UNIVERSITA' DELLA CALABRIA

Dipartimento di **Ingegneria Informatica, Modellistica, Elettronica e Sistemistica**

Dottorato di Ricerca in

Information and Communication Technologies

CICLO

XXXI

USER BEHAVIORAL PROBLEMS IN COMPLEX SOCIAL NETWORKS

Settore Scientifico Disciplinare ING-INF/05

Coordinatore: Prof. Felice Crupi

Firma ______________________________

Supervisore/Tutor: Prof. Andrea Tagarelli

Firma______________________

Dottorando: Dott. Diego Perna

Firma ______________________

UNIVERSITÀ DELLA CALABRIA

Abstract

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES)

Doctor of Philosophy

User behavioral problems in complex social networks

by Diego PERNA

Over the past two decades, we witnessed the advent and the rapid growth of numerous social networking platforms. Their pervasive diffusion dramatically changed the way we communicate and socialize with each other. They introduce new paradigms and impose new constraints within their scope. On the other hand, online social networks (OSNs) provide scientists an unprecedented opportunity to observe, in a controlled way, human behaviors. The goal of the research project described in this thesis is to design and develop tools in the context of network science and machine learning, to analyze, characterize and ultimately describe user behaviors in OSNs.

After a brief review of network-science centrality measures and ranking algorithms, we examine the role of trust in OSNs, by proposing a new inference method for controversial situations. Afterward, we delve into social boundary spanning theory and define a ranking algorithm to rank and consequently identify users characterized by alternate behavior across OSNs. The second part of this thesis deals with machine-learning-based approaches to solve problems of learning a ranking function to identify lurkers and bots in OSNs. In the last part of this thesis, we discuss methods and techniques on how to learn a new representational space of entities in a multilayer social network.

"*Behavior analysis is not merely the sum of its basic and applied research and conceptual programs. It is their interrelationship, wherein each branch draws strength and integrity from the others. With the unity of behavior analysis clarified, the whole of behavior analysis emerges as greater than the sum of its parts.*"

Edward K. Morris

Acknowledgments

First and foremost I wish to express my sincere gratitude to my advisor, Prof. Andrea Tagarelli, for his distinguished continuous support, guidance and motivation through the rough road of the Ph.D. My sincere gratitude for each moment spent on this project, for believing in this amazing experience, for allowing me to grow as a person and as a research scientist during these years. Also, I would like to thank him for the brilliant comments and suggestions, but also for the hard questions which encouraged me to widen my research from various perspectives. Without him, it would not be the same. Thank you very much for believing in my research ambitions, thank you for all the inspiring conversations that raised in me the motivation to pursue working every time I was almost giving it up.

Last but not least, I am particularly thankful to my family and my partner for never stopping motivating me, to infuse great optimism and energy especially in the darkest days.

Finally, I am infinitely thankful for my determination, courage, and ambition. For the endless effort, dedication, and love to learn and discover.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"*In an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it.*"

Herbert Alexander Simon

1.1 Behaviorism in the modern era

Over the past two decades, we witnessed the advent and the rapid growth of numerous social networking platforms. Their extensive diffusion across the globe and in our lives dramatically changed the way we communicate and socialize with each other. These platforms introduced new paradigms and imposed new constraints within their boundaries. Conversely, online social networks (OSNs) provide to scientist an unprecedented opportunity to observe, in a controlled way, human behaviors.

1.1.1 Behaviorism

Behaviorism focuses on observable behavior as a means of studying the human psyche. The primary principle of behaviorism is that psychology should concern itself with the observable behavior of people and animals, rather than intangible events that take place in their minds. While the behaviorists criticized the mentalists for their inability to demonstrate empirical evidence to support their claims, the behaviorist school of thought claims that behaviors can be described scientifically without requiring either to internal physiological events or to hypothetical constructs such as thoughts and beliefs, making behavior a more productive area of focus for understanding human or animal psychology.

During the first half of the twentieth century, John B. Watson devised methodological behaviorism, which rejected introspective methods and sought to understand behavior by only measuring observable behaviors and events. The main contributors to behaviorist psychology were Ivan Pavlov, who investigated *classical conditioning* [1], Edward Lee Thorndike, who introduced the concept of *reinforcement* [2], [3] and was the first to apply psychological principles to learning; John B. Watson, who rejected introspective methods and sought to restrict psychology to experimental methods [4], [5]; and Burrhus Frederic Skinner, who conducted research on *operant conditioning* [6], [7].

Pavlov's findings support the idea that we develop responses to certain stimuli that are not naturally occurring. For instance, when we touch a hot stove, our reflex pulls our hand back. We do this instinctively, with no learning involved. This reflex is merely a survival instinct. Pavlov discovered that we make associations that cause us to generalize our response to one stimulus onto a neutral stimulus it is paired with. Many of our behaviors today are shaped by the pairing of stimuli. The smell of cologne, the sound of a certain song, or the occurrence of a specific day of the year can trigger distinct memories, emotions, and associations. When we make these types of associations, we are experiencing *classical conditioning* [1].

Operant conditioning is another type of learning that refers to how an organism operates on the environment or how it responds to what is presented to it in the environment. The stimuli at the basis of operant conditioning are reinforcement and punishment. Reinforcement means to strengthen, and is used in psychology to refer to any stimulus, both positive (e.g., giving a treat) or negative (e.g., studying to avoid getting a bad grade), which strengthens or increases the probability of a specific response. Whereas punishment aims to weaken a behavior by positive (e.g., spanking a child) or negative stimulus (e.g., telling the child to go to his room) [6], [7].

Among the several types of stimuli at the basis of operant conditioning, positive reinforcement has been found to be the most powerful. Adding a positive reinforcement to increase a response not only works better but allows both parties to focus on the positive aspects of the situation. Punishment, when applied immediately following the negative behavior, can be effective, but results in extinction when it is not applied consistently. Punishment can also invoke other negative responses such as anger and resentment.

John B. Watson promoted a change in psychology through his behaviorist approach and conducted research on animal behavior, child rearing, and advertising while gaining notoriety for the controversial "Little Albert" experiment. This experiment set out to show how the recently discovered principles of classical conditioning could be applied to condition fear into Little Albert, an 11-month-old boy.

Burrhus Frederic Skinner called his particular brand of behaviorism *radical behaviorism*. Radical behaviorism is the philosophy of the science of behavior. It seeks to understand behavior as a function of environmental histories of reinforcing consequences. This applied behaviorism does not accept private events such as thinking, perceptions, and unobservable emotions in a causal account of an organism's behavior.

Skinner invented the operant conditioning chamber, popularly referred to as the Skinner box (Figure 1.1), used to measure responses of organisms and their orderly interactions with the environment. The box had a lever and a food tray, and a hungry animal inside the box could get food delivered to the tray by pressing the lever. Skinner observed that when an animal was first put into the box, it would wander around, sniffing and exploring, and would usually press the bar by accident, at which point a food pellet would drop into the tray.

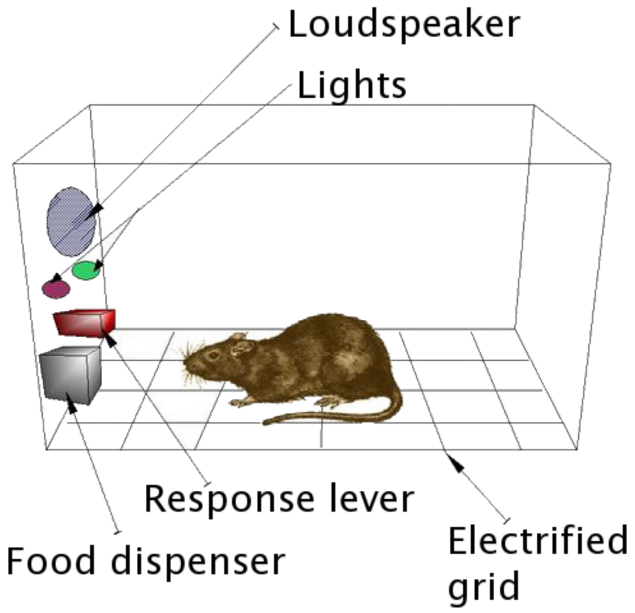


FIGURE 1.1: Illustration of the Skinner Box

After that happened, the rate of bar pressing would increase dramatically and remain high until the animal was no longer hungry.

Negative reinforcement was also exemplified by Skinner placing animals into an electrified chamber that delivered unpleasant shocks. Levers to cut the power were placed inside these boxes. By running a current through the box, Skinner noticed that the animals, after accidentally pressing the lever in a frantic bid to escape, quickly learned the effects of the lever and consequently used this knowledge to stop the currents both during and prior to electrical shock. These two learned responses are known as escape learning and avoidance learning [6]. The Skinner box led to the principle of reinforcement, which is the probability of something occurring based on the consequences of a behavior.

1.1.2 The rise of online social networks

Ahead of its time, in 1997 the website *Six Degrees*, which is said to be the first ever online social network site, was created. Two years later it was purchased for only 125 million dollars, and after another two years, *Six degrees* was shut down. The origin of OSNs that are still in use today can be traced back to the early 2000s, when OSNs such as *Linkedin* (2002), *Friendster* (2002), *Myspace* (2003), and finally *Facebook* (2004), *Reddit* (2005), *Youtube* (2005) and *Twitter* (2006) were founded and revealed to the public.

Generally, a social network can be defined as a web-based service that allows its users to i) construct a public or semi-public profile within a bounded system, ii) articulate a list of other users with whom they share a connection, and iii) view and traverse their list of connections and those made by others within the system. The nomenclature and name of these connections may vary from platform to platform. Depending on the social network, members may be able to contact any other member, or in other cases, members can contact anyone they have a connection to. Some services require members to have a preexisting connection to contact other members. While social networking has gone on almost as long as societies themselves have existed, the unparalleled potential of the World Wide Web to facilitate such connections has led to an exponential and ongoing expansion of that phenomenon.

