of a social network based on their interests (topics). Their model, however, considers topological features (such as the number of neighbors) but disregards the actual connections among users. On the other hand, models such as those proposed by Natarajan et al. [47] and by Yang et al. [48] explicitly describe user-user relationships as dependent on a latent variable.

The second group of approaches is formed by partitional or discriminative methods, which main goal is to find communities according to either an objective function or a conditional probability model that combine textual and structural information.

For example, the algorithms proposed by Zhang et al. [49] and by Gupta et al. [50] propose a measure of similarity among users: this similarity is higher if two people use the same words used in their posts, but also if they share the same friends or they reply to posts generated by the same user. The algorithms try to maximize the similarity of users belonging to the same community.

Although the above mentioned investigations represent a fundamental step towards the understanding of emerging communities in online social networks, they still suffer some important limitations. In particular, they do not consider:

- **Edge strength and directionality** The main characteristics of the hybrid approaches is that they exploit undirected and unweighted social relationships.However, strength and orientation of social relationships can be very informative to derive more accurate community affiliation.
- **Informal language and network sparseness** Most of the works are based on a generative paradigm to model both relationships and language of user-generated contents. Concerning the contents, these models could be biased by the words that are irrelevant with respect to the topic of interest underlying a given target community. Regarding the relationships, an open problem is the sparseness of relations in a real-world settings. Many microblogs on social networks exhibit a relatively low edge density, with a significant consequence on generative models of link structure. Since the main goal of these approaches is to estimate distribution parameters given a sample of edges, the fitness of the model decreases according to the sparseness of the graph [51].
- **Scalability** Most of the existing approaches do not scale with respect to the size of the social networks that we regularly observe in the real world. The time complexity underlying these approaches make unfeasible their application to massive datasets.

In the following sections we propose new algorithms that use both structure and textual content of social networks, with the objective of overcoming the outlined limitations.

3.3 Kernel-Based Community Discovery

Among the approaches proposed in literature (as detailed above), many do not consider that community structures of influential users (opinion leaders) are different from that of others. It has been shown in the literature that in many social network, especially online social networks such as Twitter, Facebook and Google Plus, the average degree of connections of opinion leaders is almost ten times more than other users [52].

Most of the approaches for community and opinion leader detection available in literature are based on the assumption that each influential user should be placed in a different community with its relative followers/friends. However, this assumption does not reflect the real world, where a community is likely to be composed of a kernel of several users (as opinion leaders) and a group of auxiliary members.

In order to define a community and detect its opinion leaders, the *community kernel detection* problem has been introduced in [52], composed of two subtasks: (1) the identification of kernel nodes, i.e. influential members of the network and (2) the identification of auxiliary nodes (non-influential members) and their association to a kernel to form a community.

In literature, very few approaches have been proposed to address this problem [52, 53]. Among these, one of the most promising is the Greedy - Weight-balanced Community detection algorithm (WeBA), which combines multiple steps to first identify the kernels, and subsequently the auxiliary nodes to form the communities.

However, this approach suffers from two important limitations:

- **Overlapping communities** Most actual social networks are made of highly overlapping cohesive subgroups of nodes, simply because individuals often belong to numerous different kinds of communities simultaneously [54]. Members of a network may participate in many social circles according to their interests, hobbies, and relationships connected to their educational background, working environment and family.

  WeBA does not take into account the possibility of overlapping communities when detecting the auxiliary nodes, that can be associated only to one kernel (each of them is assigned only to the most similar kernel).

  For this reason, we introduce a *Overlapping Auxiliary Community Detection* approach that can overcome the limits of the existing method.

- **Node Attributes** Existing approaches for community detection usually take into account only one source of information: the relationships among the network members, e.g. friendship or following/followee relationships.

  Social networks, however, often provide a large amount of information that is not directly included in the relationships. For example, online social networks like Twitter and Facebook allow their members to write and share textual messages (posts), which can be very informative attributes of the user representing interests and ideas.

  Still, most community detection algorithms do not exploit this information to improve their performance. The WeBA algorithm is based on the assumption that each member of a kernel has more connections to/from the kernel than a vertex outside the kernel does. However, this assumption does not consider that two users may share similar interests even when not directly connected by a relationship.

  Therefore, we introduce an improved version of the WeBA algorithm that includes both network structure and information from node attributes.

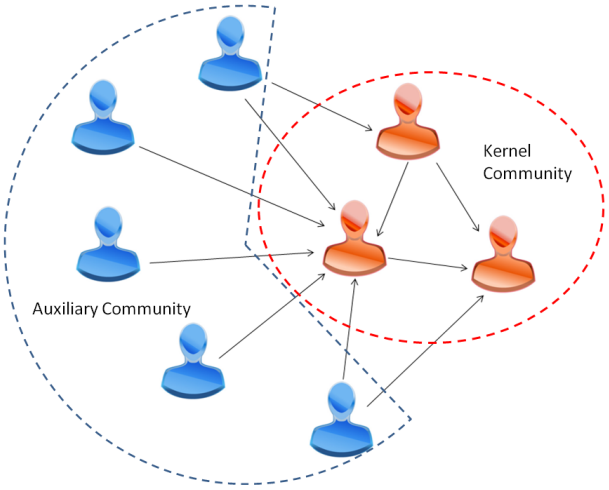


FIGURE 3.4: Example of kernel community (red members) and auxiliary community (blue members).

This section is organized as follows: first we provide the preliminary concept to define a kernel community; subsequently, we give the details of the algorithms that extends WeBA. Finally, we present the experimental settings and the obtained results.

3.3.1 Preliminaries

Before discussing the details of the proposed method, we introduce some important notations.

The task of Community detection aims at finding a set of communities $C = \{c_1, c_2, \ldots, c_k\}$, where communities c_i are formed by groups of vertices with dense intra-community connections, but sparse inter-community links.

As discussed above, communities are assumed to be composed by a kernel and an auxiliary community (see Fig.3.4 for an example). They are defined as follows:

Def: Kernel Community Given an oriented graph $G = (V, E)$, k disjoint subsets $\{K_1, \ldots, K_k\}$ of vertices are called kernel communities if:

$$|E(u, K_i)| \geq |E(v, K_i)| \wedge |E(K_i, u)| \geq |E(K_i, v)|, \quad \forall i \in \{1, \ldots, k\}, \forall u \in K_i, \forall v \notin K_i \quad (3.7)$$

where $E(A, B) = \{(u, v) \in E | u \in A, v \in B\}$ for $A, B \subseteq V$.

While the kernel compose the core of a community, the other nodes belonging to that same community are called *auxiliary*, and defined as follows:

Def: Auxiliary Community Given a set of kernel communities K, k associated subsets $\{A_{K_i},\ldots,A_{K_k}\}$ of vertices are called auxiliary communities if:

- $A_{K_i} \cap K_i = \varnothing, \forall i \in \{1,\ldots,k\}$;
- $|E(v,K_i)| \geq \left|E(v,K_j)\right|$,
  $\forall i \in \{1,\ldots,k\}, \forall j \neq i, \forall v \in A_{K_i}$;
- $|E(A_{K_i},K_i)| \geq |E(K_i,K_i)|, \forall i \in \{1,\ldots,k\}$.

For any $i \in \{i,\ldots,k\}$, each vertex in K_i is a kernel member and each vertex in A_{K_i} is an auxiliary member.

Node attributes are additional information that can be used to detect communities in networks. Node attributes are generally obtained from the user-generated content such as text. In order to model this information, we introduce a function $\tau(u) : u \rightarrow t^u$ which maps a network user $u \in V$ to its feature vector representation t^u as:

$$t^u = \left(t^u_1, t^u_2, \ldots, t^u_{|X|}\right) \tag{3.8}$$

where $|X|$ is the number of *attributes* shared by all the users. In our case, attributes can represent any kind of information related to the user (gender, age, job titles, etc.), denoted by binary values.

For any $u,v \in V$ represented as in Eq. 3.8, we derive a similarity matrix M, with $|M| = |V| \times |V|$, defined as follows:

$$M_{u,v} = \cos(t^u, t^v) = \frac{< t^u \cdot t^v >}{\|t^u\| \cdot \|t^v\|} \tag{3.9}$$

This similarity matrix will be used to include the influence of user-generated content into the proposed algorithm.

3.3.2 Algorithm

In order to overcome the limitations of the approaches reported in Sec. 3.2, we propose an extended and revised version of the kernel-based community detection algorithm WeBA, presented in [52].

This baseline algorithm consists of three main steps:

- A Greedy approach based on maximum cardinality search, aimed at finding l kernels nodes for each community with dense internal connections allowing also dense external relations;
- A Weight-balanced heuristic to tune the solution find by Greedy in order to revise the initial community of kernels taking into account information provided by the connection of non-kernel members;
- An Auxiliary Community Detection approach to find the auxiliary communities: it associates at each node a ranked list of kernels (kernel-based association).

In the following we detail the novel methods proposed in this section: first, we describe the new greedy and weight-balanced algorithms for exploiting node attributes in the detection of kernel communities; then, we introduce a variant of the Auxiliary Community Detection method that can detect overlapping communities.

3.3.2.1 Community detection with node attributes

A major limit of the existing algorithm is its inability to take into account all the sources of information available in the networks. Specifically, node attributes can be considered to improve the performance of the community detection task.

In order to improve the original algorithm shown in [52], we separately modify the procedures for *greedy* and *weight-balanced* as following:

Greedy Given an undirected graph $G = (V,E)$ and kernel size l, initialize a subset $S \subseteq V$ to be a random vertex $v \in V$. Then, iteratively enlarge S by adding the vertex with the maximum number of connections to S. If there are multiple vertices with the maximum number of connections to S, pick the one with the highest degree $d(u) = \sum_{v \in V} E(u,v)$ (if there are several nodes with the same highest degree, randomly pick one of them). This subroutine is repeatedly executed $O(|V|/k)$ times to obtain steady-state results and reduce the effect of the random selection of the initial point.

This original *Greedy* algorithm has been extended in order to take into account content similarity of nodes. The proposed algorithm takes as additional input the similarity matrix M (defined in Eq. 3.9), to evaluate how close are the attributes of each couple of nodes. When the algorithm selects a vertex u as kernel node, it will evaluate not only the number of edges $d(u)$, but also the similarity of contents among u and the all the other kernel members already assigned to the same kernel community. In particular, instead of evaluating only the degree $d(u)$ as indication of node importance, we define $p(u)$ as:

Algorithm 1 Revised Greedy

Input: $G = (V, E)$, similarity matrix M, kernel size l
Output: Community Kernels $K = \{k_1, k_2, \ldots, k_l\}$
$Y = V$
$K \leftarrow \varnothing$
repeat
  $S \leftarrow$ a random vertex $v \in V$
  while $|S| \leq l$ **do**
    $R^* = \left\{ \underset{u \notin S}{\text{argmax}} \left(\frac{|E(u,S)| + \sum_{t=1}^{|S|} M(u,t)}{2} \right) \right\}$
    if $|R^* = 1|$ **then**
      $S \leftarrow S \cup R^*$
    else
      $U^* = \left\{ u \in R^* | \underset{u \notin S}{\text{argmax}}\ p(u) \right\}$
      if $|U^* = 1|$ **then**
        $S \leftarrow S \cup U^*$
      else
        $S \leftarrow S \cup$ random $u \in U^*$
  if $S \notin K$ **then**
    $K \leftarrow K, S$
  $Y = Y - S$
until $Y \neq \varnothing$
return K

FIGURE 3.5: Pseudocode for the revised Greedy algorithm.

$$p(u) = \sum_{v \in V} \frac{E(u,v) + M(u,v)}{2} \tag{3.10}$$

The pseudo-code is reported in Fig.3.5.

Weight-Balanced Starting from the initial result generated by the *Greedy* algorithm, the kernels are refined and optimized by the *Weighted-Balanced* Algorithm. Given a kernel size l and an initial subset S to refine, the original WeBA algorithms assigns a weight $w(v) = 1$ to each vertex $v \in S$, and a weight $w(v) = 0$ to each vertex $v \notin S$. Let $N(v)$ be the set of neighboring vertices of v, i.e. $N(v) = \{u \in V | (u,v) \in E\}$. Then, at each iteration, the algorithm searches for a pair of vertices $u, v \in V$ satisfying both of the following relaxation conditions:

a) $w(u) < 1$

b) $w(v) > 0$

c) $nw(u) > nw(v)$

where $nw(u)$ is the neighboring weight of u, i.e. $nw(u) = \sum_{v \in N(u)} w(v) \cdot E(u,v)$.

Algorithm 2 Revised WeBA

Input: $G = (V, E)$, similarity matrix M, kernel size l
Output: Community Kernels $K = \{k_1, k_2, \ldots, k_l\}$
$K \leftarrow \varnothing$
$K_g \leftarrow \text{GREEDY}(G, M, l)$
for $S \in K_g$ **do**
  $\forall v \in S, \quad w(v) \leftarrow 1$
  $\forall v \notin S, \quad w(v) \leftarrow 0$
  while $\exists u, v \in V$ *satisfying the relaxation conditions a),b),c)* **do**
    $\alpha \leftarrow (1 - w(u)) \dfrac{E(u,v) + M(u,v)}{2}$
    $\beta \leftarrow w(v) \dfrac{E(u,v) + M(u,v)}{2}$
    $\gamma \leftarrow \dfrac{\text{nw}^*(u) + \text{nw}^*(v)}{2}$
    $\delta \leftarrow \min(\alpha, \beta, \gamma)$
    pick the pair u, v with the maximum δ value
    $w(u) = \min(w(u) + \delta, 1)$
    $w(u) = \max(w(v) - \delta, 0)$
  $C \leftarrow \{v \in V | w(v) = 1\}$
  if $C \notin K$ **then**
    $K \leftarrow \{K, C\}$
return K

FIGURE 3.6: Pseudocode for the revised Weight-Balanced algorithm.

Similarly to Greedy, also Weight-Balanced has been extended in order to deal with the content similarity. In order to include it, we consider the neighboring weight according to both links and content similarity:

$$nw^*(u) = \sum_{v \in N(u)} w(v) \cdot \frac{E(u,v) + M(u,v)}{2} \tag{3.11}$$

The pseudocode for the revised Weight-Balanced is reported in Fig. 3.6.

3.3.2.2 Overlapping Auxiliary Communities

The detection of auxiliary communities has been revised and improved to allow auxiliary communities to overlap. Given a node v, the proposed approach takes into account a *popularity measure* relative to v when choosing the auxiliary community A_{K_i}. In particular, v is associated to A_{K_i} if two conditions are satisfied:

- v is the node with the highest number of edges pointing to the community $C_i = \bigcup\{K_i, A_{K_i}\}$, i.e.

$$|E(v, C_i)| \geq |E(v, C_j)| \quad \text{for } j \neq i \tag{3.12}$$

Algorithm 3 Revised Auxiliary Community

Input: Community Kernels $K = \{k_1, k_2, \ldots, k_k\}$
Output: Auxiliary Communities $A = \{A_{k_1}, A_{k_2}, \ldots, A_{k_k}\}$
$\forall i \in \{1, \ldots, k\}, \quad A_{k_i} \leftarrow \varnothing$
repeat
  $C_i = \bigcup\{k_i, A_{k_i}\}, \quad \forall i \in \{1, \ldots, k\}$
  for $i \leftarrow 1$ **to** k **do**
    $S \leftarrow \{v \in C_i | \forall u \in \bigcup C_i, \forall j \in \{1, \ldots, k\},$
    (1) $|E(v, C_i)| \geq |E(v, C_j)| > 0$
    (2) $\sum_{n=1}^{k} |E(v, C_n)| \geq \sum_{n=1}^{k} |E(u, C_n)|\}$
    $A_{k_i} \cup S$
until *no more vertices can be added*
return A

FIGURE 3.7: Pseudocode for the revised Auxiliary Community detection algorithm.

- There is no other node $u \notin C_i$ such that u has more edges pointing to all the communities C_n than v, i.e.

$$\sum_{n=1}^{l} |E(v, C_n)| \geq \sum_{n=1}^{l} |E(u, C_n)| \tag{3.13}$$

While the first condition was included in the original version of the algorithm, the second one ensures that we consider first the nodes having a higher number of connections (as indication of popularity) to all the communities.

If both conditions are satisfied for more than one community C_i, the node is associated to all of the corresponding A_{K_i}.

In Fig. 3.7 we report the pseudocode for the algorithm.

The final communities C_i will be formed by the association of the kernel community K_i with the corresponding auxiliary community A_{K_i}.

3.3.3 Experimental Settings

Datasets description In order to evaluate the performance of the proposed kernel-based community detection method, we considered three benchmarks used in the state of the art:

- **Philosophers** The philosophers network [55] consists of Wikipedia articles about famous philosophers. Nodes represent Wikipedia articles about philosophers, and directed edges indicate whether one article links to another. The attributes of a given node u are represented by a binary indicator vector of out-links from node u to other non-philosopher

TABLE 3.1: Datasets statistics. N: number of nodes, E: number of edges, C: number of communities, K: number of node attributes, S: average community size, A: community membership per node.

Dataset	N	E	C	K	S	A
Philosophers	1546	7971	907	5770	6,86	6,87
Twitter	125120	2248406	3140	33569	15,54	0,39
Facebook	4089	170174	193	175	28,76	1,36

Wikipedia articles (e.g. if a philosopher page links to a Wikipedia article "Mathematician", the binary value of the attribute "Mathematician" for the corresponding philosopher will be equal to one). The Wikipedia network is formed by 1546 nodes and 7971 edges.

Moreover, Wikipedia provides categories (e.g. "Hindu philosophers", or "Austrian psychologists") for each article. We consider each category with more than five philosophers as a ground-truth community, obtaining a total of 907 overlapping communities.

- **Twitter** The Twitter network is a ego-network available from the Stanford Large Network Dataset Collection (http://snap.stanford.edu/data) [54]. The ground truth communities are obtained from Twitter "lists" manually labeled by the owner of the ego-network. Node attributes are defined by processing the tweets (posts) generated by each user of the network. We use a "bag of words" representation, where each binary attribute indicates that a specific word appeared in the user's tweets. In particular, we consider only specific words called "hashtags", i.e. words appearing in the tweets preceded by the character "#". The network contains a total of 125120 nodes and 2,248,406 edges, and a total of 3,140 communities.

- **Facebook** Like the Twitter network, the Facebook network is composed of ego-networks from the Stanford Large Network Dataset Collection [54]. Node attributes are extracted from user profiles, such as gender, job titles, institutions, etc. Ground truth communities have been manually labeled by the owner of the ego-network, and represent his "social circles". The size of the full network is 4089 nodes and 170174 edges, with 193 communities.

The statistics related to the benchmarks are reported in Table 3.1.

Baseline for comparison In order to investigate whether overlapping communities and node attributes can aid the community detection task, we perform a comparative analysis with the following algorithms:

- **Standard WeBA algorithm** We first test the performance of the original algorithm, without node attributes and with non-overlapping auxiliary community detection.

- **Overlapping WeBA** The second algorithm is the original version of WeBA, but with the addition of our algorithm for detecting overlapping auxiliary communities.
- **Overlapping WeBA with Node Attributes** Finally, we test the complete version of our method, considering both overlapping communities and the availability of node attributes.