Other than personal, social networks can be also used by business purposes, such as increase brand recognition and loyalty by making the company more accessible to new customers and more recognizable for existing customers, or help promote a brand's voice and content by spreading information about its products or services. In the end, social networks have proved to be an essential tool for businesses, particularly to help companies to find and retain their share of the market and their customers.

The vast majority of OSNs cannot be considered to be generalist platforms since they are limited to a specific niche or purpose (e.g., photos, work-related, video, etc.). Even Facebook, which is now considered by most the ultimate tool for social interactions, at its start was not more than a dating site, and later a "face book" (i.e., a student directory featuring photos and basic information). Therefore, with the exception of today's Facebook platform, each social network tries to leverage the need for social interaction in a distinctive way. However, the reasons that push a user to join a social network seems to be driven by the trend of the moment rather than a more significant added value, or public utility (see Figure 1.2).

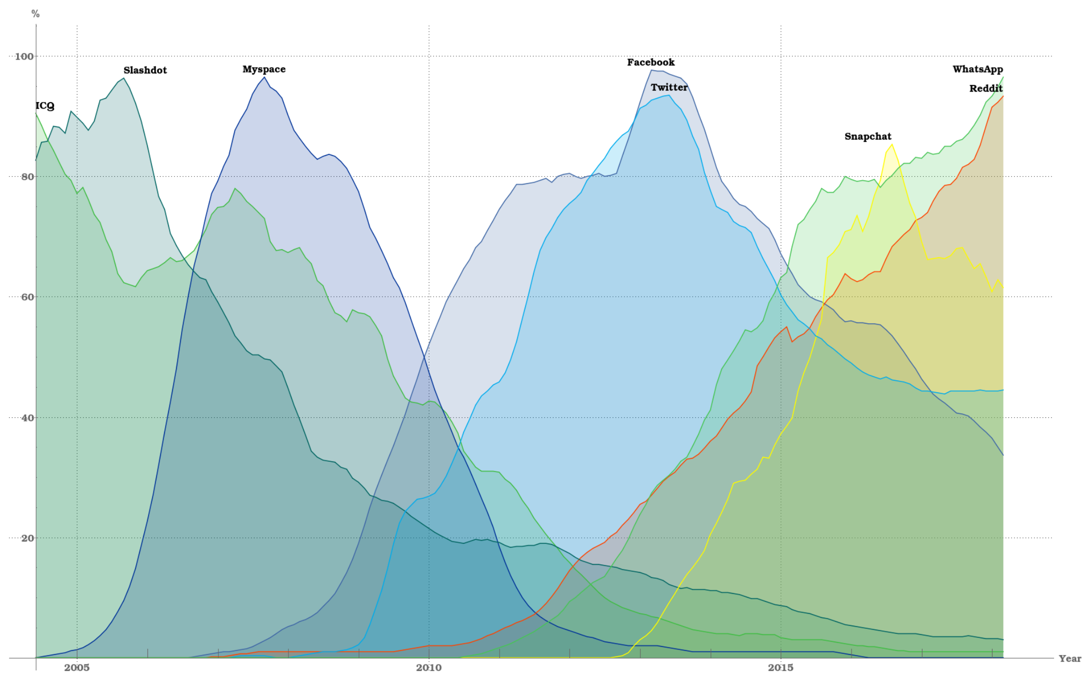


FIGURE 1.2: Relative search interest smoothed by a six-month average (Data from Google Trends)

As often stated, the alleged final goal of a social network is to easily connect people and enable new fast ways to share information and experiences. However, from a less naive and more pragmatic perspective, user profiling and hence marketing remain the fundamentals of their business models. In the end, a social network platform is merely a tool that enables the tool-owner to extract information from the actual product, the user. Businesses that base their revenue on such models have been defined as *attention merchants*, platforms that evaluate their success on measures like *time spent on the platform*, *number of access per day*, and so on, are characterized by a single goal, that is profit.

Furthermore, the monopoly that Facebook created and its continuous effort to provide a complete suite of tools limits the needs and reasons for which its users could feel the need to leave the platform. This could result in limiting the online experience to one single platform, which increases the chances to be confined in an *echo chamber*. An *echo chamber* (also known as filter bubble), is commonly defined as a metaphorical description of a situation in which beliefs are amplified or reinforced by communication and repetition inside a closed system, represented, in this case, by friends and relatives present on the platform. Other examples of *echo chamber* are those caused by the excessive use of personalization algorithms, which generates personalized e-comfort zones (i.e., feedback loops in which biases are reinforced rather than dismantled or simply confronted) for each one of us. Being confined in an *echo chamber* may increase political and social polarization and extremism, and limit the opportunities of changing idea or even thinking about a personal belief.

Conversely, the massive pervasiveness of these platforms gives to scientists the unprecedented opportunity of analyzing human behavior in a controlled environment and in a quantitative way. Similarly to a Skinner box1.1, a social network platform can be considered a digital box, in which human behavior is constantly monitored and rewards are in the form of instant gratification (e.g., notifications).

1.2 Objectives and organization of this thesis

In the fast-growing research area of social network analysis, by leveraging ranking and machine-learned-based methods, we focus our attention to the identification of users characterized by a specific type of behavior (i.e., automated, lurking, etc.), and also to users that show multiple, opposite behaviors in more than one social media platforms, as well as learning new representation of graph data in order to enable the application of a vast and variegated type of methods.

The aim of this research project is the development of method and algorithms that can have a high potential impact on the users of one or more social networks. For instance, methods able to identify lurkers can be used to put in practice policies of user engagement. Conversely, detecting users who act as a bridge between multiple social networks can help to identify real content creators by unmasking plagiarism behaviors.

In the following chapters, we firstly describe well-studied measures used in the context of network science to characterize user behavior. Then, we devoted the third chapter to the social boundary spanning theory, with the goal of identifying users which share and transfer their knowledge across the borders of a social network, we describe our approach to alternate behaviors ranking in multilayer social networks. The fourth chapter contains the description of two applications of the supervised learning framework Learning-to-rank (LTR) in the context of OSNs. More in detail, through leveraging state-of-the-art LTR algorithms, we aim to learn a ranking function to identify and rank users according to their bot or lurking status. In the last chapter of this dissertation, in the context of supervised learning, we try to advanced research on the network embedding problem, by developing and testing several new methods designed to be able to deal with multilayer networks.

Chapter 2

User behavior in network science

In the first part of this chapter, we describe well-known centrality measures and ranking algorithms in order to introduce, by a more technical point of view, the following chapters. In the second part of the chapter, we focus our attention on the problem of trust inference in controversial situations.

2.1 Ranking and centrality measures

Hyperlinks provide a valuable source of information for web search. In fact, the analysis of the hyperlink structure of the Web, commonly called *link analysis*, has been successfully used for improving both the retrieving of web documents (i.e., which webpages to crawl) and the scoring of web documents according to some notion of "quality" (i.e., how to rank webpages). The latter is in general meant either as the relevance of the documents with respect to a user query or as some query-independent, intrinsic notion of *centrality*. In network theory and analysis, the identification of the "most central" nodes in the network (e.g., documents in the web network, actors in a social network, etc.) represents a core task.

The term centrality commonly resembles that of importance or prominence of a vertex in a network, i.e., the status of being located in strategic locations within the network. However, there is no unique definition of centrality, as for instance one may postulate that a vertex is important if it is involved in many direct interactions, or if it connects two large components (i.e., if it acts as a bridge), or if it allows for quick transfer of the information also by accounting for indirect paths that involve intermediaries. Consequently, there are only very few desiderata for a centrality measure, which can be expressed as follows:

- A vertex centrality is a function that assigns a real-valued score to each vertex in a network. The higher the score, the more important or prominent the vertex is for the network.
- If two graphs G_1, G_2 are isomorphic and $m(v)$ denotes the mapping function from a node v in G_1 to some node v' in G_2, then the centrality of v in G_1 needs to be the same as the centrality of $m(v) = v'$ in G_2. In other terms, the centrality of a vertex is only depending on the structure of the network.

The term centrality is originally designed for undirected networks. In the case of directional relations, which imply directed networks, the term centrality is still used and refers to the "choices made", or out-degrees of vertices, while the term *prestige* is introduced to examine the "choices received", or in-degrees of vertices [8]. Moreover, the vertex centrality scores can be aggregated over all vertices in order to obtain a single, network-level measure of centrality, or alternatively *centralization*, which aims to provide a clue on the variability of the individual vertex centrality scores with respect to a given centrality notion. In the following, we will overview the most prominent measures of centrality and prestige, and their definitions for undirected and directed networks. Particularly, we will focus on two well-known methods, namely PageRank and Hubs & Authorities, which have been widely applied to web search contexts.

Through the rest of this section and in the subsequent sections of this chapter, we will denote with $G = (V, E)$ a *network graph*, which consists of two sets V and E, such that $V \neq \emptyset$ and E is a set of pairs of elements of V. If the pairs in E are ordered the graph is said *directed*, otherwise is *undirected.* The elements in V are the vertices (or nodes) of G, while the elements in E are the edges (or links) of G.

2.1.1 Basic measures

Vertex-level centrality. The most intuitive measure of centrality for any vertex $v \in V$ is the degree centrality, which is defined as the number of edges incident with v, or degree of v:

$$c_D(v) = deg(v). \tag{2.1}$$

Being dependent only on adjacent neighbors of a vertex, this type of centrality focuses on the most "visible" vertices in the network, as those that act as major point of relational information; by contrast, vertices with low degrees are peripheral in the network. Moreover, the degree centrality depends on the graph size: indeed, since the highest degree for a network (without loops) is $|V|-1$, the relative degree centrality is:

$$\widehat{c_D}(v) = \frac{c_D(v)}{|V|-1} = \frac{deg(v)}{|V|-1}. \tag{2.2}$$

The above measure is independent on the graph size, and hence it can be compared across networks of different sizes.