Evaluation metrics We quantify the performance in terms of the agreement between the ground-truth communities and the communities detected by the algorithms. To compare a set of ground truth communities C^* to a set of detected communities C, we use the following measures: Precision (P), Recall (R) and F-Measure (F) (Eq. 3.14-3.16), which evaluate the number of correct pairs of vertices clustered into the same community kernel.

$$P(C_i, C_j^*) = \frac{\left|C_j^* \cap C_i\right|}{|C_i|} \tag{3.14}$$

$$R(C_i, C_j^*) = \frac{\left|C_j^* \cap C_i\right|}{\left|C_j^*\right|} \tag{3.15}$$

$$F(C_i, C_j^*) = \frac{2 \times P(C_i, C_j^*) \times R(C_i, C_j^*)}{P(C_i, C_j^*) + R(C_i, C_j^*)} \tag{3.16}$$

Moreover, we consider Jaccard Index (J) to measure the pairwise resemblance of C with C^* (Eq. 3.17).

$$J(C_i, C_j^*) = \frac{\left|C_j^* \cap C_i\right|}{\left|C_j^* \cup C_i\right|} \tag{3.17}$$

Finally, we introduce an index, based on the Jaccard measure, that evaluates the percentage of ground truth communities that have been successfully associated to the generated communities. This measure, called Equivalence (Q), takes value in the range $[0, 1]$ and is defined as follows:

$$Q(C, C^*) = \frac{1}{|C^*|} \left| \left\{ \underset{C_j^* \in C^*}{\operatorname{argmax}} J\left(C_i, C_j^*\right), \forall C_i \in C \right\} \right| \tag{3.18}$$

that is, the equivalence measures the percentage of ground truth communities "matched" by the generated communities.

3.3.4 Results

In this section we report the detailed results of our experimental investigation. In the first part of the section we describe a sensitivity analysis of the considered algorithms. In the second part, we report the best results obtained by each algorithm for the datasets shown in Sec. 3.3.3.

3.3.4.1 Sensitivity Analysis

The number of communities to be detected in the network depends on the parameter l, that regulated the number of kernel members of each community. In order to evaluate the performance of the algorithms varying the parameter l, a sensitivity analysis has been performed.

In Fig. 3.8 we report the results of our analysis, performed on the Philosophers dataset, computed in terms of Equivalence (as detailed in Eq. 3.18).

We can see that, in general, all three algorithms show their best performance when the kernel size l is small.

In particular, for Standard and Overlapping the performance decreases sharply for $l \geq 7$, indicating that the nodes forming a kernel are usually very few. When we consider node attributes, however, the performance remains high for a larger value of l. This behavior is mainly due to the attribute similarities considered as "textual relationships" between nodes. These "relationships" derived by the textual similarity usually outnumber structural relationships, therefore leading to larger kernels.

However, the performance starts dropping since $l = 6$, a value consistent with the result obtained by the other two algorithms.

An analogous sensitivity analysis has been performed on the other two benchmarks. It emerges that, also for bigger datasets, the number of kernel members are quite low.

The results of this sensitivity analysis (Fig. 3.8) suggests that the experiments should be performed considering a small kernel size, within the range of 3-6 nodes.

3.3.4.2 Comparative Results

We performed experiments on the three benchmarks described in Sec. 3.3.3 starting from the conclusions drawn from the sensitivity analysis step.

In Table 3.2 we report the results related to the Philosophers dataset. In order to make the results comparable, we run the three algorithms (WeBA, Overlapping WeBA and Overlapping WeBA

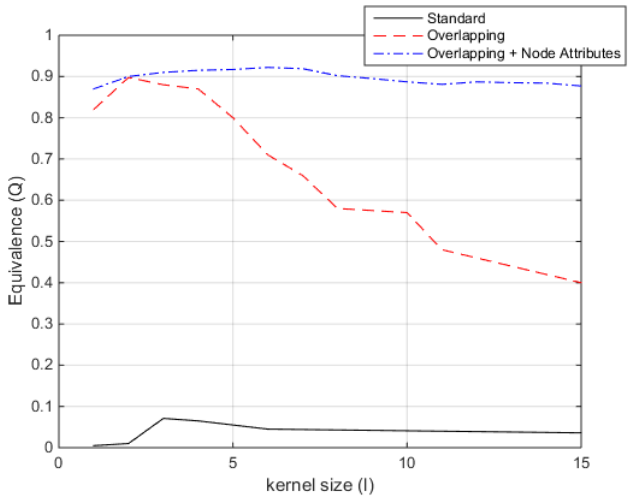


FIGURE 3.8: Sensitivity analysis on Philosophers dataset.

TABLE 3.2: Performance results on the Philosophers dataset. Best results for each row are marked in bold.

Measures	Standard WeBA	Overlapping WeBA	Overlapping WeBA with Node-Attributes
Recall (average)	39,05 ± 3,41	30,95 ± 0,77	**44,30** ± 1,39
Precision (average)	16,25 ± 2,27	**48,35** ± 1,44	35,77 ± 0,99
F1 Score (average)	21,66 ± 2,31	32,41 ± 0,47	**36,44** ± 1,77
Jaccard Index (average)	12,20 ± 1,47	20 ± 0,42	**23,08** ± 0,81

TABLE 3.3: Performance results on the Twitter dataset. Best results for each row are marked in bold.

Measures	Standard WeBA	Overlapping WeBA	Overlapping WeBA with Node-Attributes
Recall (average)	31,61 ± 1,36	**58,22** ± 2,75	47,51 ± 2,93
Precision (average)	19,96 ± 0,98	31,73 ± 1,14	**40,10** ± 1,12
F1 Score (average)	24,43 ± 3,01	32,47 ± 2,53	**36,04** ± 2,37
Jaccard Index (average)	10,98 ± 0,89	19,74 ± 0,77	**22,53** ± 1,01

with Node Attributes) with a kernel size $l = 3$, which has been previously proven as a good value for all three algorithms.

In this case, the equivalence measure highlights a performance of $7,12 \pm 1,97$ for the Standard algorithm, $87,21 \pm 2,71$ for Overlapping WeBA, and $90,91 \pm 3,95$ for Overlapping WeBA with Node Attributes. The first thing we can observe is a large increment in the equivalence measure when adding overlapping communities. This effect can be explained by the number of communities detected by the original WeBA algorithm, which is very low compared to the number of ground truth communities. A similar behaviour can be observed for all the other measures: the

TABLE 3.4: Performance results on the Facebook dataset. Best results for each row are marked in bold.

Measures	Standard WeBA	Overlapping WeBA	Overlapping WeBA with Node-Attributes
Recall (average)	27,85 ± 2,01	40,02 ± 1,92	**55,60** ± 2,47
Precision (average)	32,10 ± 3,30	37,16 ± 3,05	**48,75** ± 2,99
F1 Score (average)	29,80 ± 1,40	36,45 ± 1,73	**51,99** ± 1,54
Jaccard Index (average)	17,48 ± 0,78	22,64 ± 1,66	**35,19** ± 1,44

introduction of overlapping communities in the algorithm lead to a better performance in terms of F-Measure (from 22% to 32%) and Jaccard index (from 12% to 20%).

When considering node attributes, we can observe that the equivalence value is relatively unchanged (the value increase from 87% to 90%). This means that the addition of overlapping communities is generally sufficient for detecting the majority of the ground truth communities. However, we can see that the values of F-Measure and Jaccard score increase significantly (from 32% to 36%, and from 20% to 23% respectively). Thus, the communities obtained by the algorithm that exploits node attributes are closer to the ground truth communities than overlapping communities only.

In Table 3.3 and Table 3.4 we report the results obtained on the Twitter and Facebook datasets. Although Overlapping WeBA with Node Attributes always outperforms the other two approaches for all the considered performance measures and for any value of the kernel size l, we only report the results obtained for $l = 3$. This kernel size has been selected because it provides a good tradeoff between performance and computational cost for the three algorithms.

We can observe that the results for Twitter and Facebook are consistent with those obtained on the Philosophers dataset.

Allowing overlapping communities strongly improves the performance of the algorithm for both datasets (+8% and +7% for F-Measure, and +9% and +%7 for Jaccard index). This confirms that this improvement is essential when dealing with online social networks, whose users usually belong to multiple communities.

The performance improvement obtained by considering node attributes together with overlapping communities is higher on the Facebook dataset than the Twitter one. This behavior is mainly related to the nature of the node attributes that have been considered. While Twitter attributes are related to words used by the social network users in their posts, Facebook attributes are related to their personal information (school institution, name of the company where they work, etc.) which may be more informative when determining the community to which they belong.

The increment in the Twitter dataset, however, suggests that node attributes play a fundamental role even when they are obtained from a noisy source of information like user generated posts.

3.4 A Relational Clustering Algorithm for Interest-Based Community Detection

Social networks users naturally tend to aggregate to form communities. The definition of community is generally based on the relationships among users, where it refers to a highly connected group of users. However, the information provided by the contents shared by the users enables a different definition of community based not only on relationships among users but also on content similarity User content provides very specific information about the nature of the relationships of connected users: sharing similar contents is an indication of affiliation to the same group.

According to these considerations, in this section we assume a community as a group of users in a network that share *any kind of affinity or interest on a given topic*, whose evidence is based both on contents and relationships. Contents can contribute to infer to which community a user belongs to when the network structure is not informative (for instance a user can be uniformly connected to members of different communities, therefore making uncertain its membership). Conversely, the relationships established by the users can contribute to place them into the same community, even if one of them has limited and ambiguous content information. It's therefore a natural conclusion that it is important to take into account both sources of information together and consider network communities as sets of users that not only are densely connected, but also share common contents.

However, not all relationships are significant. Structural connections among users, such as friendships and leader-follower, represent a weak indication about their membership to the same *interest*-based community. For instance, two friends may be structurally connected, but they could belong to different communities whose interests are related to opposite political parties. In this scenario, interest-based relationships (i.e. interactions originated by retweet in Twitter and Like button in Facebook), are more informative because they can vary over time and they can be used to strength ties among people.

To this purpose, a two-fold contribution is given. First, a novel network modelling is introduced to capture both contents and interest-based relationships available in social networks. Then, we present INCA (short for **IN**terest-based **C**lustering **A**lgorithm), a partitional relational clustering approach for addressing the community detection problem. INCA is grounded on two assumptions that respect what happens in real social networks: (1) users belonging to the same

community are more likely to share similar content and (2) users affiliated to the same community are more likely to be connected through an interest-based relationship.

3.4.1 Approval Network

People generally have significant relationships with others who tend to be like themselves over a number of different aspects. **Homophily** is the principle underlying these kinds of relationships: *a contact among similar people occurs at a higher rate than among dissimilar people*. This principle implies that differences in terms of social characteristics translates into network distance, i.e. the number of relationships through which a piece of information must travel to connect two individuals [56]. Homophily can be distinguished in ***status homophily*** and ***value homophily***. Status homophily states that the similarity among people is based on informal, formal, or ascribed status, while value homophily is based on values, attitudes, and beliefs [56]. Status homophily includes the major sociodemographic dimensions like race, ethnicity, sex, or age, and acquired characteristics like religion, education, occupation, or behavior patterns. Value homophily includes the wide array of internal states presumed to shape our orientation toward future behavior.

Besides homophily, Carley [57] has developed a sociological theory called **Constructuralism**, whose main assumption is that *people who share knowledge are more likely to interact* (i.e. form ties). In particular, constructuralism argues that individual learning from interactions takes place on two levels. First, social interactions enable to collect over time new knowledge that represents similarity among users better than static sociodemographic dimensions like gender, age or co-working. Second, as humans receive and share knowledge with interaction partners, we can *learn* what we expect them to know. Paired with homophily, constructuralism explains how social relationships evolve via interactions as the knowledge that two actors share increases [58]. More knowledge is shared by two users, more is the evidence about their affinity and more strength is the relationship between them.

Most of the approaches for community detection consider only the status homophily, which is usually modeled through a "bond" relationship (e.g. friendships in Facebook, following/-follower in Twitter, circles in Google+), disregarding constructuralism. However, considering only bond connections could be a weak assumption when communities need to be automatically discovered:

- being friends, followers or belonging to a circle does not necessarily mean being part of the same community on a particular topic (e.g. two friends discussing about politics could belong to different political parties, being follower of a famous singer does not imply to belong to the same java programming community);

- interactions based on attitudes and believes (value homophily) should be preferred compared to static sociodemographic relationships (status homophily): an extensive experimental literature in social psychology established that attitude, belief, and value similarity lead to attraction and interaction enabling community creation [59];
- social interactions allow us to collect over time new knowledge that represents similarity among users better than static sociodemographic dimensions.

According to these considerations, value homophily and constructuralism have been considered in this work as principles underlying the community establishment and evolution. The two sociological processes have been captured into the subsequent definition of Approval Network, where strength and directionality of relationship derived by *approval interactions* are key elements to enable the discovery of interest-based communities of users.

Intuitively, an ***approval relationship*** is any endorsement given by a user towards posts emitted by other users (e.g. retweet in Twitter, "like" in Facebook and +1 in Google+ are considered as approval relationships). More formally, we can initially define a Directed Approval Graph as follows:

Definition 5. *A **Directed Approval Graph** is a quadruple* $DAG = \{V, E, \Theta, \Omega\}$*, where* $V = \{v_1, ..., v_n\}$ *represents the set of active users about the chosen topic;* $E = \{(v_i, v_j) | v_i, v_j \in V\}$ *is the set of approval edges, meaning that* v_i *approved* v_j*'s posts;* $\Omega = \{\omega_{i,j} | (v_i, v_j) \in E\}$ *is the set of weights assigned to approval edges, indicating that* v_i *approved* $\omega_{i,j}$ *posts of* v_j *on the chosen topic;* $\Theta = \{\theta_i | v_i \in V\}$ *is the set of coefficients related to nodes, where* θ_i *represents the total number of posts of* v_i *about the chosen topic.*

In order to reflect the relative importance of the approved messages, and therefore to smooth accordingly the weights $\omega_{i,j}$, a Normalized Directed Approval Graph can be derived as follows:

Definition 6. *Given an Approval Graph* $DAG = \{V, E, \Theta, \Omega\}$*, a **Normalized Directed Approval Graph** is derived as a triple* $N\text{-}DAG = \{V, E, \widetilde{\Omega}\}$*, where* $\widetilde{\Omega}$ *is the set of normalized weights of approval edges defined as:*

$$\widetilde{\Omega} = \left\{ \widetilde{\omega}_{ij} \;\middle|\; \widetilde{\omega}_{ij} = \frac{\omega_{ij}}{\max\limits_{z} \omega_{iz}} \times log_2 \left(1 + \frac{\omega_{ij}}{\theta_j} \right) \right\} \tag{3.19}$$

The measure presented in Eq. 3.19, which takes value in the interval $[0, 1]$, tries to capture the behavior of users on social networks as joint probability to belong to the same community. First, the common characteristic of an approval network is that most of the users usually approve only one message of a target user, and very few users approve two or more messages [60]. Thus, a logarithmic distribution should be used instead of a linear one. Second, the number of approvals

between two users does not necessarily indicate how much they agree with a particular topic. It could be influenced by the interest and originality of the target user's messages. For example, a user A could completely agree with user B but approves it one or two times only because the weak originality of B's messages. For this reason, the number of approvals from user A to B has been normalized by considering the maximum number of approvals from any user connected to B. Finally, this measure penalizes users who approve few messages of a particular target user if there are other users who approve many posts of the same target user.

Finally, given a N-DAG we can easily derive a heterogeneous graph that models not only user-user connections but also user-message relationships:

Definition 7. *Given a N-DAG* $= \{V, E, \widetilde{\Omega}\}$*, let* $P = \{p_1, \cdots, p_m\}$ *be the set of nodes representing posts on the chosen topic and* $A = \{(v_i, p_t) | v_i \in V, p_t \in P\}$ *be the set of arcs that connect the user* v_i *and the post* p_t*. A* ***Heterogeneous Normalized Directed Approval Graph*** *is a quintuple* $\Phi = \{V, E, \widetilde{\Omega}, P, A\}$*, also referred to as HN-DAG.*

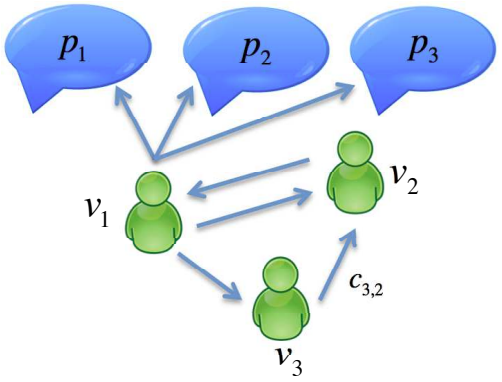


FIGURE 3.9: Example of HN-DAG representing user-post and user-user (approval) dependencies

3.4.2 INCA: Interest-based Clustering Algorithm

In this section we introduce INCA, a novel relational clustering algorithm based on a partitional paradigm, whose main goal is to find clusters (communities) representing the affinity of the users based both on contents that they share and relationships that they establish. Considering that the strength $\widetilde{\omega}_{ij}$ (defined in Eq. 3.19) and directionality of approval edges can be viewed as a reference for which the user v_i reveals an interest on a specific topic provided by the user v_j, the community detection algorithm should benefit from this kind of *embedded supervision*. Similarly to [61], this form of *supervision* can be exploited to guide the assignment process jointly with content information. More precisely, since not all connections among users are equally significative, the strength and directionality encoded by HN-DAG can contribute as an evidence of a probabilistic relationships. From a clustering point of view, since the coefficient

$\widetilde{\omega}_{ij}$ is associated to an approval that the user v_j gives to a message provided by the user v_i, it can be interpreted as an indication - and in particular as a joint probability - to belong to the same community, i.e.:

$$P(v_i \in C_t \wedge v_j \in C_t | v_j \in Ne(v_i)) \approx \widetilde{\omega}_{ij} \tag{3.20}$$

where $\mathrm{Ne}(v_i) = \left\{ v_j | (v_i, v_j) \in E \right\}$ denotes v_i neighborhood. Contents and relational information can now converge into a unified model able to smooth the community (cluster) assignment based on textual representation and probabilistic relationships both encoded by HN-DAG. More formally, given a HN-DAG Φ the main goal is to maximize the posterior probability of v_i to belong to a given community C_t:

$$\arg\max_g P(v_i \in C_g | \Phi) \tag{3.21}$$

In order to determine the optimal community assignment for each user v_i, we should start by estimating the most probable cluster placement on the base of contents. Considering that the community label is hidden, we can only estimate how much the user contents are similar to a representative element c_t of the community C_t. Following the Vector Space Model [62] derived by considering all the messages posted by the user v_i, we can approximate the probability of v_i to belong to the community C_t as the angle between v_i and the centroid c_t:

$$P(v_i \in c_t) \approx sim(v_i, c_t) = \frac{\langle \overrightarrow{v_i}, \overrightarrow{c_t} \rangle}{|\overrightarrow{v_i}||\overrightarrow{c_t}|} \tag{3.22}$$

Starting from this initial evidence, we can smooth the probability assignment of the user v_i by considering the strength and directionality of approval relationships related to its neighbors. In particular, given $Ne(v_i)$ we have to consider two possible cases: v_i and $Ne(v_i)$ are in the same cluster, and on the contrary v_i and $Ne(v_i)$ are in two different clusters. The two cases are approached as follows.