The definitions of both absolute and relative degree centrality and degree prestige of a vertex in a directed network are straightforward. In that case, the degree and the set of neighbors have two components: we denote with $B_i = \{v_j | (v_j, v_i) \in E\}$ the set of *in-neighbors* (or "backward" vertices) of v_i, and with $R_i = \{v_j | (v_i, v_j) \in E\}$ the set of *out-neighbors* (or "reference" vertices) of v_i. The sizes of sets B_i and R_i are the *in-degree* and the *out-degree* of v_i, denoted as $in(v_i)$ and $out(v_i)$, respectively.

Note also that the degree centrality is also the starting point for various other measures; for instance, the span of a vertex, which is defined as the fraction of links in the network that involves the vertex or its neighbors, and the ego density, which is the ratio of the degree of the vertex to the theoretical maximum number of links in the network.

Unlike degree centrality, closeness centrality takes also into account indirect links between vertices in the network, in order to score higher those vertices that can quickly interact with all others because of their lower distance to the other vertices [9]:

$$c_C(v) = \frac{1}{\sum_{u \in V} d(v,u)}, \tag{2.3}$$

where $d(v,u)$ denotes the graph theoretic, or geodesic, distance (i.e., length of shortest path) between vertices v, u. Since a vertex has the highest closeness if it has all the other vertices as neighbors, the relative closeness centrality is defined as:

$$\widehat{c_C}(v) = (|V|-1)c_C(v) = \frac{|V|-1}{\sum_{u \in V} d(v,u)}. \tag{2.4}$$

In the case of directed networks, closeness centrality and prestige can be computed according to outgoing links (i.e., how many hops are needed to reach all other vertices from the selected one) or incoming links (i.e., how many hops are needed to reach the selected vertex from all other vertices), respectively. Note that the closeness centrality is only meaningful for a connected network—in fact, the geodesics to a vertex that is not reachable from any other vertex are infinitely long. One remedy to this issue is to define closeness by focusing on distances from the vertex v to only the vertices that are in the *influence range* of v (i.e., the set of vertices reachable from v) [8].

Besides (shortest) distance, another important property refers to the ability of a vertex to have control over the flow of information in the network. The idea behind betweenness centrality is to compute the centrality of a vertex v as the fraction of the shortest paths between all pairs of vertices that pass through v [10]:

$$c_B(v) = \sum_{u,z \in V, u \neq v, z \neq v} \frac{m_{u,z}(v)}{m_{u,z}(V)}, \tag{2.5}$$

where $m_{u,z}(v)$ is the number of shortest paths between u and z and passing through v, and $m_{u,z}(V)$ is the total number of shortest paths between u and z. This centrality is minimum (zero) when the vertex does not fall on any geodesic, and maximum when the vertex falls on all geodesics, which is equal to $(|V|-1)(|V|-2)/2$. Analogously to the other centrality measures, it's recommended to standardize the betweenness to obtain a relative betweenness centrality:

$$\widehat{c_B}(v) = \frac{2c_B(v)}{(|V|-1)(|V|-2)}, \tag{2.6}$$

which should be divided by 2 for directed networks. Note that, unlike closeness, betweenness can be computed even if the network is disconnected.

It should be noted that the computation of betweenness centrality is the most resource-intensive among the above-discussed measures: while standard algorithms based on Dijkstra's or breadth-first search methods require $\mathcal{O}(|V|^3)$ time and $\mathcal{O}(|V|^2)$ space, algorithms designed for large, sparse networks require $\mathcal{O}(|V|+|E|)$ space and $\mathcal{O}(|V||E|)$ and $\mathcal{O}(|V||E|+|V|^2\log|V|)$ time on unweighted and weighted networks, respectively [11]. A number of variants of betweenness centrality have also been investigated; for instance, in [12], an extension of betweenness to edges is obtained by replacing the term $m_{u,z}(v)$ in equation (2.5) by a term $m_{u,z}(e)$ calculating the number of shortest (u, z)-paths containing the edge e. An application of this version of edge betweenness is the clustering approach by Newman and Girvan [13], where edges of maximum betweenness are removed iteratively to decompose a graph into relatively dense subgraphs.

Besides computational complexity issue, a criticism to betweenness centrality is that it assumes that all geodesics are equally likely when calculating if a vertex falls on a particular geodesic. However, a vertex with large in-degree is more likely to be found on a geodesic. Moreover, in many contexts, there may be equally likely that other paths than geodesics are chosen for the information propagation, therefore the paths between vertices should be weighted depending on their length. The index defined by Stephenson and Zelen [14] builds upon the above generalization, by accounting for all paths, including geodesics, and assigning them with weights, which are computed as the inverse of the path lengths (geodesics are given unitary weights). The same researchers also developed an *information centrality* measure, which focuses on the information contained in all paths that originate and end at a specific vertex. The information of a vertex is a function of all the information for paths flowing out from the vertex, which in turn is inversely related to the variance in the transmission of a signal from a vertex to another. Formally, given an undirected network, possibly with weighted edges, a $|V|\times|V|$ matrix $\mathbf{X}$ is computed as follows: the i-th diagonal entry is equal to 1 plus the sum of weights for all incoming links to vertex v_i, and the (i, j)-th off-diagonal entry is equal to 1, if v_i and v_j are not adjacent, otherwise is equal to 1 minus the weight of the edge between v_i and v_j. For any vertex v_i, the information centrality is defined as:

$$c_I(v_i) = \frac{1}{y_{ii} + \frac{1}{|V|}(\sum_{v_j \in V} y_{jj} - 2\sum_{v_j \in V} y_{ij})}, \tag{2.7}$$

where $\{y_{ij}\}$ are the entries of the matrix $\mathbf{Y} = \mathbf{X}^{-1}$. Since function c_I is only lower bounded (the minimum is zero), the relative information centrality for any vertex v_i is obtained by dividing $c_I(v_i)$ by the sum of the c_I values for all vertices.

Network-level centrality. A basic network-level measure of degree centrality is simply derived by taking into account the (standardized) average of the degrees:

$$\frac{\sum_{v \in V} c_D(v)}{|V||V-1|} = \frac{\sum_{v \in V} \widehat{c_D}(v)}{|V|}, \tag{2.8}$$

which is exactly the density of the network.

Focusing on a global notion of closeness, a simplification of this type of centrality stems from the graph-theoretic *center* of a network. This is in turn based on the notion of *eccentricity* of a vertex v, i.e., the distance to a vertex farthest to v. Specifically, the Jordan center of a network is the subset of vertices that have the lowest maximum distance to all other vertices, i.e., the subset of vertices within the *radius* of a network.

A unifying view of network-level centrality is based on the notion of network centralization, which expresses how the vertices in the network graph G differ in centrality [15]:

$$\mathsf{C}(G) = \frac{\sum_{v \in V} \mathsf{maxC} - \mathsf{C}(v)}{\max \sum_{v \in V} \mathsf{maxC} - \mathsf{C}(v)}, \tag{2.9}$$

where $\mathsf{C}(\cdot)$ is a function that expresses a selected measure of relative centrality, and maxC is the maximum value of relative centrality over all vertices in the network graph. Therefore, centralization is lower when more vertices have similar centrality, and higher when one or few vertices dominate the other vertices; as extreme cases, a star network and a regular (e.g., cycle) network have centralization equal to 1 and 0, respectively. According to the type of centrality considered, the network centralization assumes different form. More specifically, considering the degree, closeness, and betweenness centralities, the denominator in equation (2.9) is equal to $(n-1)(n-2)$, $(n-1)(n-2)/(2n-3)$, and $(n-1)$, respectively.

2.1.2 Eigenvector centrality and prestige

None of the previously discussed measures reflects the importance of the vertices that interact with the target vertex when looking at (in)degree or distance aspects. Intuitively, if the influence range of a vertex involves many prestigious vertices, then the prestige of that vertex should also be high; conversely, the prestige should be low if the involved vertices are peripheral. Generally speaking, a vertex's prestige should depend on the prestige of the vertices that point to it, and their prestige should also depend on the vertices that point to them, and so "ad infinitum" [16]. It should be noted that the literature usually refers to the above property as *status*, or *rank*.

The idea behind status or rank prestige by Seeley, denoted by function $r(\cdot)$, can be formalized as follows:

$$r(v) = \sum_{u \in V} A(u,v) r(u), \tag{2.10}$$

where $A(u,v)$ is equal to 1 if u points to v (i.e., u is an in-neighbor of v), and 0 otherwise. Equation (2.10) corresponds to a set of $|V|$ linear equations (with $|V|$ unknowns) which can be rewritten as:

$$\mathbf{r} = \mathbf{A}^{\mathrm{T}}\mathbf{r}, \tag{2.11}$$

where $\mathbf{r}$ is a vector of size $|V|$ storing all rank scores, and $\mathbf{A}$ is the adjacency matrix. Or, rearranging terms, we obtain $(\mathbf{I} - \mathbf{A}^{\mathrm{T}})\mathbf{r} = \mathbf{0}$, where $\mathbf{I}$ is the identity matrix of size $|V|$.

Katz [17] first recommended to manipulate the matrix $\mathbf{A}$ by constraining every row in $\mathbf{A}$ to have sum equal to 1, thus enabling equation (2.11) to have finite solution. In effect, equation (2.11) is a characteristic equation used to find the eigensystem of a matrix, in which $\mathbf{r}$ is an eigenvector of $\mathbf{A}^{\mathrm{T}}$ corresponding to an eigenvalue of 1. In general, equation (2.11) has no non-zero solution unless $\mathbf{A}^{\mathrm{T}}$ has an eigenvalue of 1.