1. *Agreement between v_i and $v_j \in Ne(v_i)$*: if a user v_i is evaluated w.r.t. a given cluster C_t and its neighbor v_j has been previously assigned to the same cluster, then v_i will be *more attracted* to C_t. Given a user v_i and its set of neighbors $\mathrm{Ne}^A(v_i) = \left\{ v_j \in C_t | (v_i, v_j) \in E, v_i, v_j \in C_t \right\}$, with $\mathrm{Ne}^A(v_i) \subseteq \mathrm{Ne}(v_i)$, we can compute the probability to assign v_i to the cluster C_t as:

$$\begin{aligned} P(v_i \in C_t | \mathrm{Ne}^A(v_i)) &= P(v_i \in C_t) \prod_{j=1}^{|\mathrm{Ne}^A(v_i)|} \frac{\frac{1}{2}\left(1 + P(v_i \in C_t \wedge v_j \in C_t)\right)}{P(v_j \in C_t)} \\ &\approx sim(v_i, c_t) \prod_{j=1}^{|\mathrm{Ne}^A(v_i)|} \frac{\frac{1+\widetilde{w}_{ij}}{2}}{sim(v_j, c_t)} \end{aligned} \tag{3.23}$$

The coefficient $\beta_{ij} = \frac{1+\widetilde{w}_{ij}}{2}$, which takes values in the interval $(0.5, 1]$, has the main goal to attract those users connected by approval relationships. In particular, β_{ij} can be viewed as: where γ is a constant and denotes what we called *equilibrium point*, i.e. a coefficient that allows users connected by at least an approval edge to be attracted towards the same community according to the strength w_{ij}. If a user v_i approved few posts of its neighbor v_j, then β_{ij} will tend to its lower bound implying that v_i will be slightly attracted to v_j and therefore to the same community. On the contrary, if the user approved many posts of its neighbor, then the coefficient β_{ij} will tend to its upper bound leading to a very strong attraction of v_i to C_t. A value of $\gamma \leq 0.5$ would lead to repulse two users even if there exists an approval edge between them.

Since $P(v_i \in C_t | \text{Ne}^A(v_i))$ is computed through an approximation, in order to have a coherent probability distribution, we need to compute the probability assignment of v_i to the complementary cluster $\overline{C_t}$ as follows:

$$\begin{aligned} P(v_i \in \overline{C}_t | \text{Ne}^A(v_i)) &= P(v_i \in \overline{C}_t) \prod_{j=1}^{|\text{Ne}^A(v_i)|} \frac{\frac{1}{2}\left(1 + P(v_i \in \overline{C}_t \wedge v_j \in C_t)\right)}{P(v_j \in C_t)} \\ &\approx 1 - sim(v_i, c_t) \prod_{j=1}^{|\text{Ne}^A(v_i)|} \frac{1 - \frac{1+\widetilde{w}_{ij}}{2}}{sim(v_j, c_t)} \end{aligned} \tag{3.24}$$

According to Eq. 3.24, if we are trying to assign a user v_i to a cluster $\overline{C}_t$ and its neighbors v_i have been assigned to C_t, then v_i will be attracted to C_t thanks to the coefficient β_{ij}. Combining Eq.3.23 and Eq.3.24, the probability of v_i to belong to the same cluster of its neighbors $\text{Ne}^A(v_i)$ is normalized:

$$P_N(v_i \in C_t | \text{Ne}^A(v_i)) = \alpha \langle P(v_i \in C_t | \text{Ne}^A(v_i)), P(v_i \in \overline{C_t} | \text{Ne}^A(v_i)) \rangle \tag{3.25}$$

where α is a normalizing constant.

2. *Disagreement between v_i and $v_j \in Ne(v_i)$*: if a user v_i is evaluated w.r.t. a given cluster C_t and its neighbor v_j has been previously assigned to a different cluster C_z, then v_i will be *repulsed* from C_t. Given a user v_i and its set of neighbors $\text{Ne}^D(v_i) = \{v_j \in C_z | (v_i, v_j) \in E, v_i \in C_t, t \neq z\}$, with $\text{Ne}^D(v_i) \subseteq Ne(v_i)$, we can compute the probability assignment of v_i to the cluster C_t as follows:

$$P(v_i \in C_t|\mathrm{Ne}^D(v_i)) = P(v_i \in C_t) \prod_{j=1}^{|\mathrm{Ne}^D(v_i)|} \frac{\frac{1}{2}\left(1+P(v_i \in C_t \wedge v_j \in C_z)\right)}{P(v_j \in C_t)}$$
$$\approx sim(v_i, c_t) \prod_{j=1}^{|\mathrm{Ne}^D(v_i)|} \frac{1-\frac{1+\widetilde{w}_{ij}}{2}}{sim(v_j, c_t)} \tag{3.26}$$

In this case, β_{ij} allows users connected by at least an approval edge to avoid being attracted towards different communities. If we are trying to assign user v_i to the community C_t while its neighbor v_j has been placed in C_z, v_i will be slightly repulsed from C_t. Analogously, if the user approved many posts of its neighbor, then it will be strongly repulsed from C_t. Similarly to the agreement case, we obtain a coherent distribution by computing the probability assignment of v_i to the complementary cluster $\overline{C}_t$:

$$P(v_i \in \overline{C_t}|\mathrm{Ne}^D(v_i)) = P(v_i \in \overline{C_t}) \prod_{j=1}^{|\mathrm{Ne}^D(v_i)|} \frac{\frac{1}{2}\left(1+P(v_i \in \overline{C}_t \wedge v_j \in C_z)\right)}{P(v_j \in C_z)}$$
$$\approx 1 - sim(v_i, c_t) \prod_{j=1}^{|\mathrm{Ne}^D(v_i)|} \frac{1-\frac{1+\widetilde{w}_{ij}}{2}}{sim(v_j, c_t)} \tag{3.27}$$

Combining Eq.3.26 and Eq.3.27, the probability of v_i to belong to the cluster C_t with regard to its neighbors $\mathrm{Ne}^D(v_i)$ is normalized as follows:

$$P_N(v_i \in C_t|\mathrm{Ne}^D(v_i)) = \alpha\langle P(v_i \in C_t|\mathrm{Ne}^D(v_i)), P(v_i \in \overline{C_t}|\mathrm{Ne}^D(v_i))\rangle \tag{3.28}$$

where α is a normalizing constant.

The proposed algorithm combines the probability models given in Eq.3.25 and Eq. 3.28 in order to choose the optimal cluster assignment for each user:

$$\arg\max_{C_t} \left\{ P_N(v_i \in C_t|\mathrm{Ne}^A(v_i)) \times P_N(v_i \in C_t|\mathrm{Ne}^D(v_i)) \right\} \tag{3.29}$$

According to the working principle that have been previously detailed, some considerations can be derived. If we make explicit the objective function underlying the proposed community detection approach, we can derive its lower and the upper bound.

If we consider the expanded version of Eq. 3.25 and Eq. 3.28, we can define the objective function as follows:

$$\max \sum_{i=1}^{|V|}\sum_{k=1}^{|C|}\sum_{l=1}^{|C|} \frac{\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{\beta_{ij}}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)\prod_j^{|\mathrm{Ne}^D(v_i)|}\left(\frac{1-\beta_{ij}}{\operatorname{sim}(v_j,c_l)}\cdot z_{jl}\right)}{\begin{array}{c}\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{\beta_{ij}}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)\prod_j^{|\mathrm{Ne}^D(v_i)|}\left(\frac{1-\beta_{ij}}{\operatorname{sim}(v_j,c_l)}\cdot z_{jl}\right)+\\ (1-\operatorname{sim}(v_i,c_k))\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{1-\beta_{ij}}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)\prod_j^{|\mathrm{Ne}^D(v_i)|}\left(\frac{\beta_{ij}}{\operatorname{sim}(v_j,c_l)}\cdot z_{jl}\right)\end{array}}\cdot z_{ik} \tag{3.30}$$

where the decision variable z_{ik} is equal to 1 if user i belongs to the k^{th} community, otherwise is equal to 0.

The upper bound is reached when every node has only agreeing neighbors (i.e. neighbors belonging to the same community). Therefore, for estimating the upper bound, Eq. 3.30 can be rewritten as:

$$\begin{aligned}
&\max \sum_{i=1}^{|V|}\sum_{k=1}^{|C|} \frac{\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{\beta_{ij}}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)}{\begin{array}{c}\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{\beta_{ij}}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)+\\ (1-\operatorname{sim}(v_i,c_k))\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{1-\beta_{ij}}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)\end{array}}\cdot z_{ik} = \\
&= \max \sum_{i=1}^{|V|}\sum_{k=1}^{|C|} \frac{\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{1}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\beta_{ij}\cdot z_{jk}\right)}{\begin{array}{c}\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\frac{1}{\operatorname{sim}(v_j,c_k)}\cdot z_{jk}\right)\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\beta_{ij}\cdot z_{jk}\right)+\\ (1-\operatorname{sim}(v_i,c_k))\prod_j^{|\mathrm{Ne}^A(v_i)|}\left((1-\beta_{ij})\cdot z_{jk}\right)\end{array}}\cdot z_{ik} = \\
&= \max \sum_{i=1}^{|V|}\sum_{k=1}^{|C|} \frac{\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\beta_{ij}\cdot z_{jk}\right)}{\begin{array}{c}\operatorname{sim}(v_i,c_k)\prod_j^{|\mathrm{Ne}^A(v_i)|}\left(\beta_{ij}\cdot z_{jk}\right)+\\ (1-\operatorname{sim}(v_i,c_k))\prod_j^{|\mathrm{Ne}^A(v_i)|}\left((1-\beta_{ij})\cdot z_{jk}\right)\end{array}}\cdot z_{ik}
\end{aligned}$$

Considering that the similarity value can vary in the range $[0,1]$, while β_{ij} takes value in the interval $(0.5,1]$, the upper bound is achieved when $\operatorname{sim}(v_i,c_k)\cdot z_{ik}\ \forall i \in V$, and $\beta_{ij}\cdot z_{jk}\ \forall j \in \mathrm{Ne}^A(v_i)$, are both equal to one.

According to these considerations, the bounds of Eq. 3.30 are $[0,|V|]$.

3.4.3 Experimental Settings

3.4.3.1 Datasets and algorithms

In this section we present the case studies that will be used to assess the validity of the proposed approach. We focused our experimental investigation on real datasets collected from Twitter, a popular microblog platform. Differently from the case presented in Sec. 3.3, there are no datasets available in literature with all the data required to build an approval network with the characteristics described in Sec. 3.4.1. For this reason, we decided to create our own datasets by crawling Twitter data, as detailed as follows. We have created three datasets composed of a set of users (V), the posts they have written (P) about a specific topic, and the underlying approval relationships relative to the topic. In Twitter the posts are generally referred as "tweets", and the user can approve another user's tweet by an action called "retweet".

In order to create the datasets suitable for the subsequent experimental investigation, the following steps have been performed:

1. We selected the most discussed topics on 11 June 2013 ('*trending topics*'):
   - ***iOS7***, the seventh major release of the iOS mobile operating system designed by Apple Inc.;
   - ***Prism***, the surveillance program under which the United States National Security Agency (NSA) have collected internet communications of foreign nationals from major US internet companies (disclosed by the press on 06 June 2013);
   - ***Superman***, (Man of Steel) is a 2013 superhero film based on the DC Comics character Superman.
2. For each topic, we crawled the tweets containing the corresponding word over two days (11-12 June 2013), collecting data about tweets and users who posted them;
3. Starting from the obtained posts, the retweet connections have been collected to finally derive the complete approval networks.

The characteristics of the three datasets[1] are reported in Table 3.5.

In order to compare the proposed approach with the state of the art, three main algorithms have been considered as baseline:

- ***K-Means*** [63], a text-based approach that works only on the user posts (this approach has been exploited in [64, 65] for community detection problems).

[1]The datasets can be downloaded at `http://www.mind.disco.unimib.it/public/site_files/INCA-dataset.zip`.

TABLE 3.5: Characteristics of the Twitter datasets.

Dataset	# Users	# Posts	# Approvals
iOS7	30206	82143	24779
Prism	15211	37610	16930
Superman	6125	15601	5614

- ***Spectral Clustering***, a graph-based approach that works only on the network structure (this approach has been exploited in [66, 67])
- ***Hybrid Spectral***, a hybrid approach (based on spectral clustering) that combines both textual and structural information in a unified model (this approach has been exploited in [50, 68, 69] for community detection issues).

All the considered clustering algorithms require to specify the number of communities. To this purpose, we performed several experiments with different numbers of clusters k, ranging from 2 to 256, with the following progression $k = \{2,4,8,16,32,64,128,256\}$. Since all the considered approaches are intrinsically stochastic, i.e. each execution of the algorithm can lead to a different result, we performed 500 experiments for each value of k in order to guarantee statistically significant results.

3.4.3.2 Performance Measures

Considering that the main goal is to create communities based on text and relationships, the performance evaluation should take into account both kinds of sources. No ground truth, however, is available in the collected datasets. For this reason, we measured the effectiveness of the investigated algorithms by considering three (internal) evaluation metrics:

- **Average Cosine Similarity (ACS)** This index evaluates the compactness of clusters using the textual similarity among posts provided by the users assigned in the same community. Given a set of clusters C, the Average Cosine Similarity index is defined as follows:

$$ACS(C) = \frac{1}{|C|} \sum_{k=1}^{|C|} \sum_{v_i \in C_k} \sum_{v_j \in C_k} cos(v_i, v_j) \tag{3.31}$$

  where $cos(v_i, v_j)$ is the cosine similarity between the posts provided by v_i and v_j. ACS takes value in the interval $[0,1]$, where a higher ACS indicates a better clustering.

- **Modularity** This index measures the capability of the algorithm of forming well-connected communities. Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. Given the set of clusters C, the modularity score (Q) is defined as follows:

$$Q(C) = \frac{1}{2|E|} \sum_{i,j} \left(x_{ij} - \frac{d_i d_j}{2|E|} \right) \delta(v_i, v_j) \tag{3.32}$$

  where $|E|$ is the number of approval edges, d_i is the degree of node i, x_{ij} is equal to 1 if there is an edge from v_i to v_j and 0 otherwise, $\delta(i, j)$ is equal to 1 if there v_i and v_j are in the same community and 0 otherwise.

  Networks with high modularity have dense connections between the nodes within the same community, but sparse connections between nodes that belong to different clusters.

- **Harmonic Mean** To evaluate the overall performance, we compute the harmonic mean of Average Cosine Similarity and Modularity index in order to measure how homogenous are the clusters with respect to both textual and relational aspects. The harmonic means (H) is defined as follows:

$$H(C) = \frac{2 \times ACS(C) \times Q(C)}{ACS(C) + Q(C)} \tag{3.33}$$

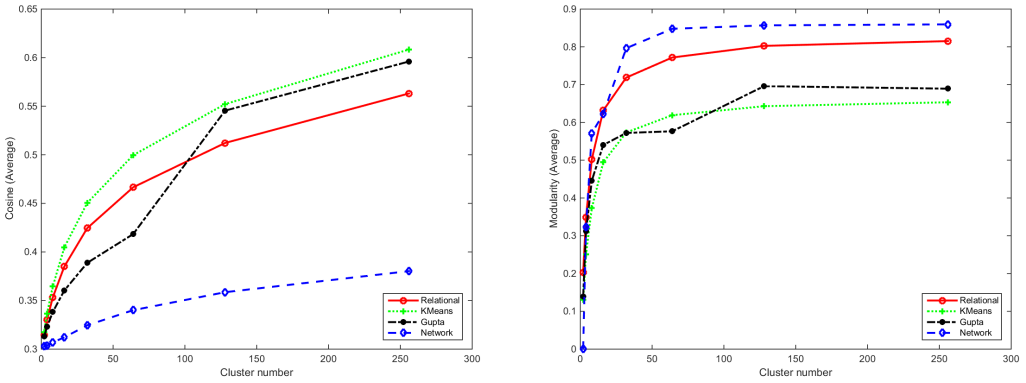
3.4.4 Results

3.4.4.1 Analysis of the full network

We start our analysis by showing the results obtained by the proposed approach compared with the baseline algorithms on the three datasets. In Fig. 3.10 we report the separate performance in terms of ACS and Modularity on the iOS7 network[2]. As would be expected, K-Means reaches the best performance in terms of average cosine similarity, but has the worst performance in terms of Modularity. On the other hand, Spectral Clustering has the best performance in terms of modularity, while its ACS performance is very low.

In particular, for $k \leq 8$, Spectral Clustering is not able to derive a good partitioning of the network because of the high number of connected components available in the original graph. In this case, the algorithm tries to create less communities (i.e. 8 clusters) than the existing connected components (i.e. 6441 connected components). In Fig. 3.11 we show a snapshot of the iOS7 network, where we can easily note that the network is composed of one larger connected component (around 11000 nodes) and a huge number of smaller ones, often composed of two nodes only.

[2]The results for Prism and Superman are reported in Appendix A.



(a) (b)

FIGURE 3.10: Average Results for Cosine Similarity (a) and Modularity (b) on the iOS7 dataset. Each colored line indicates an algorithm, as follows: INCA (red), Hybrid (black), K-Means (green), Spectral (blue).

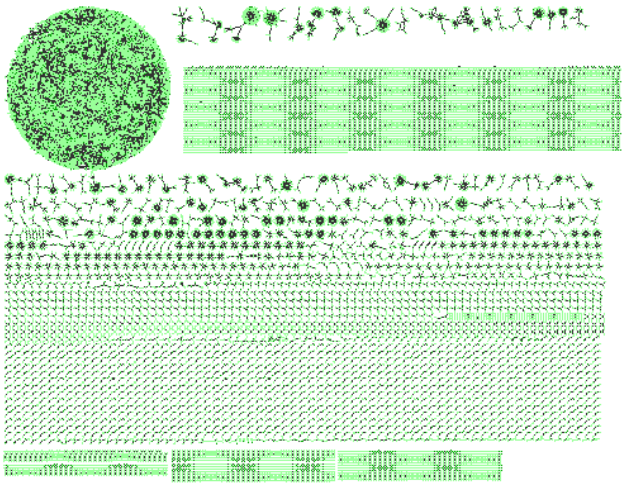


FIGURE 3.11: Full iOS7 network

Concerning Hybrid Spectral and the proposed approach, a good performance on ACS and Q would be expected thanks to their explicit modeling of both information sources. However, we can see that Hybrid Spectral is able to obtain good performance in terms of textual similarity only for a higher number of clusters, while its Modularity score becomes asymptotic to the one obtained by K-Means for almost all the number of clusters required. The poor performance of Hybrid Spectral with respect to the Modularity index is mainly due to the propositionalization of relationships. In fact, Hybrid Spectral makes use of the spectrum (eigenvalues) of a similarity matrix derived as linear combination of contents and (undirected) relationships. In this way, the relationships available in the network are *flattened*, leading to a biased representation. The proposed approach, on the contrary, deals with both information sources in their native form and therefore is able to guarantee (sub)optimal performance for any value of k. This means that the

proposed approach is able to model the multi-objective problem, where both contents similarity and structural cohesion are optimized.

TABLE 3.6: Average Harmonic Mean over 500 runs on the iOS7 dataset. **Bold** numbers denotes the best performing approach, while underlined numbers represents the second best. Symbol * indicates that INCA outperforms a given baseline by 99% statistical confidence. Symbol $^+$ indicates the outperformance by 95% statistical confidence.

	K-Means	Spectral	Hybrid	INCA
k=2	0.180*	0.002*	0.193*	**0.232**
k=4	0.285*	0.297$^+$	0.316$^+$	**0.335**
k=8	0.367*	0.399*	0.384*	**0.413**
k=16	0.445*	0.416*	0.432*	**0.478**
k=32	0.504*	0.461*	0.463*	**0.534**
k=64	0.553*	0.485*	0.485*	**0.581**
k=128	0.594*	0.506*	0.612*	**0.625**
k=256	0.630*	0.527*	0.639*	**0.666**

If we focus on Table 3.6, where the Harmonic Mean (H) is reported as the average results over 500 runs, several remarks can be pointed out[3]:

- The proposed approach is robust despite the stochasticity of the process (as well as the other state of the art approaches). By computing the confidence interval[4] on H over the considered runs, our method takes endpoint values between ± 0.0069 and ± 0.0003, ensuring small variations among different runs.