A generalization of equation (2.11) was suggested by Bonacich [18], where the assumption is that the status of each vertex is proportional (but not necessarily equal) to the weighted sum of the vertices to whom it is connected. The result, known as eigenvector centrality, is expressed as follows:

$$\lambda \mathbf{r} = \mathbf{A}^{\mathrm{T}} \mathbf{r}. \tag{2.12}$$

Note that the above equation has $|V|$ solutions corresponding to $|V|$ values of λ. Therefore, the general solution can be expressed as a matrix equation:

$$\lambda \mathbf{R} = \mathbf{A}^{\mathrm{T}} \mathbf{R}, \tag{2.13}$$

where $\mathbf{R}$ is a $|V| \times |V|$ matrix whose columns are the eigenvectors of $\mathbf{A}^{\mathrm{T}}$ and λ is a diagonal matrix of eigenvalues.

Katz [17] also proposed to introduce in equation (2.11) an "attenuation parameter" $\alpha \in (0, 1)$ to adjust for the lower importance of longer paths between vertices. The result, known as Katz centrality, measures the prestige as a weighted sum of all the powers of the adjacency matrix:

$$\mathbf{r} = \sum_{i=1}^{\infty} \alpha^i \mathbf{A}^{\mathrm{T}i} \mathbf{r}. \tag{2.14}$$

When α is small, Katz centrality tends to probe only the local structure of the network; as α grows, more distant vertices contribute to the centrality of a given vertex. Note also that the infinite sum in the above equation converges to $\mathbf{r} = [(\mathbf{I} - \alpha \mathbf{A}^{\mathrm{T}})^{-1} - \mathbf{I}]\mathbf{1}$ as long as $|\alpha| < 1/\lambda_1$, where λ_1 is the first eigenvalue of $\mathbf{A}^{\mathrm{T}}$.

All the above measures may fail in producing meaningful results for networks that contain vertices with null in-degree: in fact, according to the assumption that a vertex has no status if it does not receive choices from other vertices, vertices with null in-degree do not contribute to the status of any other vertex. A solution to this problem is to allow every vertex some status that is independent of its connections to other vertices. The Bonacich & Lloyd centrality [19], probably better known as alpha-centrality, is defined as:

$$\mathbf{r} = \alpha \mathbf{A}^{\mathrm{T}} \mathbf{r} + \mathbf{e}, \tag{2.15}$$

where $\mathbf{e}$ is a $|V|$-dimensional vector reflecting *exogenous* source of information or status, which is assumed to a vector of ones. Moreover, parameter α here reflects the relative importance of endogenous versus exogenous factors in determining the vertex prestige. The solution of equation (2.15) is:

$$\mathbf{r} = (\mathbf{I} - \alpha \mathbf{A}^{\mathrm{T}})^{-1} \mathbf{e}. \tag{2.16}$$

It can easily be proved that equation (2.16) and equation (2.14) differ only by a constant (i.e., one) [19].

2.1.3 PageRank

In [20], Brin and Page presented *PageRank*, the Google's patented ranking algorithm. There are four key ideas behind PageRank. The first two are also shared with the previously discussed eigenvector centrality methods, that is: a page is prestigious if it is chosen (pointed to) by other pages, and the prestige of a page is determined by summing the prestige values of all pages that point to that page. The third idea is that the prestige of a page is propagated to its out-neighbors as distributed proportionally. Let $\mathbf{W}$ be a $|V|\times|V|$ matrix such that columns refer to those vertices whose status is determined by the connections received from the row vertices:

$$W(i,j) = \begin{cases} 1/out(v_i) & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Note that $\mathbf{W} = \mathbf{D}_{\mathrm{out}}^{-1}\mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}_{\mathrm{out}}$ is a diagonal matrix storing the out-degrees of the vertices (i.e., $\mathbf{D}_{\mathrm{out}} = diag(\mathbf{A}\mathbf{1})$). Using matrix $\mathbf{W}$, the first three ideas underlying the PageRank can be expressed as $\mathbf{r} = \mathbf{W}^{\mathrm{T}}\mathbf{r}$, or equivalently, for every $v_i \in V$:

$$r(v_i) = \sum_{v_j \in B_i} \frac{r(v_j)}{out(v_j)}. \tag{2.18}$$

Therefore, vector $\mathbf{r}$ is the unique eigenvector of the matrix corresponding to eigenvalue 1. It should be noted that equation (2.18) is well-defined only if the graph is strongly connected (i.e., every vertex can be reached from any other vertex). Under this assumption, this equation has an interpretation based on random walks, called the *random surfer* model [20]. It can be shown that vector $\mathbf{r}$ is proportional to the stationary probability distribution of the random walk on the underlying graph. It should be remarked that, in contrast to PageRank, alpha-centrality does not have a natural interpretation in terms of probability distribution, i.e., the sum of the values in the alpha-centrality vector (cf. equation (2.15)) is not necessarily equal to 1.

However, the assumption of graph connectivity behind equation (2.18) needs to be relaxed for practical application of PageRank, since the Web and, in general, real-world networks are far from being strongly connected. It might be useful to recall here that the Web and many other directed networks have a structure which is characterized by five types of components (cf., e.g., [21]): (i) a large strongly connected component (SCC), (ii) an in-component, which contains vertices that can reach the SCC but are not reachable from the SCC, and an out-component, which contains vertices that are reachable from the SCC but cannot reach the SCC, (iii) in-tendrils and out-tendrils, which are vertices that are only connected to the out-component (via out-links) and vertices that are only connected to the in-component (via in-links), (iv) tubes, which are vertices reachable from the in-component and able to reach the out-component, but have neither in-links nor out-links with the SCC, and (v) isolated components, which contain vertices that are disconnected from each of the previous components. Most of these components violate the assumptions needed for the convergence of a Markov process. In particular, when a random surfer enters the out-component, she will eventually get stuck in it; as a result, vertices that are not in the out-component will receive a zero rank, i.e., one cannot distinguish the prestige of such vertices. More specifically, equation (2.18) needs to be modified to prevent anomalies that are caused by two types of structures: *rank sinks*, or "spider traps", and *rank leaks*, or "dead ends". The former are sets of vertices that have no links outwards, the latter are individual vertices with no out-links.

If leak vertices would be directly represented in matrix $\mathbf{W}$, then they would correspond to rows of zero, thus making $\mathbf{W}$ substochastic: as a result, by re-iterating equation (2.18) for a certain number k of times (i.e., by computing $\mathbf{W}^{\mathrm{T}k}\mathbf{r}$), then some or all of the entries in $\mathbf{r}$ will go to 0. To solve this issue, two approaches can be suggested: (i) modification of the network structure, and (ii) modification of the random surfer behavior. In the first case, leak vertices could be removed from the network so that they will receive zero rank; alternatively, leak vertices could be "virtually" linked back to their in-neighbors, or even to all other vertices. The result will be a row-stochastic matrix, that is, a matrix that is identical to $\mathbf{W}$ except that it will have the columns corresponding to leak vertices that sum to 1. If we denote with $\mathbf{d}$ a vector indexing the leak vertices (i.e., $d(i) = 1$ if v_i has no outlinks, and $d(i) = 0$ otherwise), this row-stochastic matrix $\mathbf{S}$ is defined as:

$$\mathbf{S} = \mathbf{W} + \mathbf{d}\mathbf{1}^{\mathrm{T}}/|V|. \tag{2.19}$$

However, equation (2.19) will not solve the problem of sinks. Therefore, Page and Brin [20] also proposed to modify the random surfer behavior by allowing for *teleportation*, i.e., the random surfer who gets stuck in a sink, or simply gets "bored" occasionally, she can move by randomly jumping to any other vertex in the network. This is the fourth idea behind the PageRank measure, which is implemented by a damping factor $\alpha \in (0, 1)$ that enables to weigh the mixture of random walk and random teleportation:

$$\mathbf{r} = \alpha\mathbf{S}^{\mathrm{T}}\mathbf{r} + (1 - \alpha)\mathbf{p}. \tag{2.20}$$

Above, vector $\mathbf{p}$, usually called *personalization vector*, is by default set to $\mathbf{1}/|V|$, but it can be any probability vector. Equation (2.20) can be rewritten as:

$$\mathbf{G} = \alpha\mathbf{S} + (1 - \alpha)\mathbf{E}, \tag{2.21}$$

where $\mathbf{E} = \mathbf{1}\mathbf{p}^{\mathrm{T}} = \mathbf{1}\mathbf{1}^{\mathrm{T}}/|V|$. The convex combination of $\mathbf{S}$ and $\mathbf{E}$ makes the resulting "Google matrix" $\mathbf{G}$ to be both *stochastic and irreducible*.[1] This is important to ensure (i) the existence and uniqueness of the PageRank vector as stationary probability distribution $\boldsymbol{\pi}$, and (ii) the convergence of the underlying Markov chain (at a certain iteration k, i.e., $\boldsymbol{\pi}^{(k+1)} = \mathbf{G}\boldsymbol{\pi}^{(k)}$) independently of the initialization of the rank vector.[2]

[1] A matrix is said *irreducible* if every vertex in its graph is reachable from every other vertex.

[2] Recall that the property of irreducibility of a matrix is related to those of primitivity and aperiodicity. A nonnegative, irreducible matrix is said *primitive* if it has only one eigenvalue on its spectral circle; a simple test by Frobenius states that a matrix $\mathbf{X}$ is primitive if and only

Computing PageRank. As previously indicated, the computation of PageRank requires to solve equation (2.21), which is equivalent to find the principal eigenvector of matrix $\mathbf{G}$. Therefore, similarly to other eigenvector centrality methods, the power iteration algorithm is commonly used. Starting from any random vector $\mathbf{r}^{(0)}$, it iterates through equation (2.20) until some termination criterion is met; typically, the power method is assumed to terminate when the residual (as measured by the difference of successive iterations) is below some predetermined threshold. Actually, as first observed by Haveliwala (cf. [23]), the ranking of the PageRank scores are more important than the scores themselves, that is, the power method can be iterated until ranking stability is achieved, thus leading to a significant saving of iterations on some datasets.

The power iteration method lends itself to efficient implementation thanks to the sparsity of real-world network graphs. Indeed, computing and storing matrix $\mathbf{S}$ (cf. equation (2.19)), and hence $\mathbf{G}$, is not required, since the power method can be rewritten as [23]:

$$\boldsymbol{\pi}^{\mathrm{T}(k+1)} = \boldsymbol{\pi}^{\mathrm{T}(k)}\mathbf{G} = \alpha\boldsymbol{\pi}^{\mathrm{T}(k)}\mathbf{W} + (\alpha\boldsymbol{\pi}^{\mathrm{T}(k)}\mathbf{d})\mathbf{1}^{\mathrm{T}}/|V| + (1-\alpha)\mathbf{p}^{\mathrm{T}}, \tag{2.22}$$

which indicates that only sparse vector/matrix multiplications are required. When implemented in this way, each step of the power iteration method requires $nonzero(\mathbf{W})$ operations, where $nonzero(\mathbf{W})$ is the number of nonzero entries in $\mathbf{W}$, which approximates to $\mathcal{O}(|V|)$.

Choosing the damping factor. The damping factor α is by default set to 0.85. This choice actually finds several explanations. One is intuitively based on the empirical observation that a web surfer is likely to navigate following 6 hyperlinks (before discontinuing this navigation chain and randomly jumping on another page), which corresponds to a probability $\alpha = 1 - (1/6) \approx 0.85$. In addition, there are also computational reasons. With the default value of 0.85, the power method is expected to converge in about 114 iterations for a termination tolerance threshold of 1.0E-8 [23]. Moreover, since the second largest eigenvalue of $\mathbf{G}$ is α [22], it can be shown that the asymptotic rate of convergence of the power method is $-\log_{10} 0.85 \approx 0.07$, which means that about 14 iterations are needed for each step of accuracy improvement (in terms of digits).

In general, higher values of α imply that the hyperlink structure is more accurately taken into account, however along with slower convergence and higher sensitivity issues. In fact, experiments with various settings of α have shown that there can be significant variation in rankings produced by different values of α, especially when α approaches 1; more precisely, significant variations are usually observed for mid-low ranks, while the top of the ranking is usually only slightly affected [23], [24].

Choosing the personalization vector. As previously discussed, the personalization vector $\mathbf{p}$ can be replaced with any vector whose non-negative

if $\mathbf{X}^k > 0$ for some $k > 0$, which is useful to determine whether the power method applied to $\mathbf{X}$ will converge [22]. An irreducible Markov chain with a primitive transition matrix is called an *aperiodic chain*.

components sum up to 1. This hence includes the possibility that the vertices in V might be differently considered when the random surfer restarts her chain by selecting a vertex v with probability $p(v)$, which is not necessarily uniform over all the vertices.

The teleportation probability $p(v)$ can be determined to be proportional to the score the vertex v obtains with respect to an external criterion of importance, or to the contribution that the vertex gives to a certain topological characteristic of the network. For instance, one may want to assign any vertex with a teleportation probability that is proportional to the in-degree of the vertex, i.e.,

$$p(v) = \frac{in(v)}{\sum_{u \in V} in(u)}.$$

The personalization vector can also be used to boost the PageRank score for a specific *subset of vertices that are relevant to a certain topic*, thus making the PageRank to be *topic-sensitive*.

2.1.4 Hubs and authorities

A different approach to the computation of vertex prestige is based on the notions of *hubs* and *authorities*. In a web search context, given a user query, authority pages are ones most likely to be relevant to the query, while hub pages act as indices of authority pages without being necessarily authorities themselves. These two types of webpages are related to each other by a mutual reinforcement mechanism: in fact, if a page is relevant to a query, one would expect that it will be pointed to by many other pages; moreover, pages pointing to a relevant page are likely to point as well to other relevant pages, thus inducing a kind of bipartite graph where pages that are relevant by content (authorities) are endorsed by special pages that are relevant because they contain hyperlinks to locate relevant contents (hubs)—although, it may be the case that a page is both an authority and a hub.

The above intuition is implemented by the Kleinberg's *HITS* (Hyperlink Induced Topic Search) algorithm [25], [26]. Like PageRank and other eigenvector centrality methods, HITS still handles an iterative computation of a fix-point involving eigenvector equations; however, it originally views the prestige of a page as a two-dimensional notion, thus resulting in two ranking scores for every vertex in the network. Also in contrast to PageRank, HITS produces ranking scores that are query-dependent. In fact, HITS assumes that hubs and authorities are identified and ranked for vertices that belong to a *query-focused sub-network*. This is usually formed by an initial set of randomly selected pages containing the query terms, which is expanded by also including the neighborhoods of those pages.

Let $\mathbf{a}$ and $\mathbf{h}$ be two vectors storing the authority and hub scores, respectively. The hub score of a vertex can be expressed as proportional to the sum of the authority scores of its out-neighbors; analogously, the authority score of a vertex can be expressed as proportional to the sum of the hub scores of its in-neighbors.

Formally, HITS equations are defined as:

$$\mathbf{a} = \mu \mathbf{A}^{\mathrm{T}} \mathbf{h} \tag{2.23}$$

$$\mathbf{h} = \lambda \mathbf{A} \mathbf{a}, \tag{2.24}$$

where μ, λ are two (unknown) scaling constants that are needed to avoid that the authority and hub scores will grow beyond bounds; in practice, $\mathbf{a}$ and $\mathbf{h}$ are normalized so that the largest value in each of the vectors equals 1 (or, alternatively, all values in each of the vectors sum up to 1). Therefore, HITS works as follows:

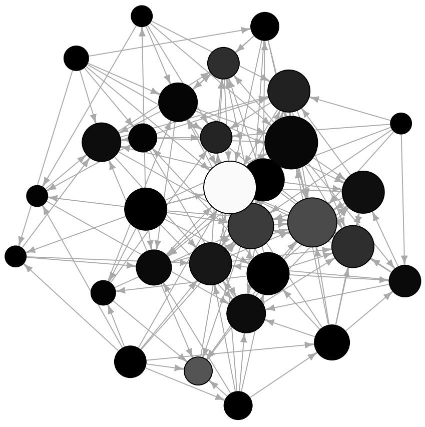
1. For every vertex in the expanded query-focused subnetwork, initialize hub and authority score (e.g., to 1).
2. Compute the following steps until convergence (i.e., a termination tolerance threshold is reached):
   (a) authority vector $\mathbf{a}$ using equation (2.23);
   (b) hub vector $\mathbf{h}$ using equation (2.24);
   (c) normalize $\mathbf{a}$ and $\mathbf{h}$.

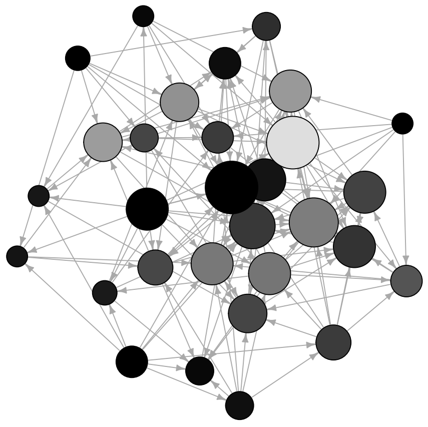
Note that, at the first iteration, $\mathbf{a}$ and $\mathbf{h}$ are none other than the vertex in-degrees and the out-degrees, respectively.

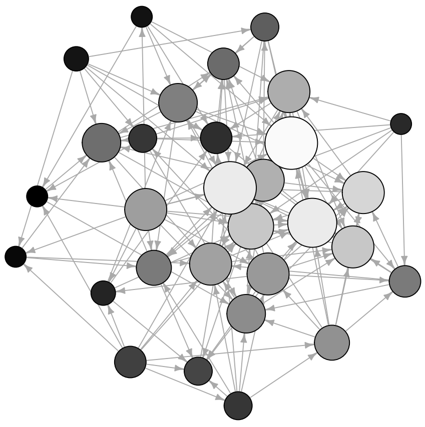
By substituting equation (2.23) and equation (2.24) in each other, hub and authority can in principle be computed independently of each other, through the computation of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ (for the hub vector) and $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ (for the authority vector). Note that, the (i, j)-th entry in matrix $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ corresponds to the number of pages jointly referred by pages i and j; analogously, the (i, j)-th entry in matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ corresponds to the number of pages that jointly point to pages i and j. However, both matrix products lead to matrices that are not as sparse, hence the only convenient way to compute $\mathbf{a}$ and $\mathbf{h}$ is iteratively in a mutual fashion as described above. In this regard, just as in the case of PageRank, the rate of convergence of HITS depends on the eigenvalue gap, and the ordering of hubs and authorities becomes stable with much fewer iterations than the actual scores.

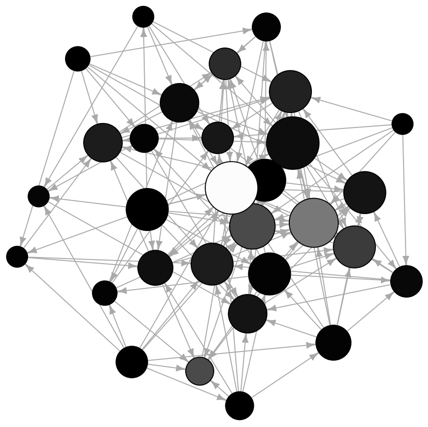
It should be noted that the assumption of identifying authorities by means of hubs might not hold in other information networks other than the Web; for instance, in citation networks, important authors typically acknowledge other important authors. This has somehow impacted on the probably less popularity of HITS with respect to PageRank—which, conversely, has been successfully applied to many other contexts, including citation and collaboration networks, lexical/semantic networks inferred from natural language texts, recommender systems, and social networks.