- The overall performance of INCA is higher than the other algorithms for any value of k. Comparing the proposed approach to K-Means, we notice that it is able achieve better performance, as it combines the information from the contents as well as the network. Similarly, it also outperforms the network-based approach, which only focuses on structural relationships. In particular, the proposed approach never performs worse than state-of-the-art methods that use only a single source of data. The strong performance of INCA is not obvious, as it would be entirely possible that combining two sources of data would confuse the algorithm and degrade the overall performance. In fact notice that Hybrid Spectral, which uses both kinks of information, performs worse than than K-Means and Spectral for some values of K. Thus, we believe that the strong performance of INCA is an indication that it is able to combine the best ingredients from both worlds.

[3] see Appendix A for the Average Harmonic Mean on the Prism and Superman datasets.

[4] Confidence intervals provide a range about the observed "effect size", allowing us to understand how likely the generated solutions are: the smaller the confidence interval, the more certain we are about the solution. In our case, intervals have been estimated with a confidence level of 95%.

- The relative improvement that INCA is able to provide across datasets over the baselines is statistically significative. The proposed approach yields the best performance in 22 out of 24 cases with a confidence level of 99%, and in all the cases with a confidence of 95%. In particular, INCA is able to ensure a relative improvement that ranges between 5% and 28% on K-Means, 3-26% on Spectral and 2-20% on Hybrid Spectral.

3.4.4.2 Focus on a reduced network

Previously we focused on the quantitative evaluation of the performance of the algorithms. In order to analyze in a more detailed way the results obtained by the investigated approaches, we provide a qualitative evaluation that can describe the obtained communities both by a structural and a textual point of view. We perform this qualitative evaluation on a smaller network, which allows us to give a closer look to the communities generated by the algorithms.

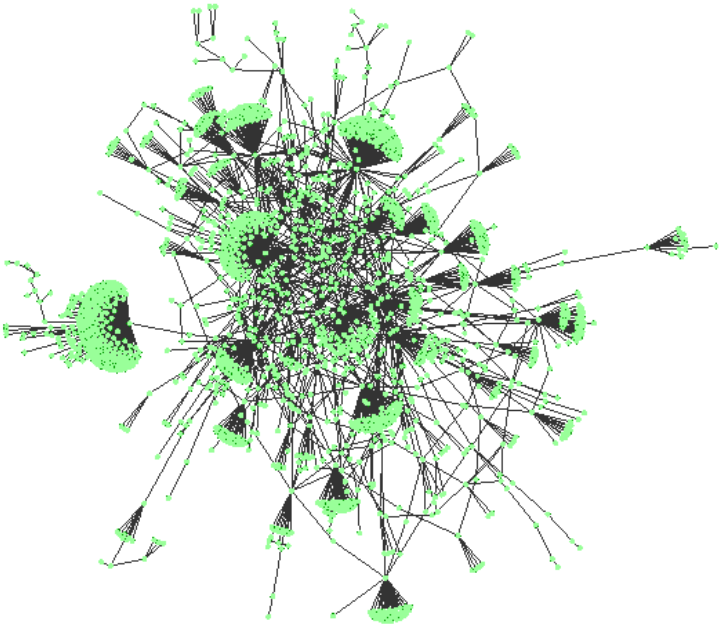
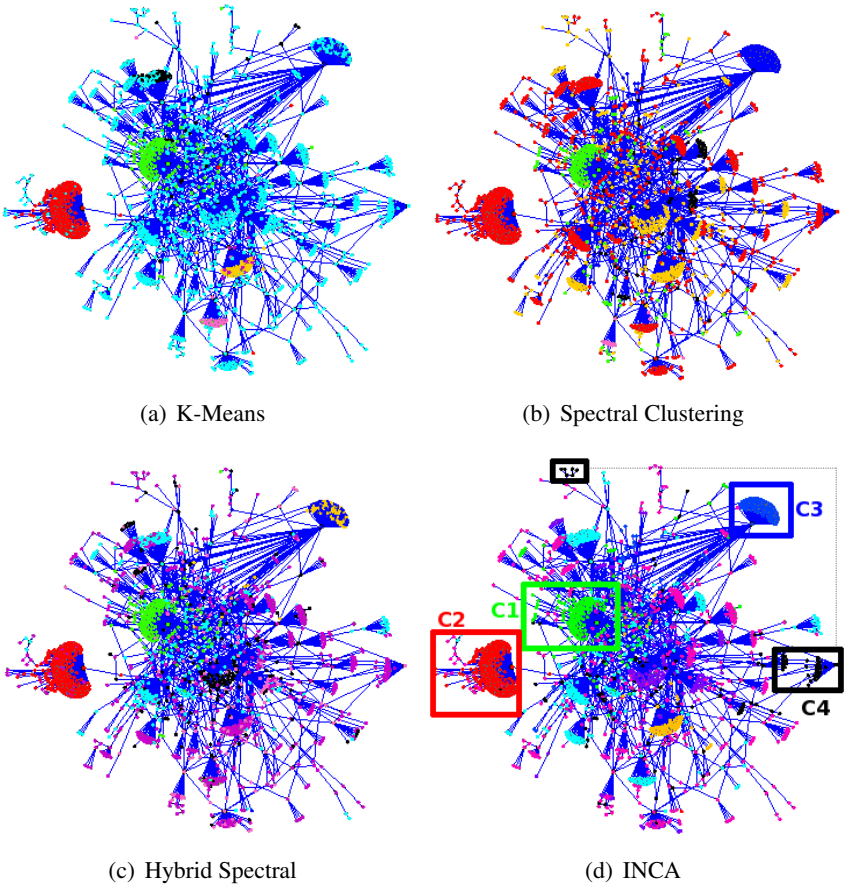


FIGURE 3.12: Reduced iOS7 Network

For this reason, we derived a reduced version of the iOS7 network (see Fig. 3.12), by considering 15% of the nodes from its larger connected component. The reduced network is composed of 1813 users, 2162 approval relationships and 3931 tweets. In this scenario, we compared the community detection algorithms by creating up to 32 communities (this to avoid an extensive fragmentation of users and therefore extremely small communities).

In order to perform the qualitative analysis, we study the results obtained for k=8, a reasonable number of clusters that allows to retain a good performance in terms of Harmonic Mean



(a) K-Means (b) Spectral Clustering

(c) Hybrid Spectral (d) INCA

FIGURE 3.13: Communities identified on the Reduced iOS7 network (K=8).

(H), while keeping an understandable representation of the communities for analysing the corresponding structures and contents. In Figure 3.13 we show a representation of partitioned networks generated by the four algorithms, reporting their best solution for $k = 8$.

We can see that, as expected, few communities are commonly discovered by more than one algorithm. In the following, we will consider a sample of communities discovered by INCA, and proceed through a comparison with the other approaches to highlight peculiarities and limitations:

- **Community C1**: *Apple Worldwide Developer Conference (WWDC)*
  This community, whose the underlying topic is about the Apple Worldwide Developer Conference, is the only one detected by all the considered algorithms. The community is mainly characterized by a central user, i.e. the conference organizer, who spreads contents about the upcoming event. The approval relationships represent the retweets about the conference news, and posts are the related to the news themselves. This redundancy of information leads all the approaches, both based on relationships and/or contents, to be able to detect this community. In Figure 3.14, the word cloud of this commonly identified community is reported.

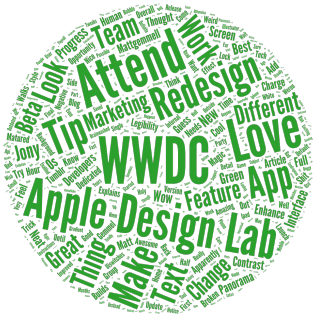


FIGURE 3.14: Tag cloud for the community "Apple Worldwide Developer Conference (WWDC)"

- **Community C2**: *iOS7 Blocking Message Feature*

  This community, whose main topic is about the iOS7 Blocking Message Feature, is found by Hybrid Spectral, k-Means and INCA. The Spectral Clustering approach highlights a different result: although several groups of nodes share dissimilar contents (e.g. Beta Version, Design Lab and Icons), the presence of a retweeted central user leads to an erratic aggregation where heterogeneous users are placed in the same cluster. This behaviors can be observed in the word cloud reported in Figure 3.15.

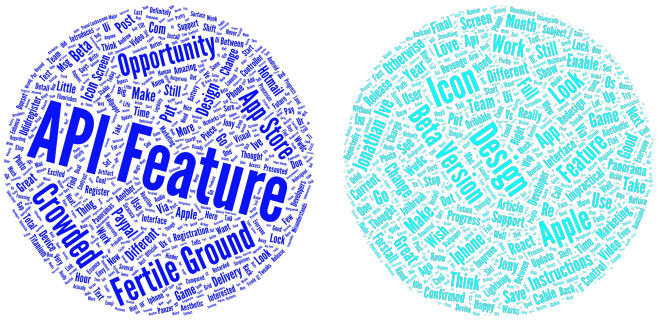
(a) K-Means, Hybrid and INCA (b) Spectral Clustering

FIGURE 3.15: Word cloud representing the community "iOS7 Blocking Message Feature"

- **Community C3**: *iOS API Features*

  The set of nodes denoted as C3 represents a group of users discussing about the API features provided by the new release of iOS7. While K-Means and Hybrid Spectral split this set of users into two communities, Spectral and INCA use these nodes to create a single community. If we analyze more in detail the structure and the contents of this set of nodes, we can see that there are two groups of users retweeting two tweets (generated by a central user) that are syntactically and lexically different, but semantically strongly related. In this case, while the approval relationships are able to guide the process of Spectral and INCA to group these two sets of users into the same community, the approaches strongly based on the text similarity as K-Means are biased by the syntactic difference of the retweetted messages. In particular, K-Means recognizes two separate communities with a very unbalanced partition. In fact, the light blue "macro-community" covers more than 66% of

the nodes. In this scenario, only approaches strongly grounded on the network structure are able to overcome syntactical/lexical differences of messages by exploiting the relational information. The word cloud for the resulting community generated by INCA and K-Means are compared in Figure 3.16.



(a) INCA (b) K-Means

FIGURE 3.16: Tag cloud for the community "iOS7 API Features"

- **Community C4**: *iOS7 Beta Version*
  Concerning the set of users marked as C4, we can see that INCA detected a black-colored community that includes other users placed in the topmost area. The content of the tweets posted by the two groups of users is related to the "beta version" of iOS7, also containing words suggesting a positive attitude towards Apple's operating system (see Figure 3.17(a)). All the other approaches are not able to detect this community. K-Means merges the set of users into the large light blue "macro-community", losing the focus on which topic people are discussing about (See Figure 3.16(b)). Similarly to K-Means, Hybrid Spectral erroneously associates these users to a large community (orange nodes), where people mention not only the iOS7 Beta version, but also hardware features of the new iPhone (specifically, about lightning and cables - see Figure 3.17(b)). Spectral Clustering, which is only driven by the network structure information, erroneously splits this set of users in two communities (i.e. red and orange nodes) although they discuss about the same topic (see Figure 3.17(c) and (d)).

For sake of completeness, we also report a quantitative analysis on the Reduced Network. In Table 3.7 the average Harmonic Mean computed over 500 runs is detailed. It is easy to note that the proposed approach is able to outperform the other methods in all the cases with a confidence level of 99%. If we compare the results obtained on the reduced network (Table 3.7) with the ones derived from the full dataset (Table 3.6), we can highlight that the gap (absolute improvement) between INCA and the other algorithms becomes wider. The main reason is the different structure of the graph: the reduced network has more edges than nodes, while the complete network has more nodes than edges.

(a) INCA (b) Hybrid Spectral

(c) Spectral (Red) (d) Spectral (Orange)

FIGURE 3.17: Tag cloud for the community "iOS7 Beta Version"

TABLE 3.7: Average Harmonic Mean over 500 runs on the reduced iOS7 dataset. **Bold** numbers denotes the best performing approach, while underlined numbers represents the second best. INCA outperforms all the given baselines by 99% statistical confidence.

	K-Means	**Spectral**	**Hybrid**	**INCA**
k=2	0.192	0.151	0.220	**0.243**
k=4	0.324	0.242	0.281	**0.373**
k=8	0.394	0.347	0.341	**0.454**
k=16	0.422	0.414	0.453	**0.505**
k=32	0.427	0.486	0.465	**0.534**

In this scenario, the relational information exploited by the proposed approach is able to contribute more to create homogeneous communities of users. Also on the reduced network, the proposed approach is robust despite the stochasticity of the process. By computing the confidence interval over the considered runs, our method is characterized by a confidence interval of H that takes even smaller endpoint values between ± 0.005 and ± 0.0001, ensuring small variations among different runs. Also in this case, the overall performance of INCA is higher than the other algorithms for any value of k, ensuring the relative improvement reported in Table 3.8.

TABLE 3.8: INCA Relative Improvement (%)

	INCA vs K-Means	**INCA vs Spectral**	**INCA vs Hybrid**
k=2	26.65	60.41	10.21
k=4	15.10	54.17	32.64
k=8	15.24	30.72	32.95
k=16	19.51	21.85	11.46
k=32	25.08	9.86	14.83

3.4.4.3 Analysis of the computational complexity

In this section we analyze the computational complexity of INCA compared to other community detection algorithms available in literature. In order to make easier the following discussion to the reader, we redefine these cardinalities as follows: the number of nodes $|V|$ will be denoted by n, the number of edges $|E|$ as m, the number of communities $|C|$ as k. In the following, we will also consider d as the total number of words appearing in users posts.

Community detection has been shown to be a NP-Hard problem [32, 70].

We show that, for real-world social networks, INCA takes linear time in the number of nodes and edges of the network.

At each iteration, INCA computes the best assignment for each node of the graph. The assignment phase can be split into three steps: (1) calculation of textual similarity, (2) smoothing due to agreeing neighbors, and (3) smoothing due to disagreeing neighbors.

The first step computes the similarity between each node and all the centroids by using the cosine similarity. This process requires $O(nkd)$.

The second step computes the smoothing in case of agreement between the node and its neighbors. For each neighbor, this process needs to estimate the similarity between the neighbor and all the centroids with a cost of $O(d)$. In the worst case a node has n neighbors, thus the total cost of the agreement smoothing will be $O(n^2kd)$. Considering that the similarity between a neighbor v_j and a centroid c_t represents an invariant when computing $P(v_i \in C_t|Ne^A(v_i))$ and $P(v_i \in \overline{C_t}|Ne^A(v_i))$ and therefore can be omitted, the cost of each agreement smoothing reduces to $O(n^2k)$.

The third step is equivalent to the second, thus its total cost is $O(n^2k)$.

In the worst case, the total cost for each iteration is $O(nkd+n^2k+n^2k) = O(nkd+n^2k)$. However, social networks are usually sparse and the average number of neighbors of a node is generally low. Therefore, the cost of steps 2 and 3 reasonably takes a time which depends on the

total number of edges m, leading to a cost of $O(mk)$ instead of $O(n^2k)$. Hence, the total cost of a single iteration of INCA will be $O(nkd+mk+mk) = O(nkd+mk)$, which is linear in the number of edges in the network.

In Table 3.9, we report the computational complexity of INCA together with some other hybrid algorithms available in literature. The proposed approach results to be the most efficient method (together with that of Yang et al. [48]). This computational complexity could be further reduced by adopting a parallel computing paradigm. In particular, at each iteration each node could be processed separately to determine the optimal community, increasing the efficiency up to a factor of n.

TABLE 3.9: Computational complexity of hybrid community detection algorithms. n: number of nodes, m: number of edges, d: number of words, k: number of communities, T: number of topics (only for [47]).

Algorithm	Computational Complexity
INCA	$O(nkd+mk)$
Hu et al. [46]	$O(n^2k^2+k^2d+kd)$
Natarajan et al. [47]	$O(mk+kTd)$
Yang et al. [48]	$O(nkd+mk)$
Gupta et al. [50]	$O(n^3+nk^2)$
Qi et al. [71]	$O(md+nkd)$
Ruan et al. [72]	$O(n^2logn)$
Yang et al. [73]	$O((kn)^2)$

3.5 Conclusion

In this section we addressed the task of community detection in social networks, focusing on the limitations that the existing approaches show.

The main limit is the inability of most methods to exploit the information provided by user relationships and user-generated content at the same time. We therefore presented two approaches to overcome this problem, starting from two different point of views.

The first contribution extends a graph-based algorithm to include the additional information provided by the textual content generated by the users. This algorithm aims at detecting a traditional type of community where users belonging to it are generally tightly connected. In this case a user may belong to multiple groups at the same time, so we improved the base approach to allow the inference of overlapping communities. The comparison with the baseline algorithm shows that the ability to find overlapping communities is important for detecting the correct groups of users in social networks, but that the inclusion of node attributes can provide important additional information, leading to results which better fit the real communities.

The second contribution approaches the problem from a different perspective. The task of community detection is usually addressed in literature by considering only the socio-demographic connections among users. However, these kind of relationships are a weak indicator that people belong to the same interest-based community. Users continuously generate textual messages that can provide useful additional information for detecting communities, and that *approval* relationships can indicate a stronger connection compared to friendship. To this purpose, we introduced INCA, a scalable relational clustering algorithm that makes use of both textual content and approval relationships to detect interest-based communities in social networks. We have shown that INCA outperforms state-of-the-art baselines which are only based on text data, network data, or their combination. The strong performance of INCA is not obvious, as it would be entirely possible that combining two sources of information would degrade the overall result. We performed both a quantitative analysis and a qualitative analysis of the generated communities, and the results suggest that using both sources of data can improve the quality of the communities.

Overall, it has been shown that the combined use of both textual content and relationships is beneficial to improve the performance of community detection algorithms.

Chapter 4

Semi-Supervised Relational Machine Learning for Opinion Detection

4.1 Introduction

Opinion detection, also called *opinion mining* or *sentiment analysis*, is an important task in social network analysis. According to Pang and Lee [74], "*an important part of our information-gathering behaviour has always been to find out what other people think*".

Opinions are fundamental in all human activities that involve a decision, as they influence our behaviour. Businesses and organization are always interested in finding consumer or public opinions regarding their products and services. Even individual consumers want to know the opinion of other people prior to make a purchase, or to understand the idea of other people about political candidates before an election.

In the past, such opinions were gained through friends or family for individuals, or through opinion polls or surveys for businesses. The advent of social networks changed this situation, allowing people to express their opinion directly, and often without being asked, simply by writing something on their favourite social media page. This availability of opinionated content in the web opened new perspectives on different target applications.

One of the principal applications of opinion mining is the analysis of reviews, i.e. comments that people write to evaluate a service or a product. Several search engines have been developed to analyze reviews and comments written by social network users (following methods already used for studying reviews on websites like Epinions or TripAdvisor) [75].

Several opinion mining studies have tried to exploit the sentiment in textual sources for improving predictions in many fields. Liu et al. [76] addressed the problem of sales performance prediction, inferring the subjectivity of the reviews to determine their importance.