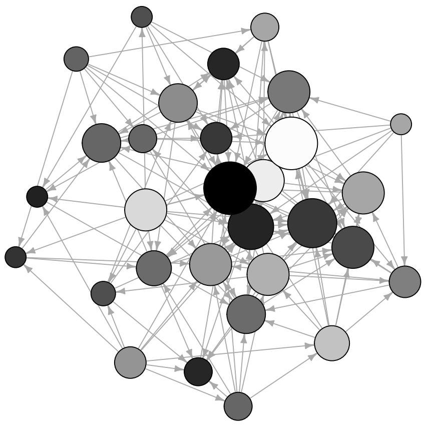
The TKC effect. Beyond limited applicability, HITS seems to suffer from two issues that are related to both the precision and coverage of the query search results. More precisely, while the coverage of search results directly affects the size of the subnetwork, the precision can significantly impact the *tightly knit communities* (TKC) effect, which occurs when relatively many pages are


(a) closeness


(b) betweenness


(c) eigenvector centrality


(d) PageRank


(e) Kleinberg's hub score

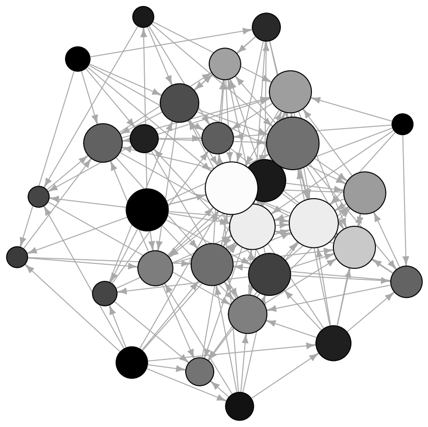

(f) Kleinberg's authority score

FIGURE 2.1: Comparison of ranking performance of different centrality methods on the same example graph. Node size is proportional to the node degree. Lighter gray-levels correspond to higher rank scores.

identified as authoritative via link analysis although they actually pertain to one aspect of the target topic; for instance, this is the case when hubs point both to actual relevant pages and to pages that are instead relevant to "related topics" [27]. The latter phenomenon is also called *topic drift*.

While the TKC effect can be attenuated by accounting for the analysis of contents and/or the anchor texts of the webpages (e.g., [28], [29]), other link analysis approaches have been developed to avoid overly favoring the authorities of tightly knit communities. Lempel and Morgan [27] propose the Stochastic Approach for Link Structure Analysis, dubbed *SALSA*. This is a variation of Kleinberg's algorithm: it constructs an expanded query-focused subnetwork in the same way as HITS, and likewise, it computes an authority and a hub score for each vertex in the neighborhood graph (and these scores can be viewed as the principal eigenvectors of two matrices). However, instead of using the straight adjacency matrix, SALSA weighs the entries according to their in and out-degrees. More precisely, the authority scores are determined by the stationary distribution of a two-step Markov chain through random walking over in-neighbors of a page and then random walking over out-neighbors of a page, while the hub scores are determined similarly with inverted order of the two steps in the Markov chain. Formally, the Markov chain for authority scores has transition probabilities:

$$p_a(i,j) = \sum_{v_q \in B_i \cap B_j} \frac{1}{in(v_i)} \frac{1}{out(v_k)} \tag{2.25}$$

and the Markov chain for hub scores has transition probabilities:

$$p_h(i,j) = \sum_{v_q \in R_i \cap R_j} \frac{1}{out(v_i)} \frac{1}{in(v_k)}. \tag{2.26}$$

Lempel and Morgan proved that the authority stationary distribution $\mathbf{a}$ is such that $a(v_i) = in(v_i)/\bigcup_{v \in V} in(v)$, and that the hub stationary distribution $\mathbf{h}$ is such that $h(v_i) = out(v_i)/\bigcup_{v \in V} out(v)$. Therefore, SALSA does not follow the mutual reinforcement principle used in HITS, since hub and authority scores of a vertex depend only on the local links of the vertex. Also, in the special case of a single-component network, SALSA can be seen as a one-step truncated version of HITS [30]. Nevertheless, the TKC effect is overcome in SALSA through random walks on the hub-authority bipartite network, which implies that authorities can be identified by looking at different communities.

Figure 2.1 shows an illustrative comparison of various centrality methods discussed in this section, on the same example network graph. The nodes in the graph are colored using a gray palette, such that lighter gray-levels correspond to higher rank scores that a particular centrality method has produced over that graph.

2.1.5 SimRank

SimRank [31] is a general, iteratively mutual reinforced similarity measure on a *link graph*, which is applicable in any domain with object-to-object relationships. The main intuition behind SimRank is that "two objects are similar if they are related to similar objects". For instance, on a hyperlinked document domain like the Web, two webpages can be regarded as similar if there exist hyperlinks between them, or in a recommender system, we might say that two users are similar if they rate similar items (and, in a mutual reinforcement fashion, two items are similar if they are rated by similar users). The underlying model of SimRank is the "random surfer-pairs model", i.e., SimRank yields a ranking of vertex pairs. The basic SimRank equation formalizes the intuition that two objects are similar if they are referenced by similar objects. Given any two vertices u and v, their similarity, denoted as $S(u, v)$, is defined as 1 if $u = v$, otherwise an iterative process is performed, in which the similarity between u and v is recursively calculated in terms of in-neighbors of u and v, respectively. The generic step of random walk of this process is defined as:

$$S(u,v) = \alpha \frac{1}{|B_u||B_v|} \sum_{i \in B_u} \sum_{j \in B_v} S(i,j), \tag{2.27}$$

where α is a constant between 0 and 1. As a particular case, if either u or v has no in-neighbors, then $S(u, v) = 0$. It should be noted that equation (2.27) expresses the average similarity between in-neighbors of u and in-neighbors of v. Moreover, it is easy to see that SimRank scores are symmetric.

The basic SimRank equation lends itself to several variations, which account for different contingencies in a network graph. One of these variations allows for resembling the HITS algorithm (cf. Section 2.1.4), since it considers that vertices in a graph may take on different roles, like hub and authority for importance. Within this view, equation (2.27) can be replaced by two mutually reinforcing functions that express the similarity of any two vertices in terms of either their in-neighbors or out-neighbors:

$$S_1(u,v) = \alpha_1 \frac{1}{|R_u||R_v|} \sum_{i \in R_u} \sum_{j \in R_v} S_2(i,j) \tag{2.28}$$

and

$$S_2(u,v) = \alpha_2 \frac{1}{|B_u||B_v|} \sum_{i \in B_u} \sum_{j \in B_v} S_1(i,j), \tag{2.29}$$

where constants α_1, α_2 have the same semantics as α in equation (2.27). Another variation of SimRank is the *min-max* variation, which captures the commonality underlying two similarity notions that express the endorsement of one vertex towards the choices of another vertex, and vice versa. Given vertices u, v, two intermediate terms are defined as:

$$S_u(u,v) = \alpha \frac{1}{|R_u|} \sum_{i \in R_u} \max_{j \in R_v} S(i,j) \tag{2.30}$$

and

$$S_v(u,v) = \alpha \frac{1}{|R_v|} \sum_{i \in R_v} \max_{j \in R_u} S(i,j), \tag{2.31}$$

with final similarity score computed as $S(u, v) = \min\{S_u(u, v), S_v(u, v)\}$, which ensures that each vertex chooses the other's choices.

SimRank is a computationally expensive method. The space required for each iteration is simply $O(|V|^2)$, whereas the time required is $O(I|V|^2 d)$, where d denotes the average of $|B_u||B_v|$ over all vertex-pairs u, v, and I is the number of iterations. One way to reduce the computational burden is to prune the link graph, which avoids to compute the similarity for every vertex-pair by considering only vertex-pairs within a certain radius from each other [31]. Many other methods to speed up the SimRank computation have been developed in the literature. For instance, Fogaras and Racz [32] proposed a probabilistic approximation based on the Monte Carlo method. Lizorkin et al. [33] proposed different optimization techniques, including partial sums memoization that can reduce repeated calculations of the similarity among different pairs by caching part of similarity summations for later reuse. Antonellis et al. [34] extended SimRank using evidence factor for incident nodes and link weights. More recently, Yu et al. [35] proposed a fine-grained memoization method to share the common parts among different partial sums; the same authors also studied efficient incremental SimRank computation over evolving graphs. At the time of writing of this chapter, the most recent study is that by Du et al. [36], which have focused on SimRank problems in uncertain graphs.

In this section we discuss main approaches to make the process of ranking web documents, or similarly their corresponding users, *topic-sensitive*. The general goal is to drive the ranking mechanism in such a way that the obtained ordering and scoring reflects a target scenario in which the vertices in the network are to be evaluated based on their relevance to a topic of interest. The term topic is here intentionally used with two different meanings, which correspond to different perspectives of quality of web resources: the one normally refers to the *content* of web documents, whereas the other one refers to the relation of web documents with web spammers, and more specifically to their trustworthiness, or likelihood of not being a spammer's target.

2.1.6 Content as topic

As previously mentioned, the PageRank personalization vector $\mathbf{p}$ can be replaced with any probability vector defined to boost the PageRank score for a specific subset of vertices that are relevant to a certain topic of interest.

A natural way to implement the above idea is to make the teleportation query-dependent, in such a way that a vertex (page) is more likely to be chosen if it covers the query terms. More precisely, if we denote with $\mathcal{B} \subseteq V$ a subset of vertices of interest, then $\mathbf{p} = \mathbf{1}/|V|$ is replaced with another vector biased by $\mathcal{B}$, $\mathbf{p}_{\mathcal{B}}$ whose entries are set to $\mathbf{1}/|\mathcal{B}|$ only for those vertices that belong to $\mathcal{B}$, and zero otherwise. Because of the concentration of random walk restarts at vertices from $\mathcal{B}$, these vertices will obtain a higher PageRank score than they obtained using a conventional (non-topic-biased) PageRank.

Intuitively, this way of altering the behavior of random surfing reflects the different preferences and interests that the random surfer may have and specify as terms in a query. Moreover, for efficient indexing and computation, the subset $\mathcal{B}$ usually corresponds to a relatively small selection of vertices that cover some small number of topics. For instance, one might want to constrain the restart of the random walk to select only pages that are classified under one or more categories (e.g., “politics”, “sports”, “technology”, and so on) of the Wikipedia topic classification system[3] or any other publicly available web directory. The consequence of this topic-biased selection is that not only the random surfer will be at pages that are identified as relevant to the selected topics, but also any neighbor page and any page reachable along a short path (from one of the known relevant pages) will be likely relevant as well.

Topic affinity and user ranking. Determining topic affinity in web sources is central to identifying webpages that are related to a set of target pages. Topic affinity can be measured by using one or combination of the following three main approaches.

- *Text-based methods.* Besides cosine similarity in vector space models (e.g., [37]), text-based approaches can also involve *resemblance* measures (e.g., Jaccard or Dice coefficients) which are defined in terms of the overlap between two documents modeled as sets of text chunks [38].

- *Link-based methods.* The link-based topic affinity approach has traditionally borrowed from citation analysis, since the hyperlinking system of endorsement is in analogy with the citation mechanism in research collaboration networks. Particularly, *co-citation analysis* is effective in detecting cores of articles or authors given a particular subject matter. Early applications of co-citation analysis to topical affinity detection and web document clustering include [39]–[41]. Essentially, a citation-based measure of topic affinity can be formalized as co-citation strength or, alternately, as bibliographic-coupling (also called co-reference) strength; i.e., two documents are related proportionally to the frequency with which they are cited together (resp., the frequency with which they have references in common). Furthermore, similarity between two documents can be also evaluated in terms of number of direct paths between the two documents.

- *Usage-based methods.* Finally, the usage-based topic affinity approach is based on the assumption that the interaction of users with web resources (stored via user access and activity logs) can aid to improve the quality of content, thus increasing the performance of web search systems. This approach is strictly related to techniques of web personalization and adaptivity based on customization or optimization of the users’ navigational experiences, as originally studied by Perkowitz and Etzioni [42], [43].

Topic affinity detection is however not only essential to characterize similarity and relatedness of webpages by content but also helpful to drive the topic-sensitive ranking of web sources and their users.

TwitterRank. Topic-sensitive ranking in combination with topic affinity measures is in fact widely used in online user communities, such as social media networks and collaboration networks. An exemplary method is represented by the *TwitterRank* algorithm [44], which was originally designed to compute

[3]http://en.wikipedia.org/wiki/Category:Main_topic_classifications.

the prestige or influence of users in the Twitter environment; the algorithm can however be applied to other platforms similar to Twitter, or in general to any social network providing microblogging services. A directed graph $G = (V, E)$ is used to model *followship* relations among users in Twitter, i.e., there is an edge from vertex v_i to vertex v_j if the i-th user follows the j-th user. Therefore, according to PageRank, the higher the number and influence of the followers, the higher the influence of a user in Twitter. Besides the PageRank principle, a key assumption in TwitterRank is that, since followship presumes content consumption (i.e., reading, or replying to tweets), the influence a user has on each follower is determined by the relative amount of content the follower received from her. This means that in TwitterRank a random surfer performs a topic-specific random walk. Moreover, since users generally have different interests in various topics, influence of Twitter users also vary with respect to different topics. Formally, the (stochastic) transition probability matrix $\mathbf{P}_t$ used in TwitterRank is specified contextually to a topic t, and defined in such a way that, for any users v_i, v_j:

$$P_t(i,j) = \frac{|T_j|}{\sum_{(v_i,v_k)\in E}|T_k|} sim_t(i,j), \tag{2.32}$$

where T_j is the set of tweets published by user v_j, and $sim_t(i,j)$ is the similarity between v_i and v_j with respect to topic t.

To compute the similarity between two users conditionally to a given topic, TwitterRank evaluates the difference between the probability that the two users are interested in the same topic t:

$$sim_t(i,j) = 1 - |\widehat{DT}_{it} - \widehat{DT}_{jt}|, \tag{2.33}$$

where $\widehat{\mathbf{DT}}$ is the row-normalized version of the *document-topic matrix* $\mathbf{DT}$, whose (i,t)-th entry stores the number of times a word in the tweets by user v_i has been assigned to topic t. The document-topic matrix represents a low-dimensional representation of a collection of documents, where a document here corresponds to the set of tweets of a user. This document representation can in principle be obtained by using some linear projection technique, such as LSI, which is able to provide a low dimensional mapping from a high dimensional vector space, using an orthogonal transformation based on singular value decomposition to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables (or components). A document-topic representation can also be obtained via *statistical topic modeling*, which assumes that a document can be represented as a mixture of probability distributions over its constituent terms, where each component of the mixture refers to a main topic. While still using a "bag of words" assumption (a document is treated as a vector of word counts), the document representation is obtained by a generative process, i.e., a probabilistic process that expresses document features as being generated by a number of latent variables. Compared to conventional vector-space modeling, statistical topic models are generally able to involve (latent) semantic aspects underlying correlations between words to leverage the structure of topics within a document. TwitterRank utilizes the well-known Latent

Dirichlet Allocation (LDA) method [45].

As a variant of topic-biased PageRank, TwitterRank equation, for a given topic t, is defined as:

$$\mathbf{r}_t = \alpha \mathbf{P}_t \mathbf{r}_t + (1 - \alpha)\mathbf{e}_t, \tag{2.34}$$

where $\mathbf{e}_t$ is the t-th (normalized) column vector of the **DT** matrix. By aggregating over all topics, the global TwitterRank vector is given as: $\mathbf{r} = \sum_t \omega_t \mathbf{r}_t$, where ω_t is the weight associated to topic t. The authors of TwitterRank suggest a number of ways to compute these topic weights. One of these ways is to set ω_t as the prior probabilities of the various topics, estimated proportionally to the number of times unique words have been assigned to corresponding topics. Alternatively, r_t can be set as the probabilities that a particular user v_i is interested in different topics, which are calculated according to the number of times words in v_i's tweets have been assigned to corresponding topics as captured in **DT**.

2.1.7 Trust as topic

PageRank is vulnerable to adversarial information retrieval. In fact, link spamming techniques can enable webpages to achieve higher score than what they actually deserve. To do this, spammers normally create the so-called spam farms, i.e., collections of pages whose role is to support the artificial increase of the PageRank score of target pages. In a typical scenario, a spam farm owns a certain number n of supporting pages, each of which has a bidirectional connection only with the target page. Moreover, the target page has also incoming links from outside the spam farm; this is made possible by applying one or more link spamming strategies, such as inviting others to post comments on the spammer site (target page). It can easily be demonstrated that the PageRank score $r^{(s)}$ of the spammer's target page can be computed as:

$$r^{(s)} = \frac{r^{(ns)}}{1 - \alpha^2} + \frac{\alpha}{1 + \alpha} \frac{n}{N}, \tag{2.35}$$

assuming that a certain amount $r^{(ns)}$ of PageRank comes from outside the spam farm (i.e., from the pages not owned by the spammer but linked to the target page), and that there are N pages on the Web. Therefore, the size and structure of the spam farm can be manipulated to amplify the PageRank score of the spammer's target page.

Combating link spam has been a necessary task for developers of web search systems in the last years. One approach is to locate the spam farms, knowing that, as we previously discussed, they may have a typical structure where one page links to a very large number of pages, each of which links back to it. However, this approach is not scalable since spammers can always develop farm structures that differ from the known ones.

TrustRank. A different approach is instead to make the PageRank aware of spam or, in general, untrustworthy pages, by inducing topic-sensitivity in order to lower the score of those pages. Within this view, a well-known method that was introduced to combat web spam and finally detect trustworthy pages is *TrustRank* [46]. Basically, the algorithm first selects a small seed set of pages whose "spam status" needs to be determined. A human expert then examines the seed pages, and tells the algorithm if they are spam (bad pages) or not (good pages). Finally, the algorithm identifies other pages that are likely to be good based on their connectivity with the good seed pages. The pages in the seed set are classified using a function called *oracle*, which is as:

$$O(p) = \begin{cases} 0 & \text{if } p \text{ is bad} \\ 1 & \text{if } p \text{ is good.} \end{cases} \tag{2.36}$$

However, at a large scale, oracle invocations are expensive, and in fact it is used only over the seed set. Therefore, to evaluate pages without relying on the oracle function, the likelihood that a given page p is good will be estimated. A key assumption used in TrustRank to identify good pages is the so-called *approximate isolation* principle, that is, "high-quality pages are unlikely to point to spam or low-quality pages". Upon this principle, a *trust* function T is defined that yields a range of values between 0 (bad) and 1 (good). Ideally, for any page p, $T(p)$ gives the probability that p is good: $T(p) = \Pr[O(p) = 1]$. Desirable properties for the trust function are:

- Ordered Trust Property:

$$T(p) < T(q) \Leftrightarrow Pr[O(p) = 1] < Pr[O(q) = 1]$$

$$T(p) = T(q) \Leftrightarrow Pr[O(p) = 1] = Pr[O(q) = 1]$$

- Threshold Trust Property:

$$T(p) > \delta \Leftrightarrow O(p) = 1.$$

Given the network graph $G = (V, E)$, TrustRank first computes the seed set, characterized by a vector $\mathbf{s}$, via the following iterative equation:

$$\mathbf{s} = \beta \mathbf{U} \mathbf{s} + (1 - \beta) \frac{1}{|V|} \mathbf{1}, \tag{2.37}$$

where β is a decaying factor ranging between 0 and 1, and $\mathbf{U}$ is the "inverse" connectivity matrix of the graph:

$$U(i, j) = \begin{cases} 1/in(v_j) & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.38}$$

Note that pages in the seed set should be well-connected to other pages in order to propagate trust to many pages quickly. Therefore, they are chosen among those that have a large out-degree. For this purpose, PageRank is computed by reversing the in-links with out-links in the graph; here a high PageRank score will indicate that trust can flow with a small number of hops along out-links.