Hong and Skiena [77] studied how to improve sport bets (in particular for NFL) using the polarity of the comments relative to specific games in blogs and microblogs. Several works addressed the predictions of movie revenues. Asur and Huberman [78] showed how the sentiment of Twitter posts can be a powerful predictor of the movie popularity. Joshi et al. [79] enhanced the prediction of the opening weekend revenue by adding sentiment-based features. Another important field of application is finance or, more specifically, the prediction of stock markets trends. Bollen et al. [80] and Bar-Haim et al. [81] investigated the importance of sentiment polarity in tweets to determine the public mood which can affect stock markets. Zhang and Skiena [82] even proposed market strategies starting from sentiment collected from blogs and microblogs. Finally, also the analysis of political debates (in particular during important events like elections) has found a valid support in opinion mining. Tumasjan et al. [83] show that Twitter is a valid real-time indicator of political sentiment. Chen et al. [84] studied the political standpoint of senators belonging to opposite parties over different topics. Yano et al. [85] modeled the relationship between the content of political blog posts and the total number of response they received, to predict the impact the post will have on the readers. Sentiment analysis systems have also an important role as subcomponents of other tools. They are increasingly used for improving recommendation systems [86, 87], where for example they can be used to improve the advice over products or services that receive several positive reviews, or even avoid to recommend items with many negative reviews. Opinion mining tools are also used for the detection of ”flames”, i.e. the use of hostile and insulting language in social networks or other means of communications like e-mails [88].

While opinion mining has been originally developed for the analysis of general text, the availability of user-generated content in social media has raised several issues. First, the opinion is traditionally tied to the single post, i.e. the focus is on the sentiment emerging from the document; however, in social networks the focus has been moved towards the actor expressing such opinion. Second, the opinion is generally inferred using only the textual content of the document. In social networks, the relationships among users can reveal the influence that neighbors have on the opinions of another user.

In the next section, we present an overview of the literature behind opinion mining in social networks, highlighting the major issues and the new challenges this environment presents. In Sec. 4.3 and 4.4 we will then present two approaches that deal with these challenges and try to overcome the limitations shown by the existing approaches. The first contribution will focus on the opinion of the users, exploiting both the textual content of their posts and the connections with their neighbors in order to infer their opinion on a given topic. The second contribution, on

the other hand, will expand on the first one by inferring at the same time the sentiment expressed by the users and the sentiment expressed by their own posts.

4.2 Related Work

Traditional opinion mining is usually applied on large documents such as web pages, where the amount of text available allows to study the sentiment over different levels of granularity. Usually, the levels considered are document level (analyzing the opinion of a whole document), sentence level (breaking the document into sentences and determining the sentiment of each) and entity/aspect level (determining which entities are described in the document and inferring the sentiment for each one of them).

When dealing with text from social networks, however, it is hard to separate these three levels. Usually, users tend to write short messages, either willingly (to save time) or because they are effectively limited by the social media itself: Twitter, for example, imposes a limit of 140 characters for each post. Thus, the document usually contain a single sentence, and the text often refers to a single entity or aspect.

However, when considering opinions in social network, we can distinguish two other levels of granularity:

- **Post level** This level is the equivalent of document level for larger documents. Given a specific topic, the system determines if a post generated by a user in a social network expresses an overall positive or negative opinion about the topic.
- **User level** Sometimes it is much more interesting to infer the overall opinion of single users of a social network rather than the opinion expressed in a single post, which does not always reflect the general opinion of the user.

Most methods are only focused on post-level sentiment analysis, as they are derived from approaches developed for traditional opinion mining on documents, which rarely display relationships or connections. Only recently some authors have introduced algorithms explicitly developed for sentiment analysis on social networks, which consider the sentiment of the users or even the relationships among them.

In the following, we give an overview of the text-based methods developed for sentiment analysis and of the problems related to it. Subsequently, we will describe the approaches that, in addition to considering text, take into account the relations between users when determining their opinion.

4.2.1 Text-based Sentiment Analysis

Sentiment analysis is usually formulated as a text classification problem, where from the textual content (posts, reviews, etc.) the algorithm must determine the most probable of two (or more) classes. These classes reflect the text polarity (*negative* or *positive*), the subjectivity (*subjective* or *objective*), or sometimes other finer aspects of the opinion (e.g. happiness, anger, sarcasm, etc.).

Being a text classification problem, sentiment analysis has been addressed mostly using traditional supervised learning methods, e.g. Naive Bayes classifiers and Support Vector Machines [89, 90].

Typically, the textual content is processed using a *bag of words* approach (sometimes referred to as *unigrams*), i.e. by considering the frequency of occurrence of each word, but not their order of appearance. Naive Bayes and SVM have been shown to work well with unigrams, although more recent approaches have tried to consider complex features.

Some of these features are the following:

- *Occurrence of terms.* The most common features used in sentiment classification are words, both counting their individual occurrence (unigrams) and the occurrence of contiguous sequences (bigrams, trigrams and so forth). Tipically, the occurrence is weighted using the absolute frequency: this is called *Term Frequency (TF) weighting schema*. Other schemas have been considered: the *boolean* schema, which assumes binary values (either zero or one) depending on whether the word has occurred at least once in the document; and the *TF-IDF weighting schema*, which weighs the importance of each word by its rate of appearance in all the considered documents.

- *Part of speech.* The part-of-speech of each word can be an important feature for the classification of sentiment. In particular, adjectives are often employed as features [91, 92], and the presence of certain adjectives is a good indicator of sentiment (e.g. ”good”, ”bad”, ”worse”, etc.). However, considering only adjectives can limit the potential of a sentiment analysis algorithm. Pang et al. [93] have shown that using only the adjectives leads to a worse performance than using the same number of most frequent words. In fact, other part-of-speech tags, like nouns or verbs, can have a fundamental role in determining the sentiment of a sentence.

- *Sentiment shifters* Some expressions are used in common language to change the sentiment orientation of other words. In presence of these expressions, the sentiment might have to be changed from positive to negative or viceversa. An example of these expressions are negations: for example, in the sentence *”People don’t hate Obama”* is positive,

because the negative effect of the word *"hate"* is shifted by the expression *"don't"*. For a simple bag-of-words representation, the sentences "People hate Obama" and "People don't hate Obama" are considered very similar (for several similarity measures). Many approaches have been proposed to deal with negations: Das and Chen [94] suggest to attach a "NOT" tag to words which occur close to negation terms, so the sentence "I don't like iPhone" would be treated as "I *like-NOT* iPhone", and thus the word "*like-NOT*" would be considered as a different word from "like". However, not all expressions (not even negations) have this effect, so their effect must be carefully considered. For instance, it would be incorrect to consider negative a sentence like "This iPhone does not only make calls, it does wonders".

When dealing with social network, however, we also need to consider that people do not use expressions typical of regular documents or reviews, but rather an informal lexicon with misspellings and abbreviations, due also to the limited length allowed for a message.

Here we list some peculiarities of the informal language used in social media:

- *Abbreviations* Given the limited space allowed for posts in many social media, users tend to abbreviate common constructs. Sometimes, these abbreviations become so common that they start being used even in social media where the length of the post is not limited. Examples of this *"internet slang"* are "btw" (by the way), "thx" (thanks) and "any1" (anyone).

- *Misspellings* The other common characteristic when users write posts on social media is that they do it fast, and often without checking what they write. This lead to a relatively large amount of misspellings in the posts, such as "coul" instead of "cool", "teh" instead of "the". Sometimes, the difference between abbreviation and misspelling is subtle (e.g. "lovin" instead of "loving" is often intentional).

- *Stretched words* While the space is often limited, users tend sometimes to express their mood or opinion by stretching short words, as for "looooooove" instead of "love" or "happyyyyy" instead of "happy". While this can be considered an obstacle to sentiment classification (the words must be reconducted to their original form), stretched words can also be used as new features that help to understand the sentiment expressed in the post.

- *Emoticons* Finally, another common behavior when posting on social media is the use of emoticons, i.e. sequence of characters aimed at resembling facial expressions, like :-) (for happiness) or :-((for sadness). Emoticons have been used in sentiment analysis classification to improve the understanding of the user opinion [95], usually by replacing them with tags like "*EMOTICON-POS*" or "*EMOTICON-NEG*" to be used as features by

the classifier. Sometimes, the emoticons are even used as sole features to determine the sentiment of a post, in a procedure called *distant supervision* [96].

4.2.2 Adding Relations to Text-based Sentiment Analysis

Sentiment analysis has been mostly addressed as a pure text classification problem, even when dealing with data from social media. All these approaches, however, completely disregard the fact that social media are networked environments, where the interactions among users can affect the way they think about multiple topics and, ultimately, what they write in their posts.

Therefore, relationships among users in social networks may provide important insights about the sentiment of the posts they write.

People generally have significant relationships with others who tend to be like themselves over a number of different aspects. The principle behind this idea is called Homophily, as described in detail in Sec. 3.4.1.

Some studies have been proposed in literature to exploit relationships in order to improve the results obtained using only text content.

However, many of them do not consider the principle of homophily, but simply consider users relationships as additional features to use with a standard classifier, for example by counting how many ties a user has.

Gryc and Molainen [97] addressed sentiment analysis of political blogs using a Naive Bayes Multinomial classifier, considering both text-based and relations-based groups of features. The text-based features were based on a bag-of-words approach and by selecting sentiment-specific words. The authors also proposed to use ”social network features”, based on characteristics of the nodes in the network: for example, the rank of the nodes or the size of the connected component to which nodes belong.

A similar example is given by the work of Shams et al. [98], where the authors aim at classifying customer preferences from online shopping websites. The authors split the network into subgroups and use these subgroups as additional features for a Support Vector Machine classifier.

However, both these methods disregard much of the information included in the relationships among users, as the additional features do not incorporate principles like *homophily* and *social influence*.

Other algorithms, instead, directly exploit the relational structure using more complex models.

Hu et al. [99] propose a method to estimate post-level sentiment by exploring tweet content and friendship relationships among users. They formulate an optimization problem based on three

components: a component for modeling textual content, that considers messages as independent and identically distributed (the model is generated by a Least Squares method); a component for modeling message-message relations, built starting from user-user relationships and user-message relationships; and a component for handling noisy and short text messages, using a sparse representation of the text which uses only few phrases and words.

The full model is then optimized to find the most fitting sentiment polarity for the social network messages.

While most approaches focus on post-level analysis, some authors have addressed the problem of inferring user-level sentiments. Smith et al. [100] and Deng et al. [101] assume that user-level sentiment can be computed by aggregating the sentiment of all of the user's posts. Although the sentiments of users are correlated with the sentiments expressed in tweets, such simple aggregation can often produce incorrect results, in part because sentiment extracted from short texts such as tweets will generally be very noisy and error prone [102].

Speriosu et al. [103] proposed to infer both post- and user-level sentiment by exploiting the relational structure through label propagation. The authors consider the data as an heterogeneous undirected network composed of user nodes and post nodes, connected among them by weighted edges. The content of posts is evaluated using two classifiers, a baseline lexicon classifier (where positive and negative words are counted to determine the sentiment) and a maximum entropy classifier which considers features like unigrams, bigrams and emoticons. Label propagation works as a controlled random walk, which ensures that node labels are propagated through the network.

Tan et al. [104] analyze user-level sentiment using a semi-supervised probabilistic model, composed of two feature functions: a user-post feature function, which accounts for the probability that a user having a given sentiment posts messages having a given sentiment (e.g. a negative user may post a positive message); and a user-user feature function, which accounts for the probability that a user having a given sentiment has a neighbor with another given sentiment (e.g. a positive user may have a neighbor whose sentiment is negative). Each user is influenced only by his direct neighbors. The authors then use an algorithm to find the combination of sentiment labels for each user that maximized the probabilistic model.

However, the authors explicitly state that their training data lacks the annotation of the posts, which is replaced by an automatic annotation where positive users have only positive posts and viceversa, thus lacking important knowledge for the model.

In a more recent work, Zhu et al. [102] try to overcome the problems of the availability of labeled data by employing a fully unsupervised method. Their algorithm address both user-level and post-level sentiment analysis by performing clustering on a tripartite graph, i.e. a graph composed of three types of nodes: users, posts and features (single words). The clustering

algorithm then partitions the graph into three clusters, one for positive elements, one for negative elements and one for neutral elements.

In the following sections we will take inspiration from these methods, and we will try to overcome some of the limitations arisen in the literature by proposing our own methods for text- and relation-based sentiment analysis.

4.3 Relational Model for User-Level Opinion Detection

The aim of Sentiment Analysis is to define automatic tools able to extract subjective information, such as opinions and sentiments from texts in natural language, in order to create structured and actionable knowledge to be used by either a decision support system or a decision maker [105].

Most of the existing works, as discussed in the previous section, suffer of two major limitations: first, they are based only on the textual information expressed in social media, not considering that they are networked environments; second, they focus on the opinion expressed by the single post, but they do not consider that the actor of an opinion is the user. The few approaches that address user-level sentiment analysis generally aggregate the sentiment of the single posts. Although the sentiment of users is correlated with the sentiment expressed in their posts, such simple aggregation can often produce incorrect results, because sentiment extracted from short texts such as tweets (which in Twitter are limited to 140 characters) will generally be very noisy and error prone.

Moreover, there is another aspect to consider. Although social relations play a fundamental role in opinion mining on social networks, considering friendship connections is a weak assumption for modelling homophily: two friends might not share the same opinion about a given topic.

Taking into account all these considerations, we propose a framework to model user-label polarity classification by integrating post contents with approval relations (e.g. 'Like' button on Facebook and 'Retweet' on Twitter) that could better represent the principle of homophily. For this reason, we exploit the concept of *Approval Network* that was presented in Sec. 3.4.1. In the following we present a semi-supervised learning paradigm that, given a small proportion of users already labeled in terms of polarity, predicts the remaining unlabeled user sentiments.

4.3.1 Approval Model

We represent the approval network using the HN-DAG representation introduced in Sec. 3.4.1.

Given a HN-DAG, we introduce a vector of labels $\mathbf{L}^V = \{l(v_i) \in \{+,-\} | v_i \in V\}$ that defines each user as either "positive" (+) or "negative" (-) and an analogous vector of labels $\mathbf{L}^P =$

$\{l_{v_i}(p_t) \in \{+,-\} | v_i \in V, p_t \in P\}$ that represents the polarity label of each post p_t written by user v_i. Our model is intended to obey to the Markov assumption: the sentiment $l(v_i)$ of the user v_i is influenced by the sentiment labels $l_{v_i}(p_t)$ of his posts and the sentiment labels of the directly connected neighbors $N(v_i)$. This assumption leads us to adapt the probabilistic model defined in [104] to combine user-post and user-user (approval) relations:

$$\begin{aligned} \log P(\mathbf{L}^V) = & \left(\sum_{v_i \in V} \left[\sum_{p_t \in P, \alpha, \beta} \mu_{\alpha,\beta} f_{\alpha,\beta}(l(v_i), l_{v_i}(p_t)) \right.\right. \\ & \left.\left. + \sum_{v_j \in N(v_i), \alpha, \beta} \lambda_{\alpha,\beta} g_{\alpha,\beta}(l(v_i), l(v_j)) \right] \right) \\ & - \log Z \end{aligned} \tag{4.1}$$

where $\alpha, \beta \in \{+,-\}$, $f_{\alpha,\beta}(\cdot,\cdot)$ and $g_{\alpha,\beta}(\cdot,\cdot)$ are feature functions that evaluate user-post and user-user relations respectively, and $\mu_{\alpha,\beta}$ and $\lambda_{\alpha,\beta}$ are parameters to be estimated. In particular, $\mu_{\alpha,\beta}$ represent the weights considering the setting where a user with label α posts a message with label β, while $\lambda_{\alpha,\beta}$ denote the weights considering the setting where a user with label α is connected to a user with label β. Z is the normalization factor that enables a coherent probability distribution of $P(\mathbf{L}^V)$.

Before introducing how modeling user-post and user-user relations through the corresponding feature functions, we need to distinguish two categories of users: labeled (**black nodes** in the following), where the polarity labels are known, and unlabeled (**white nodes**), where polarity labels are unknown.

User-post feature function A user-post feature function evaluates whether post polarity agrees (or disagrees) with respect to the user sentiment. Formally, $f_{\alpha,\beta}(l(v_i), l_{v_i}(p_t))$ is defined as:

$$f_{\alpha,\beta}(l(v_i), l_{v_i}(p_t)) = \begin{cases} \frac{\rho_{T-black}}{|P_{v_i}|} & l(v_i) = \alpha, l_{v_i}(p_t) = \beta, v_i \text{ black} \\ \frac{\rho_{T-white}}{|P_{v_i}|} & l(v_i) = \alpha, l_{v_i}(p_t) = \beta, v_i \text{ white} \\ 0 & otherwise \end{cases} \tag{4.2}$$

where $\rho_{T-black}$ and $\rho_{T-white}$[1] represent our different level of confidence in black and white users, and $P_{v_i} \subset P$ denotes the set of posts written by user v_i.
We point out that f assumes that every $p_t \in P$ has been classified. A methodology for automatically labeling posts is described in Sect. 4.3.3.

User-user feature function A user-user feature function evaluates whether the polarity of a given user agrees (or disagrees) with his neighbor's sentiment. More specifically, we redefine

[1] Note that $\rho_{T-black}$, $\rho_{T-white}$ and ρ_{neigh} are empirically estimated (see Sect. 4.3.4.3).

the original feature function introduced in [104] in order to deal with approval networks. Given a HN-DAG we can formally define $g_{\alpha,\beta}(l(v_i),l(v_j))$ as follows:

$$g_{\alpha,\beta}(l(v_i),l(v_j)) = \begin{cases} \frac{\rho_{neigh} \cdot c_{i,j}}{\sum_{v_k \in N(v_i)} c_{i,k}} & l(v_i)=\alpha, l(v_j)=\beta \\ 0 & otherwise \end{cases} \tag{4.3}$$

where ρ_{neigh}[2] represents the level of confidence in relationships among users.

4.3.2 Parameter Estimation and Prediction

We now address the problem of estimating μ and λ for inferring the assignment of user sentiment labels which maximises $\log P(\mathbf{L}^V)$. Starting from a small set of labeled data, we can initialise the values of μ and λ using the following approach:

$$\mu_{\alpha,\beta} = \frac{\sum_{(v_i,p_t)\in E_{BU}} I(l(v_i)=\alpha, l_{v_i}(p_t)=\beta)}{\sum_{(v_i,p_t)\in E_{BU}} I(l(v_i)=\alpha, l_{v_i}(p_t)=+) + I(l(v_i)=\alpha, l_{v_i}(p_t)=-)} \tag{4.4}$$

$$\lambda_{\alpha,\beta} = \frac{\sum_{(v_i,v_j)\in E_{BP}} I(l(v_i)=\alpha, l(v_j)=\beta)}{\sum_{(v_i,v_j)\in E_{BP}} I(l(v_i)=\alpha, l(v_j)=+) + I(l(v_i)=\alpha, l(v_j)=-)} \tag{4.5}$$

where $I(\cdot)$ is the indicator function, while E_{BU} and E_{BP} are the subsets of user-user and user-post edges where users are labeled. The initial values, estimated according to Eq. (4.4)-(4.5), can be used to derive the optimal λ and μ that maximize $\log P(\mathbf{L}^V)$.

For seek of simplicity, we introduce a change of notation: we decompose $\log P(\mathbf{L}^V)$ in $\phi \cdot \Psi(\mathbf{L}^V)$, where

$$\phi = \{\mu_{\alpha,\beta}, \lambda_{\alpha,\beta}\} \tag{4.6}$$

and

$$\begin{aligned} \Psi(\mathbf{L}^V) \quad = \quad & \{ \sum_{v_i \in V} \sum_{p_t \in P,\alpha,\beta} f_{\alpha,\beta}(l(v_i), l_{v_i}(p_t)), \\ & \sum_{v_i \in V} \sum_{v_j \in N(v_i),\alpha,\beta} g_{\alpha,\beta}(l(v_i), l(v_j))\} \end{aligned} \tag{4.7}$$

In order to find the optimal values of ϕ we employed the SampleRank Algorithm [106]:

In the algorithm shown in Fig. 4.1, *Sample* is a sampling function that randomly chooses an element of $\mathbf{L}^V$ and reverts its polarity. $\mathbb{P}(\mathbf{L}^V_{new}, \mathbf{L}^V)$ is the Accuracy[2] difference between $\mathbf{L}^V_{new}$

[2]Performance measures are defined in Sect. 4.3.4.4

Algorithm 1: SampleRank

Input : HN-DAG
Initialisation: Learning rate η, initial parameters ϕ
Output : Final parameter values ϕ and full label-vector $\mathbf{L}^V$

Randomly initialise $\mathbf{L}^V$;
for *step* $\leftarrow 1$ **to** *MaxNumberOfSteps* **do**
  $\mathbf{L}^V_{new} \leftarrow Sample(\mathbf{L}^V)$;
  $\bigtriangledown \leftarrow \Psi(\mathbf{L}^V_{new}) - \Psi(\mathbf{L}^V)$;
  if $\phi \cdot \bigtriangledown > 0 \wedge \mathbb{P}(\mathbf{L}^V_{new}, \mathbf{L}^V) < 0$ *// performance is worse but the objective function is higher*
  then $\phi \leftarrow \phi - \eta\bigtriangledown$;
  if $\phi \cdot \bigtriangledown < 0 \wedge \mathbb{P}(\mathbf{L}^V_{new}, \mathbf{L}^V) > 0$ *// performance is better but the objective function is lower*
  then $\phi \leftarrow \phi + \eta\bigtriangledown$;
  if *Convergence* **then** break;
  if $\mathbb{P}(\mathbf{L}^V_{new}, \mathbf{L}^V) > 0 \vee (\mathbb{P}(\mathbf{L}^V_{new}, \mathbf{L}^V) = 0 \wedge \phi \cdot \bigtriangledown > 0)$ **then**
    $\mathbf{L}^V \leftarrow \mathbf{L}^V_{new}$;

FIGURE 4.1: SampleRank Algorithm

and $\mathbf{L}^V$ (only on the black nodes).
The algorithm converges when both the objective function $\phi \cdot \Psi(\cdot)$ and $\mathbb{P}(\mathbf{L}^V_{new}, \mathbf{L}^V)$ do not increase for a given number of steps.

4.3.3 Message polarity classification

Since the user-post feature function $f_{\alpha,\beta}$ assumes that every post $p_t \in P$ has to be classified, a sentiment classification methodology for posts is required.
The main polarity classification approaches are focused on identifying the most powerful model for classifying the polarity of a text source. However, an ensemble of different models could be less sensitive to noise and could provide a more accurate prediction [107].

For this reason we exploit the ensemble method proposed in [105], where the weighted contribution of each classifier is used to make a final label prediction. The work improves the original Bayesian Model Averaging (BMA) [108] by explicitly taking into account the marginal distribution of each classifier prediction and its overall reliability when determining the optimal label.

Given a set of $\mathcal{R}$ classifiers, the approach assigns to a post p_t the label $l_{v_i}(p_t)$ that maximizes:

$$\begin{aligned} P(l_{v_i}(p_t)|\mathcal{R},\mathcal{D}) &= \sum_{r\in\mathcal{R}} P(l_{v_i}(p_t)|r)P(r|\mathcal{D}) \\ &= \sum_{r\in\mathcal{R}} P(l_{v_i}(p_t)|r)P(r)P(\mathcal{D}|r) \end{aligned} \tag{4.8}$$

where $P(l_{v_i}(p_t)|r)$ is the marginal distribution of the label predicted by classifier r, while $P(\mathcal{D}|r)$ represents the likelihood of the training data $\mathcal{D}$ given r. The distribution $P(\mathcal{D}|r)$ can be approximated by using the F_1-measure obtained during a preliminary evaluation of classifier r:

$$P(\mathcal{D}|r) \propto \frac{2\times P_r(\mathcal{D})\times R_r(\mathcal{D})}{P_r(\mathcal{D})+R_r(\mathcal{D})} \tag{4.9}$$

where $P_r(\mathcal{D})$ and $R_r(\mathcal{D})$ denotes Precision and Recall obtained by classifier r.

The set of baseline classifiers used by BMA is composed of dictionary-based classifier, Naïve Bayes (NB), Maximum Entropy, Support Vector Machines (SVM) and Conditional Random Fields (CRF). The TF weighting schema has been considered for NB and SVM. For further details, see [105].

4.3.4 Experiments

We present a case study to validate the proposed model based on approval networks. In particular, we focused our experimental investigation on connections derived from Twitter and we compared the ability of inferring user-level polarity classification with traditional approaches based only on textual features.

4.3.4.1 Dataset

In order to evaluate the proposed model, we need a dataset composed of:

1. A set of users (V) and their manually tagged sentiment labels about a specific topic q;
2. Tweets written by users $\in V$ about a specific topic q with their manually tagged sentiment labels;
3. Retweet network about a specific topic q.

To the best of our knowledge, no datasets containing all the above information are available. In order to easily create a reliable toy dataset[3], the following steps have been performed:

- Crawling of a set of 2500 users from Twitter who tweeted about the topic 'Obama' during the period 8-10 May 2013;
- Download of the last 3200[4] tweets for each user;
- Filtering of out-of-topic posts by removing messages that do not match the regular expression "`obama|barack`";
- Selection of the most active users by considering those authors who emitted at least 50 tweets about Obama[5];
- Manual annotation of each tweet and user by 3 annotators for labelling the corresponding polarity (62 users and 159 tweets). Only positive and negative tweets have been considered;

Considering that our model has been defined for dealing with a semi-supervised environment, we need to distinguish labeled (black) from unlabeled (white) users. As black nodes we considered those users whose bio (description on Twitter) or name clearly state a positive or negative opinion about the topic 'Obama'. For instance, a positive user's bio could report "*I like football, TV series and Obama!*" and/or the name could be"*ObamaSupporter*".

4.3.4.2 BMA Settings

The Bayesian Model Averaging model has been trained by using positive and negative tweets of the *Obama-McCain Debate (OMD)*[6] dataset. As a test set, we employed the 159 tweets written by the users of our network.

Since much of the tweet are similar to SMS messages, the writing style and the lexicon of tweets is widely varied. Moreover, tweets are often highly ungrammatical, and filled with spelling errors. In order to clean the dataset, we captured a set of patterns, which are detected using dictionaries a priori defined and regular expressions. The normalised tweet is then used to train the text-based BMA. The applied filters are:

- **HTML Links**: All tokens matching the following REGEXP are deleted:

  `(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|];`

[3]The dataset can be downloaded at `http://www.mind.disco.unimib.it`
[4]Due to a limit imposed by Twitter
[5]50 is the average number of tweets posted by the users.
[6]`https://bitbucket.org/speriosu/updown/src/5de483437466/data/`

- **Hashtags**: The symbol # is removed from all the tokens;
- **Mention Tags**: The tokens corresponding to a mention tag, identified through the REGEXP @(.+?), are removed;
- **Retweet Symbols**: All the tokens matching the expression RT are removed;
- **Laughs**: If a token has a sub-pattern matching `((a|e|i|o|u)h|h(a|e|i|o|u))\\1+|(ahha|ehhe|ihhi|ohho|uhhu)+`, then the whole token is replaced with `TAGLAUGHT`;
- **Emoticons**: In order to detect positive and negative emoticons, two dictionaries have been defined. If a token appears in the dictionary of positive emoticons then it is replaced with `POSEMOTICON`, otherwise with `NEGEMOTICON`;
- **Sentiment Expressions**: Several sentiment expressions are used in English. Expressions such as 'ROFL','LMAOL','LMAO','LMAONF' represent positive expressions. In order to facilitate Term-Frequency (TF) counting, they are replaced with `POSEMOTICON` and `NEGEMOTICON`, as well;
- **Slang correction**: A dictionary of a priori defined slang expressions with their meaning, such as 'btw' (by the way), 'thx' (thanks), 'any1' (anyone) and u (you) has been built. In order to facilitate TF counting, each found slang expression is replaced with its meaning;
- **Onomatopoeic expressions**: as the previous point, a mapping dictionary has been defined for onomatopoeic expressions, such as 'bleh' (`NEGEMOTICON`) and 'wow' (`POSEMOTICON`).

In addition to filters, misspelled tokens have been corrected using the Google's Spell Checker API[7]. Since the Google's algorithm takes the neighbourhood (context) of a misspelled token into account in suggesting the correction, the whole previously filtered tweet is considered as a query rather than the single token.
Note that the adaptation and modifications of the REGEXPs adopted in these filters to other social networks is straightforward.

4.3.4.3 SampleRank and Approval Model Settings

Regarding SampleRank Algorithm, we set the maximum number of steps to 10000 and we assume that convergence is reached when results are persistent for 500 steps.
Considering that SampleRank results depend on a sampling function, we performed $k = 1, 5, 11, 15, 21, 101$ runs to get k predictions (votes) and take a majority vote among the k possible labels for each user. For each k, we performed 500 experiments to compute the average performance.

[7] `https://code.google.com/p/google-api-spelling-java/`

SampleRank and the approval model need parameters to be fixed a priori: the learning rate η, $\rho_{T-black}$, $\rho_{T-white}$ and ρ_{neigh}. In order to select the optimal combination, we empirically varied their values[8] and hold those which have exhibited the highest performance: $\eta = 0.01$, $\rho_{T-black} = 1$, $\rho_{T-white} = 0.2$ and $\rho_{neigh} = 0.9$.

Since SampleRank is a time expensive algorithm, it has been implemented using Fracture[9], a Java Multi Core library that allows to parallelise the computation on all the machine cores.

4.3.4.4 Results

The performance of the proposed model has been evaluated by measuring the well known Precision, Recall and F-Measure:

$$P = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad R = \frac{\text{TP}}{\text{TP}+\text{FN}} \quad F1 = \frac{2 \cdot P \cdot R}{P+R} \tag{4.10}$$

both for the positive and negative labels (in the sequel denoted by P_+, R_+, $F1_+$ and P_-, R_-, $F1_-$ respectively). We also measured Accuracy as:

$$\text{Acc} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}} \tag{4.11}$$

Concerning BMA, the performance computed on the 159 tweets achieves 60.37% of Accuracy, compared with the 58.49% obtained by the baseline classifier which reaches the highest performance (CRF). These results confirm the performance improvement stated in [105] when using BMA. In order to estimate the user polarity by using their tweets, we considered a majority voting mechanism to infer the final user label. For instance, if a user posts 3 positive and 2 negative tweets, the final user label will be 'positive'. When inferring labels of white nodes on the collected dataset, this heuristic reaches a 66.66% of Accuracy.

Regarding the proposed model based on approval relationships, we firstly investigate its robustness by monitoring results according to the considered k votes. In order to select the best value of k, we tried to minimise the model bias by considering tweets manually labeled.
Fig. 4.2 shows the negative correlation between the number of votes and the Accuracy variability, suggesting that a high number of votes ensures the stability of the model. The average Accuracy values and the confidence intervals (with a confidence level of 0.01) are reported in Table 4.1.

Once determined the optimal number of k, we accordingly set up our model by using the tweet labels classified by the BMA approach.

[8] Parameters ρ have been varied from 0 to 1 with an increment of 0.1, while η has been empirically fixed to 0.01
[9] `http://kccoder.com/fracture/`

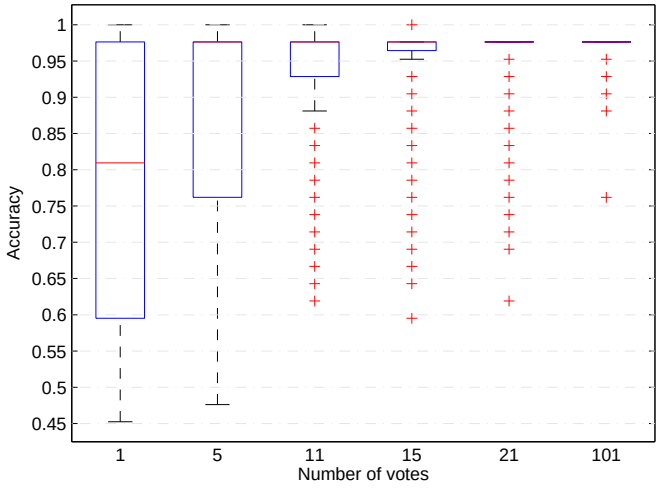


FIGURE 4.2: Negative correlation between the number of votes and the Accuracy variability

TABLE 4.1: Confidence intervals achieved on different experiments

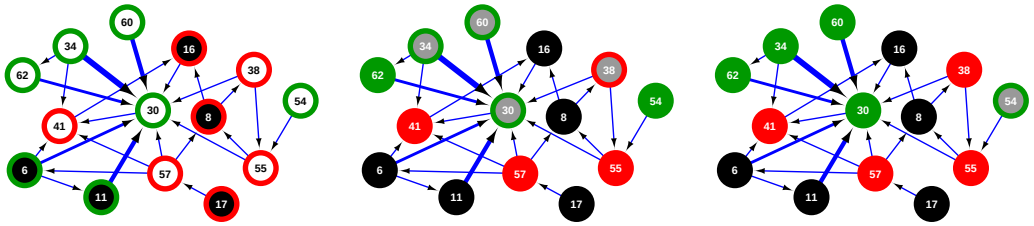
k	1	5	11	15	21	101
Interval	0.783 ± 0.022	0.881 ± 0.015	0.925 ± 0.011	0.936 ± 0.010	0.95 ± 0.007	0.976 ± 0.001

Table 4.2 reports performance achieved by our approach in respect of the BMA one. Note that our model is able to reach 93.8% of Accuracy, significantly outperforming the BMA approach. Approval relations enclosed in our model ensures an improvement of 27% with respect to the text-only method.

TABLE 4.2: Comparison between BMA (only tweets) and Relations (SampleRank) + Tweets (BMA labeled)

	P_-	R_-	$F1_-$	P_+	R_+	$F1_+$	Acc
BMA	0.655	0.826	0.731	0.692	0.474	0.563	0.666
Relations + Tweets	0.933	0.963	0.945	0.958	0.907	0.925	0.938

We report in Fig. 4.3 a snapshot that compares the ground truth network, the prediction provided by BMA and the one obtained by our model. Since BMA does not take into account any kind of relationship, the correct prediction of a user does not have any effect on adjoining users. In our case, the prediction of each user has impact on all the other nodes by a "propagation" effect, smoothing each predicted label according to adjoining nodes. This investigation finally confirms that the inclusion of relationships in predictive models, as suggested in other studies [61, 109, 110], leads to improve recognition performance when dealing with non-propositional environments.



(a) Ground truth (b) Text-only approach (c) Our model

FIGURE 4.3: Example of the studied approval network. (a) is the Ground truth derived from human annotation, (b) the BMA results and (c) results of our model. The real user labels are indicated by the border colour: green for positive and red for negative. Background colour denotes if a node is black or white and the edge thickness reflects the normalised weight. Grey nodes in (b) and (c) represent the misclassified users.

4.4 Latent Representation Model for Post- and User-level Opinion Detection

In the previous section, we have shown that considering friendship connections is a weak assumption for modelling homophily, as two friends might not share the same opinion about a given topic. Approval relationships can be used instead of friendship in order to better capture this influence. However, in the previous contribution the sentiment of the posts is used to infer the sentiment of the users, but not vice versa.

In order to overcome this limitation, in the approach presented in this section we consider social network data as a heterogeneous network, whose nodes and edges can be of different types. Ispired by the work of Jacob et al. [111], who introduced an innovative method for classifying nodes in heterogeneous networks, we propose an approach that can infer at the same time the sentiment relative to each post and the sentiment relative to each user about a specific topic. This algorithm learns a latent representation of the network nodes so that all the nodes will share a common latent space, whatever their type is. This ensures that the sentiment of the posts can influence the sentiment of the users, and in the same way the sentiment of the posts is influenced by that of the users.

For each node type, a classification function will be learned together with the latent representation, which takes as input a latent node representation and computes the sentiment polarity (positive or negative) for the corresponding node.

4.4.1 Preliminaries

In this section we introduce some preliminary concepts that will be used in our model. First, we give a definition of Heterogeneous Approval Network, which summarizes the structure of a social network and the information we require to determine the users' and posts' sentiment

polarity. Then, we give a brief description of the techniques we use to represent and treat the textual data available in the posts.

4.4.1.1 Heterogeneous Approval Network

We assume that a user who approves a given message will share the same opinion with higher probability. In Sec. 3.4.1, we defined as "approval network" a network where the nodes represent users of a social network, and a directed arc connects a user who has approved a post to the original author of that post.

We start from the definition of N-DAG, which represents a network with a single type of node, the users. The previous definition of HN-DAG referred to a heterogeneous graph which could represent both the user-user and user-post relationships. However, the network they defined does not consider relationships among posts. With the following definition, we extend their Heterogeneous Normalized Directed Approval Graph (HN-DAG) so that post-post relationships can be taken in account as well:

Definition 8. *Given a N-DAG*$_q = \{V_q, E_q^{VV}, \mathbf{W}_q^{VV}\}$*, let* $P_q = \{p_1, \cdots, p_m\}$ *be the set of nodes representing posts about q and* $E_q^{VP} = \{(v_i, p_t) | v_i \in V_q, p_t \in P_q\}$ *be the set of arcs that connect the user* v_i *and the post* p_t*. Then, let* $E_q^{PP} = \{(p_{t_1}, p_{t_2}) | p_{t_1}, p_{t_2} \in P_q\}$ *be the set of arcs that connect a post* p_{t_1} *to another post* p_{t_2}*, and* $\mathbf{W}_q^{PP} = \{w_{t_1,t_2} | (p_{t_1}, p_{t_2}) \in E_q^{PP}\}$ *is the set of weights of the post-post edges. An* ***Heterogeneous Normalised Directed Approval Graph*** *is a septuple HN-DAG*$_q = \{V_q, P_q, E_q^{VV}, E_q^{VP}, E_q^{PP}, W_q^{VV}, W_q^{PP}\}$.

4.4.1.2 Vector space document representation

In order to model the posts P generated by the users U, we use a bag-of-words representation.

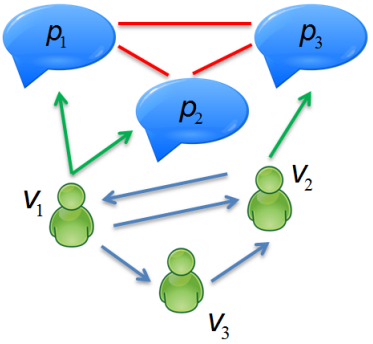


FIGURE 4.4: Example of HN-DAG representing users and posts of a social network, connected by user-user (blue), post-post (red) and user-post (green) relationships.

Given the set of posts P that are represented in the heterogeneous network, let $Q = \{q_1, q_2, \ldots, q_m\}$ be the set of all the unique words occurring in P. Then, a post p_i can be represented by an m-dimensional vector $\overrightarrow{p_i}$. A usual document encoding for sentiment classification is $\text{tf}(i,q)$, which is the frequency of a word $q \in Q$ in post p_i. Then, the vector representation of the post is:

$$\overrightarrow{p_i} = (\text{tf}(i,q_1), \text{tf}(i,q_2), \ldots, \text{tf}(i,q_m)) \tag{4.12}$$

In this work, we define the weights of the post-post edges as the value of similarity between each couple of posts. With document represented by vectors, we can measure the degree of similarity of two posts as the correlation between their corresponding vectors, which can be further quantified as the cosine of the angle between the two vectors (*Cosine Similarity*). Let $\overrightarrow{p_a}$ and $\overrightarrow{p_b}$ be the vector representation respectively of posts p_a and p_b. Their cosine similarity is computed as follows:

$$\text{similarity} = \frac{\overrightarrow{p_a} \cdot \overrightarrow{p_b}}{\|\overrightarrow{p_a}\| \|\overrightarrow{p_b}\|} = \frac{\sum_{j=1}^{l} p_{aj} \times p_{bj}}{\sqrt{\sum_{j=1}^{l} (p_{aj})^2} \sqrt{\sum_{j=1}^{l} (p_{bj})^2}} \tag{4.13}$$

4.4.2 Latent space Heterogeneous Approval Model

Following the work of Jacob et al. [111], we propose a model that can, at the same time, learn the latent representation of the nodes and infer their sentiment polarity. Differently from previous works, this model performs sentiment polarity classification on all the nodes of the network HN-DAG shown in Sec. 4.4.1.1, that means we can infer the polarity for both users and posts simultaneously.

Each of the nodes, whatever their type is, is represented by a vector space model so that all of them will share the same common latent space. The model will therefore learn the proper representation of each node, and at the same time it will learn a classification function on the latent space. This ensures that the sentiment of the posts can influence the sentiment of the users, and vice versa.

The classification function will take as input a latent node representation in order to compute the polarity (positive or negative) for the corresponding node.

The proposed approach can be summarized with the following steps:

- Each node is mapped onto a latent representation in a vector space $\mathbb{R}^Z$ where Z is the dimension of this space. This latent representation will define a metric in the $\mathbb{R}^Z$ space

such that two connected nodes will tend to have a close representation, depending on the weight of the connection (*smoothness assumption*).

The latent representation for the nodes is initialized randomly.

- A classification function for inferring the polarity of the nodes is learned on the network starting from the latent representations. Nodes with similar representations will tend to have the same sentiment polarity.

In other words, both graph and label dependencies between the different types of nodes will be captured through this learned mapping onto the latent space.

In the following we describe in details the components of the proposed approach.

Given the latent representation $z_i \in \mathbb{R}^Z$ of the i-th node, we want to predict the related sentiment y_i. We are therefore searching for a linear classification function f_θ, where θ are the parameters of the linear regression. This function is learned by minimizing the classification loss on the training data:

$$\sum_{i \in \mathcal{T}} \Delta(f_\theta(z_i), y_i) \tag{4.14}$$

where $\Delta(f_\theta(z_i), y_i)$ is the loss to predict $f_\theta(z_i)$ instead of the real label y_i, and $\mathcal{T}$ is the training set.

In order to make sure that connected nodes have similar representations, we introduce the other following loss:

$$\sum_{i,j:w_{i,j} \neq 0} w_{i,j} \|z_i - z_j\|^2 \tag{4.15}$$

which forces the approach of the latent representation of connected nodes. The complete loss function is the aggregation of the classification and similarity loss:

$$L(z, \theta) = \sum_{i \in \mathcal{T}} \Delta(f_\theta(z_i), y_i) + \lambda \sum_{i,j:w_{i,j} \neq 0} w_{i,j} \|z_i - z_j\|^2 \tag{4.16}$$

This loss will allow us to find the best classification function and, at the same time, improve the meanings of the latent space.

In the original work of [111], the authors fixed a value of λ for all the possible edges. In our work, we decided to model the problem with three different parameters to give different weights to different types of edge, instead of a single parameter λ. Three new parameters are introduced: λ_{pp} refers to the edges connecting two posts, λ_{pv} refers to the edges connecting a post to a user and λ_{vv} refers to the edges connecting two users.

Following this idea, the loss function in Eq. 4.16 can be rewritten as follows:

Algorithm 2 Learning$(x, w, \varepsilon, \lambda)$

for A fixed number of iterations **do**
  Choose (x_i, x_j) randomly with $w_{i,j} > 0$
  if $x_i \in \mathcal{T}$ **then**
    $\theta \longleftarrow \theta + \varepsilon \nabla_\theta \Delta(f_\theta(z_i), y_i)$
    $z_i \longleftarrow z_i + \varepsilon \nabla_{z_i} \Delta(f_\theta(z_i), y_i)$
  end if
  if $x_j \in \mathcal{T}$ **then**
    $\theta \longleftarrow \theta + \varepsilon \nabla_\theta \Delta(f_\theta(z_j), y_j)$
    $z_j \longleftarrow z_j + \varepsilon \nabla_{z_j} \Delta(f_\theta(z_j), y_j)$
  end if
  $z_i \longleftarrow z_i + \varepsilon \lambda \nabla_{z_i} w_{i,j} \|z_i - z_j\|^2$
  $z_j \longleftarrow z_j + \varepsilon \lambda \nabla_{z_j} w_{i,j} \|z_i - z_j\|^2$
end for

FIGURE 4.5: Stochastic Gradient Descent Algorithm

$$L(z, \theta) = \sum_{i \in \mathcal{T}} \Delta(f_\theta(z_i), y_i) + \lambda_{vv} \sum_{\substack{i,j: w_{i,j} \neq 0 \\ i \in V \wedge j \in V}} w_{i,j} \|z_i - z_j\|^2 + \tag{4.17}$$
$$\lambda_{pv} \sum_{\substack{i,j: w_{i,j} \neq 0 \\ i \in V \wedge j \in P}} w_{i,j} \|z_i - z_j\|^2 +$$
$$\lambda_{pp} \sum_{\substack{i,j: w_{i,j} \neq 0 \\ i \in P \wedge j \in P}} w_{i,j} \|z_i - z_j\|^2$$

The minimization of the loss function (Eq. 4.17) is performed by exploiting a Stochastic Gradient Descent Algorithm (see Fig. 4.5). The algorithm first chooses a pair of connected nodes randomly. After that, if the node is in the training set $\mathcal{T}$ it modifies the parameters of the classification function and the latent representation according to the classification loss following Eq. 4.14. Successively, it updates the latent representation of both the nodes depending on the difference between the two representation presented in Eq. 4.15.

4.4.3 Experiments

In order to evaluate the proposed approach, we used the "Obama" dataset described in Sec. 4.3.4.1.

Starting from this dataset, we built a HN-DAG, where the set of nodes V represent the set of users who posted something about the topic 'Obama', and the set P represent the tweets that those users posted about 'Obama'.

We have three types of arcs connecting the nodes:

- the arcs connecting a user to another user, which weight is determined by the normalized number of retweets;
- the arcs connecting a user to a post, which in our case have 0/1 weights;
- the arcs connecting a post to another post, whose weight is determined by the cosine similarity between the two posts, as explained in Sec. 4.4.1.2.

In order to assess the importance of relationships for determining the sentiment polarity of users and posts, we compare our method with two well-known approaches based only on the analysis of the textual data: a Support Vector Machine (SVM) and a L2-regularized logistic regression (LR). When only content is used, the posts are classified as positive or negative based on their content, while the users are classified based on the total polarity of their posts (the posts of a single user are merged and considered as a single document for determining the user's polarity).

We used the Support Vector Machine package available in LibSVM [112], using a linear kernel and default settings. The linear regression model was based on the library for large linear classification LibLinear [113].

We have considered as evaluation measures Precision(P), Recall(R) and F_1-measure for both positive (P^+, R^+, F_1^+) and negative (P^-, R^-, F_1^-) class.

We also measured Accuracy as:

$$Acc = \frac{\text{\# of instances successfully predicted}}{\text{\# of instances}} \tag{4.18}$$

The performance of the proposed model can be affected by the randomness of the learning algorithm, leading to less-than-optimum results. In order to reduce this effect and improve the robustness of the classification, we used a majority voting mechanism to label the instances. In particular we performed k = 1, 5, 11, 15, 21 and 101 runs to get k predictions (votes) and we took a majority vote among the k possible labels for each node. For each k, we performed 100 experiments and considered their average performance. In the following, we report the results for $k = 21$, which show a good trade-off between the performance variability and the computational complexity.

The total number of iterations of the learning algorithm has been set to 4000000, while the gradient step ε have been set to 0.1. The size of the latent representation has been set to 40.

4.4.4 Results

Initially, we tested the performance of our approach by considering a case where 66% of the nodes (randomly chosen) are considered as known. The proposed model is strongly influenced

by the parameters λ_{pp}, λ_{pv} and λ_{vv} assigned to the different types of edges. Therefore, for each λ_i, where $i \in \{vv, pp, pv\}$, we investigated different values varying in the range $\{0.01, 0.05, 0.1\}$.

In Table 4.3 and Table 4.4 we reported the best combinations of λ_i for classifying posts and users. The choice of the configuration is, at the current time, an empirical estimate. For the following experiments, we considered a trade-off between predicting the users and posts polarity, and therefore we chose as best configuration $\lambda_{pp} = 0.05$, $\lambda_{pv} = 0.05$, $\lambda_{vv} = 0.1$, as highlighted in the tables.

We compare the results obtained with these settings with the results achieved by the two textual approaches (see Table 4.5). The Latent space Heterogeneous Approval Model (LHAM) outperforms both Support Vector Machine (SVM) and Linear Regression (LR) when predicting the polarity of the posts (around 5% improvement), and strongly outperforms them when predicting the polarity of users (more than 34% of improvement in terms of accuracy).

In order to reduce the bias introduced by empirically choosing the values of λ_i, we computed the average performance over all possible combinations in the range $\{0.01, 0.05, 0.1\}$. The results (as reported in the last column of Table 4.5) show that our method still outperform the baseline algorithms when predicting the polarity of the users, maintaining a 33% of improvement in terms of accuracy, while maintaining a comparable performance when predicting the polarity of the posts.

TABLE 4.3: Best configurations of λ_i for inferring the user polarity. The highlighted line represents the chosen configuration.

λ_{vv}	λ_{pp}	λ_{pv}	P+	R+	F1+	P-	R-	F1-	Acc
0.01	0.01	0.01	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.01	0.05	0.01	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.05	0.01	0.01	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.05	0.01	0.05	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.05	0.01	0.1	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.05	0.05	0.01	0.905	0.836	0.868	0.89	0.933	0.91	0.895
0.05	0.05	0.05	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.05	**0.05**	**0.1**	**0.91**	**0.841**	**0.873**	**0.887**	**0.93**	**0.907**	**0.895**
0.05	0.1	0.05	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.05	0.1	0.1	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.1	0.01	0.05	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.1	0.01	0.1	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.1	0.05	0.01	0.925	0.839	0.878	0.913	0.953	0.932	0.914
0.1	0.05	0.05	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.1	0.05	0.1	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.1	0.1	0.05	0.91	0.841	0.873	0.887	0.93	0.907	0.895
0.1	0.1	0.1	0.91	0.841	0.873	0.887	0.93	0.907	0.895

TABLE 4.4: Best configurations of λ_i for inferring the post polarity. The highlighted line represents the chosen configuration.

λ_{vv}	λ_{pp}	λ_{pv}	P+	R+	F1+	P-	R-	F1-	Acc
0.01	0.01	0.01	0.673	0.819	0.738	0.763	0.587	0.661	0.705
0.01	0.05	0.01	0.677	0.819	0.74	0.762	0.594	0.666	0.708
0.05	0.01	0.01	0.629	0.806	0.699	0.643	0.477	0.528	0.644
0.05	0.01	0.05	0.677	0.806	0.734	0.755	0.6	0.666	0.705
0.05	0.01	0.1	0.68	0.819	0.741	0.769	0.6	0.671	0.711
0.05	0.05	0.01	0.639	0.863	0.727	0.813	0.465	0.533	0.667
0.05	0.05	0.05	0.678	0.825	0.743	0.772	0.594	0.668	0.711
0.05	**0.05**	**0.1**	**0.684**	**0.819**	**0.743**	**0.772**	**0.606**	**0.675**	**0.714**
0.05	0.1	0.05	0.671	0.813	0.734	0.756	0.587	0.658	0.702
0.05	0.1	0.1	0.678	0.825	0.743	0.772	0.594	0.668	0.711
0.1	0.01	0.05	0.669	0.794	0.724	0.743	0.594	0.657	0.695
0.1	0.01	0.1	0.676	0.806	0.734	0.755	0.6	0.666	0.705
0.1	0.05	0.01	0.606	0.869	0.707	0.826	0.394	0.481	0.635
0.1	0.05	0.05	0.666	0.806	0.728	0.751	0.581	0.652	0.695
0.1	0.05	0.1	0.669	0.806	0.73	0.751	0.587	0.656	0.698
0.1	0.1	0.05	0.673	0.819	0.738	0.761	0.587	0.661	0.705
0.1	0.1	0.1	0.673	0.806	0.732	0.753	0.594	0.661	0.702

In order to fully validate our approach, we tested it with different sizes of training and test sets. Therefore, we randomly split our dataset with different percentages $\{20, 33, 50, 66, 80\}$. Given the small size of the dataset, we perform a cross-validation by repeating the random split 30 times for each percentage, and therefore obtain significant results.

Table 4.6 and Table 4.7 show the results of posts and users classification, performed by our model and baseline models depending on training set percentage. It is clear from the tables that our model outperforms other approaches in most of the cases, in particular when the size of the training set has a larger number of instances. While the post classification shows a slight improvement by our model over SVM and Linear Regression, for user classification we are able to achieve far better results than text-only based approaches.

While our model improves its performance for larger training set sizes, the other methods do not improve, and their performance can even decrease. The most probable explanation of this behaviour is that short-text posts are very noisy: a text-only approach is therefore more affected by the introduction of more training instances (which are regarded as more noise), while our

TABLE 4.5: Accuracy of users and post classification for different algorithms.

	LR	SVM	LHAM(Best λ_i)	LHAM (Average λ_i)
Users	0.467	0.552	0.895	0.886
Posts	0.66	0.657	0.714	0.680

TABLE 4.6: Accuracy of post classification for different sizes of the training set.

% Training Set	LR	SVM	LHAM
20	0.613	0.597	0.542
33	0.629	0.620	0.662
50	0.642	0.641	0.718
66	0.679	0.679	0.722
80	0.660	0.669	0.739

TABLE 4.7: Accuracy of user classification for different sizes of the training set

% Training Set	LR	SVM	LHAM
20	0.466	0.485	0.570
33	0.494	0.521	0.823
50	0.480	0.512	0.986
66	0.467	0.531	0.982
80	0.447	0.507	0.986

model is able to face this problem with the help of the additional information carried by the edges between different nodes.

The lower perfomance of LHAM for small percentages of the training set is explained by the behaviour of the Stochastic Gradient Descent Algorithm, which randomly chooses a pair of connected nodes at each iteration. When the number of training instances is small, the chance to pick nodes that are not in the training set will be higher. In this case, the latent representations will mostly depend on the similarity among connected nodes, and less on the correct sentiment polarity.

In order to tackle this problem, we modified the algorithm in Fig. 4.5 as follows:

- At the beginning, starting from the training instances we create a list of ”allowed” nodes;
- At each iteration, the algorithm must choose a pair of nodes where at least one of the nodes is in the list of ”allowed” nodes;
- At the end of each iteration, if one of the chosen nodes was not in the list, it is added; if all the existing nodes have been added, the list is again initialized with the training instances.

The corrected algorithm allows to spread the sentiment polarity information starting from the training nodes, and gradually towards the rest of the network. This permits to outperform the baseline algorithms even when dealing with small training sets both on posts and users classification. At the same time, we maintain a good performance when the training size gets larger (see Fig.4.6 and Fig.4.7).

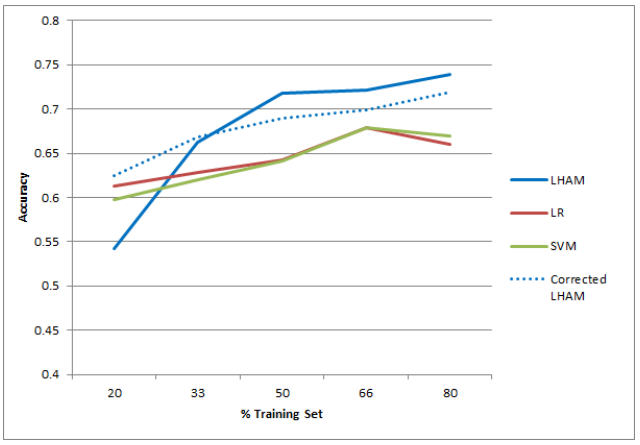


FIGURE 4.6: Accuracy of post classification for different sizes of the training set.

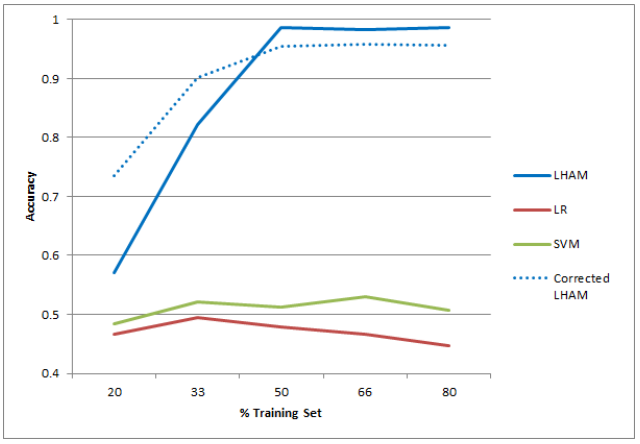


FIGURE 4.7: Accuracy of user classification for different sizes of the training set

4.5 Conclusion

While many approaches have been proposed to address the task of sentiment analysis for generic documents, only recently there have been interest for understanding the opinions written on social media like Twitter and Facebook. Traditional approaches for opinion detection are generally lacking when applied to social network data, because of two major limits: first, they do not consider that the relationships among users strongly influence their opinions on various topics; second, many methods are thought for understanding the opinion expressed in single documents, not the general opinion expressed by a user (often over several posts).

To overcome these limitations, in this chapter we introduced two approaches that conjugate the analysis of textual data with the modeling of user-user relationships, based on the social concept of homophily. Moreover, we use approval relationships rather than friendship connections, to better reflect the influence the users exert on each other.

In the first contribution, we have shown that the model ensures stable performance, significantly outperforming the text-only approach used for comparison.

While the first methods was limited to the analysis of user-level opinion, in the second contribution we overcame that limit. The second approach is able to infer the polarity of users and posts at the same time, using an heterogeneous representation of the network and a latent representation that allows the comparison of heterogeneous objects.

We have shown that the exploitation of the information obtained from the heterogeneous network can improve not only the performance of the classification of users (as already proven by the first method), but also the performance of the classification of posts.

In both works we address the problem in the context of Twitter because it is easier to retrieve data, although adaptation of this framework to other social networks is straightforward (using other approval tools, e.g. 'likes' count on Facebook).

Chapter 5

Conclusion and Future Work

The large diffusion of online social networks that started with Facebook and continues nowadays with Twitter, Flickr, Tumblr and many more has enabled a deep analysis of people behavior. Many applications, from sociology to marketing, can benefit from studying the huge amount of data that users generate every day, from posts in which they express their opinions and sentiments, to actions they take to add friends, share and like content.

Many approaches have been proposed in literature to extract knowledge from social networks. In this thesis we examined in particular those that address two important tasks in social network analysis: community discovery and opinion detection. Several existing methods, however, suffer from an important limitation: they are not able to process both the user-generated content and the network structure.

We therefore built over traditional machine learning approaches to overcome these limitations and created new *relational learning* approaches that are able to deal with both content and relationships at the same time. Four contributions have been described in this thesis. The first two contributions are new algorithms for community detection: one extends a graph-based approach to detect communities based on a traditional concept of community, while the other is a relational clustering method to deal with a new definition of community based on user interests. The following two contributions are two models for opinion detection, both using semi-supervised approaches to improve the inference of polarity in social network. While the first approach is focused only on the polarity of users, exploiting an external classifier for post polarity, the second approach addresses user and post polarity classification at the same time, using a latent representation method to make them comparable objects.

From the experimental results, it is clear that methods exploiting both sources of data outperform methods based on only one of them. From these results we can draw two main conclusions.

First of all, content and relationships appear to provide complementary information. In the four methods introduced in this thesis, we have shown that both by adding content to structure-based approaches, and by adding relationships to content-based approaches, the performance of the algorithms always increases. From neither of the two sources it is possible to obtain all the information provided by the other one, making it a need to consider both of them in machine learning approaches.

The second conclusion is related to the social network analysis tasks considered in this thesis. While community discovery is typically a structure-based task, opinion detection is mainly a content-based tasks. However, the results show that using both sources of data can improve the performance regardless of the type of task.

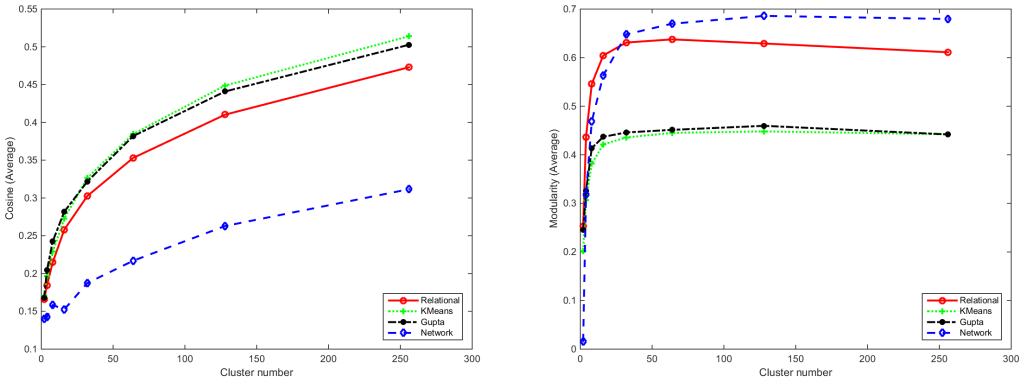
There are several possible researches that can be addressed to further improve the results presented in this thesis.

The first issue is relative to the community discovery task. While many datasets are available with ground truth communities, many of them do not include all the information needed to perform analysis based on content and relationships. Moreover, the definition of community itself is matter of debate, and indeed in Chapter 3 we introduced a new definition for which no ground truth database was ready available. In order to evaluate the results, taking inspiration from clustering methods, we proposed a new performance measure that evaluates the similarity of objects classified in the same community based on both content and relationships. However, this research could be further improved by testing several measures and combining them to find better suited evaluations of the communities. Different measures might be appropriate to use depending on the desired type of community.

A second issue is relative to the computational complexity of the algorithms. By adding a new source of information, we do not only increase the available knowledge, but we also double the time needed to evaluate it. However, algorithms for social network analysis should be fast and efficient, because they generally have to be applied to large amount of data. For this reason, there is a need to develop methods whose computational complexity is low (possibly linear). In this thesis, we have shown that INCA reaches the best performance while maintaining a linear computational complexity. Further work still has to be done to improve the efficiency of the other algorithms.

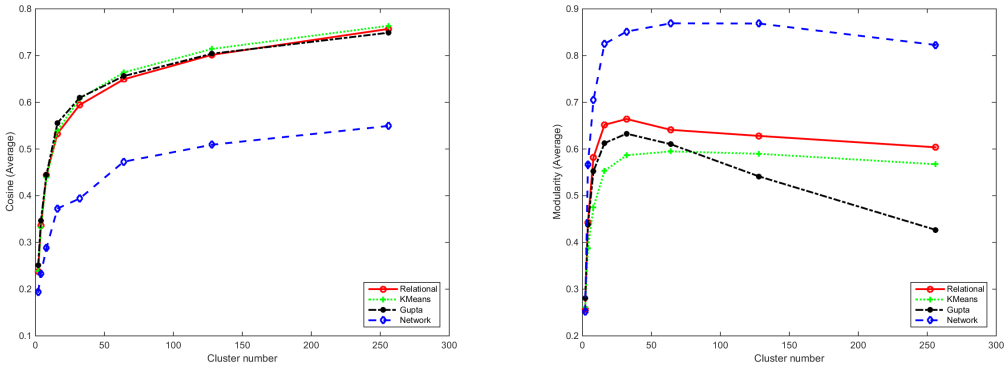
Appendix A

Additional Results for the INCA Algorithm



(a) (b)

FIGURE A.1: Average Results for Cosine Similarity (a) and Modularity (b) on the Prism dataset. Each colored line indicates an algorithm, as follows: INCA (red), Hybrid (black), K-Means (green), Spectral (blue).



(a) (b)

FIGURE A.2: Average results (a) and best results (b) for the Aggregate Index on the Superman dataset. Each colored line indicates an algorithm, as follows: INCA (red), Hybrid (black), K-Means (green), Spectral (blue).

	K-Means	Spectral	Hybrid	INCA
k=2	0.183*	0.028*	**0.199**	0.197
k=4	0.238*	0.181*	0.251	**0.257**
k=8	0.285*	0.237*	0.306	**0.307**
k=16	0.330*	0.240*	0.343*	**0.361**
k=32	0.373*	0.291*	0.374*	**0.409**
k=64	0.413*	0.328*	0.414*	**0.454**
k=128	0.448*	0.380*	0.450*	**0.496**
k=256	0.475	0.427*	0.470*	**0.533**

TABLE A.1: Average Harmonic Mean over 500 runs on the Prism dataset. **Bold** numbers denotes the best performing approach, while underlined numbers represents the second best. Symbol * indicates that INCA outperforms a given baseline by 99% statistical confidence.

	K-Means	Spectral	Hybrid	INCA
k=2	0.240	0.181*	**0.265**	0.232
k=4	0.354*	0.329*	**0.386**	0.379
k=8	0.455*	0.409*	0.493*	**0.503**
k=16	0.546*	0.513*	0.583	**0.586**
k=32	0.597*	0.538*	0.621*	**0.627**
k=64	0.627*	0.612*	0.632*	**0.645**
k=128	0.646*	0.642*	0.612*	**0.663**
k=256	0.651*	0.659*	0.543*	**0.672**

TABLE A.2: Average Harmonic Mean over 500 runs on the Superman dataset. **Bold** numbers denotes the best performing approach, while underlined numbers represents the second best. Symbol * indicates that INCA outperforms a given baseline by 99% statistical confidence.

Bibliography

[1] John Arundel Barnes. *Class and committees in a Norwegian island parish*. Plenum, 1954.

[2] Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 870–878. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

[3] Nada Lavrac. Relational data mining. 2001.

[4] Monica Bianchini, Marco Maggini, and Lakhmi C Jain. *Handbook on Neural Information Processing*. Springer, 2013.

[5] Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[6] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th international conference on World Wide Web*, pages 915–924. ACM, 2008.

[7] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[8] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[9] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[10] David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*, volume 2, pages 259–266. Citeseer, 2002.

[11] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *ACM SIGMOD Record*, volume 27, pages 307–318. ACM, 1998.

[12] David Jensen, Jennifer Neville, and Brian Gallagher. Why collective inference improves relational classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 593–598. ACM, 2004.

[13] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[14] Lise Getoor and Ben Taskar. Introduction to relational statistical learning, 2007.

[15] Nada Lavrac and Saso Dzeroski. Inductive logic programming. In *WLP*, pages 146–160. Springer, 1994.

[16] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *Acm Sigkdd Explorations Newsletter*, 5(1):59–68, 2003.

[17] Diane J Cook and Lawrence B Holder. *Mining graph data*. John Wiley & Sons, 2006.

[18] Sašo Džeroski. Multi-relational data mining: an introduction. *ACM SIGKDD Explorations Newsletter*, 5(1):1–16, 2003.

[19] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[20] Judea Pearl. Probabilistic reasoning using graphs. In *Uncertainty in Knowledge-Based Systems*, pages 200–202. Springer, 1987.

[21] Luc De Raedt, Bart Demoen, Daan Fierens, Bernd Gutmann, Gerda Janssens, Angelika Kimmig, Niels Landwehr, Theofrastos Mantadelis, Wannes Meert, Ricardo Rocha, et al. Towards digesting the alphabet-soup of statistical relational learning. In *NIPS* 2008 Workshop on Probabilistic Programming*, 2008.

[22] Manfred Jaeger. Relational bayesian networks. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 266–273. Morgan Kaufmann Publishers Inc., 1997.

[23] David Heckerman, Chris Meek, and Daphne Koller. Probabilistic entity-relationship models, prms, and plate models. *Introduction to statistical relational learning*, pages 201–238, 2007.

[24] Kristian Kersting. An inductive logic programming approach to statistical relational learning. In *Proceedings of the 2005 conference on An Inductive Logic Programming Approach to Statistical Relational Learning*, pages 1–228. IOS Press, 2005.

[25] Daan Fierens, Hendrik Blockeel, Maurice Bruynooghe, and Jan Ramon. Logical bayesian networks and their relation to other probabilistic logical models. In *Inductive Logic Programming*, pages 121–135. Springer, 2005.

[26] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[27] P Erdős and Alfréd Rényi. Some further statistical properties of the digits in cantor's series. *Acta Mathematica Hungarica*, 10(1-2):21–29, 1959.

[28] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[29] Santo Fortunato and Claudio Castellano. Community structure in graphs. In *Computational Complexity*, pages 490–512. Springer, 2012.

[30] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[31] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004.

[32] Michael R Garey, David S. Johnson, and Larry Stockmeyer. Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976.

[33] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On finding graph clusterings with maximum modularity. *WG*, 7:121–132, 2007.

[34] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[35] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[36] Charu C. Aggarwal. *Social Network Data Analytics*. Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 1441984615, 9781441984616.

[37] Joshua R Tyler, Dennis M Wilkinson, and Bernardo A Huberman. E-mail as spectroscopy: Automated discovery of community structure within organizations. *The Information Society*, 21(2):143–153, 2005.

[38] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19):198701, 2001.

[39] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[40] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[41] Andrea Capocci, Vito DP Servedio, Guido Caldarelli, and Francesca Colaiori. Detecting communities in large networks. *Physica A: Statistical Mechanics and its Applications*, 352(2):669–676, 2005.

[42] Fang Wu and Bernardo A Huberman. Finding communities in linear time: a physics approach. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):331–338, 2004.

[43] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[44] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[45] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[46] Bo Hu, Zhao Song, and Martin Ester. User features and social networks for topic modeling in online social media. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 202–209. IEEE, 2012.

[47] Nagarajan Natarajan, Prithviraj Sen, and Vineet Chaoji. Community detection in content-sharing social networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 82–89. ACM, 2013.

[48] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *Data Mining (ICDM), 2013 IEEE 13th international conference on*, pages 1151–1156. IEEE, 2013.

[49] Yang Zhang, Yao Wu, and Qing Yang. Community discovery in twitter based on user interests. *Journal of Computational Information Systems*, 8(3):991–1000, 2012.

[50] Aditi Gupta, Anupam Joshi, and Ponnurangam Kumaraguru. Identifying and characterizing user communities on twitter during crisis events. In *Proceedings of the 2012 workshop on Data-driven user behavioral modelling and mining from social media*, pages 23–26. ACM, 2012.

[51] Adam Anthony and Marie desJardins. Generative models for clustering: The next generation. In *Social Information Processing, Papers from the 2008 AAAI Spring Symposium, Technical Report SS-08-06, Stanford, California, USA, March 26-28, 2008*, pages 7–10, 2008. URL `http://www.aaai.org/Library/Symposia/Spring/2008/ss08-06-002.php`.

[52] Liaoruo Wang, Tiancheng Lou, Jie Tang, and John E Hopcroft. Detecting community kernels in large social networks. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 784–793. IEEE, 2011.

[53] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. Community detection in large-scale social networks. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 16–25. ACM, 2007.

[54] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[55] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

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

[57] Kathleen Carley. A theory of group stability. *American Sociological Review*, 56(3): 331–354, 1991.

[58] Kenneth Joseph, Geoffrey P Morgan, Michael K Martin, and Kathleen M Carley. On the coevolution of stereotype, culture, and social relationships: An agent-based model. *Social Science Computer Review*, 2013.

[59] Ted L. Huston and George Levinger. Interpersonal attraction and relationships. *Annual Review of Psychology*, 29:115–156, 1978.

[60] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 591–600, 2010.

[61] E. Fersini, E. Messina, and F. Archetti. A probabilistic relational approach for web document clustering. *Information Processing and Management*, 46(2):117–130, 2010.

[62] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[63] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[64] Wangqun Lin, Xiangnan Kong, Philip S. Yu, Quanyuan Wu, Yan Jia, and Chuan Li. Community detection in incomplete information networks. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 341–350, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187883. URL `http://doi.acm.org/10.1145/2187836.2187883`.

[65] E. Jain and S.K. Jain. Using mahout for clustering similar twitter users: Performance evaluation of k-means and its comparison with fuzzy k-means. In *Computer and Communication Technology (ICCCT), 2014 International Conference on*, pages 29–33, Sept 2014. doi: 10.1109/ICCCT.2014.7001465.

[66] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

[67] Liang Huang, Ruixuan Li, Hong Chen, Xiwu Gu, Kunmei Wen, and Yuhua Li. Detecting network communities using regularized spectral clustering algorithm. *Artificial Intelligence Review*, 41(4):579–594, 2014.

[68] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classification using matrix factorization. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 487–494. ACM, 2007.

[69] Shi Yu, Bart De Moor, and Yves Moreau. Clustering by heterogeneous data fusion: framework and applications. In *NIPS workshop*, 2009.

[70] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On finding graph clusterings with maximum modularity. *WG*, 7:121–132, 2007.

[71] G-J Qi, Charu C Aggarwal, and Thomas Huang. Community detection with edge content in social media networks. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 534–545. IEEE, 2012.

[72] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. Efficient community detection in large networks using content and links. In *Proceedings of the 22nd international conference on world wide web*, pages 1089–1098. International World Wide Web Conferences Steering Committee, 2013.

[73] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. Combining link and content for community detection: A discriminative approach. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 927–936, 2009.

[74] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2:1–135, 2008.

[75] Samaneh Moghaddam and Martin Ester. Opinion digger: an unsupervised opinion miner from unstructured product reviews. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1825–1828. ACM, 2010.

[76] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. Arsa: a sentiment-aware model for predicting sales performance using blogs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 607–614. ACM, 2007.

[77] Yancheng Hong and Steven Skiena. The wisdom of bookies? sentiment analysis vs. the nfl point spread. In *Proceedings of the international conference on Weblogs and Social media (icWSm-2010)*. Citeseer, 2010.

[78] Sitaram Asur, Bernardo Huberman, et al. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE, 2010.

[79] Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A Smith. Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296. Association for Computational Linguistics, 2010.

[80] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[81] Roy Bar-Haim, Elad Dinur, Ronen Feldman, Moshe Fresko, and Guy Goldstein. Identifying and following expert investors in stock microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1310–1319. Association for Computational Linguistics, 2011.

[82] Wenbin Zhang and Steven Skiena. Trading strategies to exploit blog and news sentiment. In *ICWSM*, 2010.

[83] Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10:178–185, 2010.

[84] Bi Chen, Leilei Zhu, Daniel Kifer, and Dongwon Lee. What is an opinion about? exploring political standpoints using opinion scoring model. In *AAAI*. Citeseer, 2010.

[85] Tae Yano and Noah A Smith. What's worthy of comment? content and comment volume in political blogs. In *ICWSM*, 2010.

[86] Junichi Tatemura. Virtual reviewers for collaborative exploration of movie reviews. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 272–275. ACM, 2000.

[87] Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter. Phoaks: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[88] Ellen Spertus. Smokey: Automatic recognition of hostile messages. In *AAAI/IAAI*, pages 1058–1065, 1997.

[89] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[90] Thorsten Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.

[91] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418, 2004.

[92] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631. ACM, 2005.

[93] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[94] Sanjiv R Das and Mike Y Chen. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388, 2007.

[95] Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL student research workshop*, pages 43–48. Association for Computational Linguistics, 2005.

[96] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12, 2009.

[97] Wojciech Gryc and Karo Moilanen. Leveraging textual sentiment analysis with social network modelling. *From Text to Political Positions: Text analysis across disciplines*, 55: 47, 2014.

[98] Mohammadreza Shams, Azadeh Shakery, and Heshaam Faili. A non-parametric lda-based induction method for sentiment analysis. In *Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on*, pages 216–221. IEEE, 2012.

[99] Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 537–546. ACM, 2013.

[100] Laura M Smith, Linhong Zhu, Kristina Lerman, and Zornitsa Kozareva. The role of social media in the discussion of controversial topics. In *Social Computing (SocialCom), 2013 International Conference on*, pages 236–243. IEEE, 2013.

[101] Hongbo Deng, Jiawei Han, Heng Ji, Hao Li, Yue Lu, and Hongning Wang. Exploring and inferring user-user pseudo-friendship for sentiment analysis with heterogeneous networks. *SDM*, pages 378—-386, 2013.

[102] Linhong Zhu, Aram Galstyan, James Cheng, and Kristina Lerman. Tripartite graph clustering for dynamic sentiment analysis on social media. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1531–1542. ACM, 2014.

[103] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proc. of the First Workshop on Unsupervised Learning in NLP*, 2011.

[104] Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. User-level sentiment analysis incorporating social networks. In *Proc. of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1397–1405, 2011.

[105] Federico Alberto Pozzi, Elisabetta Fersini, and Enza Messina. Bayesian model averaging and model selection for polarity classification. In *Proc. of the 18th International Conference on Application of Natural Language to Information Systems*, LNCS, pages 189–200, 2013.

[106] M. Wick, K. Rohanimanesh, A. Culotta, and A. McCallum. Samplerank: Learning preferences from atomic gradients. In *NIPS Workshop on Advances in Ranking*, 2009.

[107] T. G. Dietterich. Ensemble learning. In *The Handbook of Brain Theory and Neural Networks*, pages 405–508. Mit Pr, 2002.

[108] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

[109] Hossam Sharara, Lise Getoor, and Myra Norton. Active surveying: A probabilistic approach for identifying key opinion leaders. In *IJCAI*, pages 1485–1490, 2011.

[110] E. Fersini and E. Messina. Web page classification through probabilistic relational models. *International Journal of Pattern Recognition and Artificial Intelligence*, 2013.

[111] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. Learning latent representations of nodes for classifying in heterogeneous social networks. In *WSDM*, pages 373–382, 2014.

[112] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[113] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Acknowledgements

Foremost, I would like to express my most sincere gratitude to my advisor Prof. Enza Messina for the continuous support of my study and research through all these years, since when I was a bachelor student until now. Without her guidance and patience in discussing ideas and proof-reading this thesis, I would not have been able to improve. Thank you for pushing me beyond my limits, leading me to be a better researcher.

Besides my advisor, I would really like to thank Elisabetta Fersini. She has been not only a colleague, but a friend and a mentor to me. Thanks for all the help, the stimulating discussions and the sleepless nights working together before deadlines.

I also want to thank my colleagues Federico Pozzi, Debora Nozza, Valentina Beretta, Lorenza Manenti, Flavio Cecchini and Pikakshi Manchanda, with whom I spent these three important years.

Life in the lab wouldn't have been half as fun without you all.

I also would like to thank my parents and to my brother. Their constant support and help is the reason I can be here today.

Last, but surely not least, a word for Barbara: thank you for being with me every day of this long journey, whether we were close or far away. I love you =)