Once the $\mathbf{s}$ vector is computed, it is sorted in decreasing order according to the probability that every vertex belongs to the seed set. Only the top-L vertices are retained in the seed set. The next step consists in the computation of the personalization vector $\mathbf{p}$, such that $p(v_i) = 1$ if the vertex v_i belongs to the seed set and is a good page, and 0 otherwise. The TrustRank vector is finally computed using the basic PageRank equation (cf. Section 2.1.2, equation (2.20)), with personalization vector set to the normalized $\mathbf{p}$.

2.1.8 Heterogeneous networks

So far we have discussed information networks under the common assumption of representation as *homogeneous* networks, i.e., nodes are objects of the same entity type (e.g., webpages, users) and links are relationships of the same type (e.g., hypertext linkage, friendship). However, nodes and node relations can be of different types. For instance, in a research publication network context, nodes can represent authors, publications and venues, while relations can be of type "written by" (between publication nodes and author nodes), "cited by" (between publication nodes), co-authorship (between author nodes), and so on. As another example, an online social network consists not only of persons, but also of different objects like photos, tags, texts, and so on; moreover, different kinds of relations may occur among different objects (e.g., a photo may be labeled with a certain tag, a person can upload a photo, write a text or request friendship to another person). Similar scenarios can be found in a variety of application domains, including online e-commerce systems, medical systems, and many others. Consequently, such real-world networks might be conveniently modeled as *heterogeneous* or *typed* networks, in order to better capture the (possibly subtly) different semantics underlying the different types of entities and relationships.

Following [47], given a set of vertex types $\mathcal{T}$ and a set of edge types $\mathcal{R}$, a *heterogeneous information network* (HIN) is defined as a directed graph $G = (V, E)$ with a vertex type mapping function $\tau : V \to \mathcal{T}$ and an edge type mapping function $\phi : E \to \mathcal{R}$, where each vertex $v \in V$ belongs to one particular vertex type $\tau(v) \in \mathcal{T}$, each edge $e \in E$ belongs to a particular relation $\phi(e) \in \mathcal{R}$. If two edges belong to the same relation type, they share the same starting vertex type as well as the ending vertex type. Moreover, it holds that either $|\mathcal{T}| > 1$ or $|\mathcal{R}| > 1$; otherwise, as a particular case, the information network is *homogeneous*.

The *network schema*, denoted as $S_G = (\mathcal{T}, \mathcal{R})$, is a meta template for a heterogeneous network $G = (V, E)$ with set of vertex types $\mathcal{T}$ and set of edge types $\mathcal{R}$.

In [47], Sun and Han provide a set of suggestions to guide systematic analysis of HINs, which are reported as follows.

1. *Information propagation.* A first challenge is how to propagate information across heterogeneous types of nodes and links; in particular, how to compute ranking scores, similarity scores, and clusters, and how to make good use of class labels, across heterogeneous nodes and links. Objects in HINs are interdependent and knowledge can only be mined using the holistic information in a network.

2. *Exploring network meta structures.* The network schema provides a meta structure of the information network. It provides guidance of search and mining of the network and helps analyze and understand the semantic meaning of the objects and relations in the network. Meta-path-based similarity search and mining methods can be useful to explore network meta structures.

3. *User-guided exploration of information networks.* A certain weighted combination of relations or meta-paths may best fit a specific application for a particular user. Therefore, it is often desirable to automatically select the right relation (or meta-path) combinations with appropriate weights for a particular search or mining task based on user's guidance or feedback. User-guided or feedback-based network exploration can be a useful strategy.

2.1.9 Ranking in heterogeneous information networks

Ranking models are central to address the new challenges in managing and mining large-scale heterogeneous information networks. In fact many proposals have been developed for a variety of tasks such as keyword search in databases (e.g., [48]), Web object ranking (e.g., [49]), expert search in digital libraries (e.g., [50]–[52]), link prediction (e.g., [53], [54]), recommender systems and Web personalization (e.g., [55]–[57]), sense ranking in tree-structured data [58]. Some work has also been developed using path-level features in the ranking models, such as path-constrained random walk [59] and PathSim [60] for top-k similarity search based on meta-paths. Moreover, there has been an increasing interest in integrating ranking with mining tasks, like the case of ranking-based clustering addressed by RankClus [61] and NetClus [62] methods. In the following, we focus on ranking in heterogeneous information networks and provide a brief overview of main methods.

ObjectRank. One of the first attempts to use a random-walk model over a heterogeneous network is represented by ObjectRank [48]. The algorithm is an adaptation of topic-sensitive PageRank to a keyword search task in databases modeled as labeled graphs.

The HIN framework in ObjectRank consists of a *data graph*, a *schema graph* and an *authority transfer graph*. The data graph $G_D(V_D, E_D)$ is a labeled directed graph where every node v has a label $\lambda(v)$ and a set of keywords. Nodes in V_D represent database objects which may have a sub-structure (i.e., each node has a tuple of attribute name/attribute value pairs). Moreover, each edge $e \in E_D$ is labeled with a "role" $\lambda(e)$ which describes the relation between the connected nodes.

The *schema graph* $G_S(V_S, E_S)$, is a directed graph which describes the structure of G_D, i.e., it defines the set of node and edge labels. A data graph $G_D(V_D, E_D)$ conforms to a schema graph $G_S(V_S, E_S)$ if there is a unique assignment μ such that:

1. for every node $v \in V_D$ there is a node $\mu(v) \in V_S$ such that $\lambda(v) = \lambda(\mu(v))$;

2. for every edge $e \in E_D$ from node u to node v there is an edge $\mu(e) \in E_S$ from $\mu(u)$ to $\mu(v)$ and $\lambda(e) = \lambda(\mu(e))$.

The *authority transfer graph* can refer to both a schema graph or a data graph. The *authority transfer schema graph* $G_A(V_S, E_A)$ reflects the authority flow through the edges of the graph. In particular, for each edge in E_S two *authority transfer edges* are created, which carry the label of the schema graph edge forward and backward and are annotated with a (potentially different) *authority transfer rate*. The authority transfer schema graph can be based on a trial and error process or on a domain expert task.

A data graph conforms to an authority transfer schema graph if it conforms to the corresponding schema graph. From a data graph $G_D(V_D, E_D)$ and a conforming authority transfer schema graph $G_A(V_S, E_A)$ an *authority transfer data graph* $G_{AD}(V_D, E_{AD})$ can be derived. Edges of the authority transfer data graph are annotated with authority transfer rates as well, controlled by a formula which propagates the authority from a node based on the number of its outgoing edges.

ObjectRank can be used to obtain a keyword-specific ranking as well as a global ranking. Given a keyword w, the *keyword-specific ObjectRank* is a biased PageRank in which the base set is built upon the set of nodes containing the keyword w:

$$\mathbf{r}^w = \alpha \mathbf{A} \mathbf{r}^w + \frac{1-\alpha}{|S(w)|}\mathbf{s}, \tag{2.39}$$

where $S(w)$ denotes the base set specific to w, and $s_i = 1$ if $v_i \in S(w)$ and $s_i = 0$ otherwise. The *global ObjectRank* is basically a standard PageRank. The final score of a node given a keyword w is then obtained by combining the keyword-specific rank and the global rank.

In [48], Balmin et al. also discussed an optimization of the ranking task in the case of directed acyclic graphs (DAGs). More specifically, the authors showed how to serialize the ObjectRank evaluation over single-pass ObjectRank calculations for disjoint, non-empty subsets $L_1, \ldots, L_q$ obtained by partitioning the original set of vertices in a DAG. Upon a topological ordering of L_h ($h = 1..q$) that imposes no backlink from every vertex in L_j to any vertex in L_i, with $i < j$, the ranking of nodes is first computed on L_1 ignoring the rest of the graph, then only the ranking scores of vertices in L_1 connected to vertices in L_2 are reused to calculate ObjectRank for L_2, and so on.

PopRank. In [49], PopRank is proposed to rank heterogeneous web objects of a specific domain by using both web links and object relationship links. The rank of an object is calculated based on the ranks of objects of different types connected to it, and a parameter called *popularity propagation factor* is associated to every type of relation between objects of different types.

The PopRank score vector $\mathbf{r}_\tau$ for objects of type τ_0 is defined as a combination of the individual popularity $\mathbf{r}$ and influence from objects of other types:

$$\mathbf{r}_\tau = \alpha \mathbf{r} + (1-\alpha) \sum_{\tau_t} \gamma_{\tau_t \tau_0} \mathbf{M}^{\mathrm{T}}_{\tau_t \tau_0} \mathbf{r}_{\tau_t} \tag{2.40}$$

where $\gamma_{\tau_t\tau_0}$ is the *popularity propagation factor* of the relationship link from an object of type τ_t to an object of type τ_0 and $\sum_{\tau_t}\gamma_{\tau_t\tau_0} = 1$, $\mathbf{M}_{\tau_t\tau_0}$ is the row-normalized adjacency matrix between type τ_t and type τ_0 , and $\mathbf{r}_{\tau_t}$ is the PopRank score vector for type τ_t. In order to learn the popularity propagation factor $\gamma_{\tau_t\tau_0}$, a simulated annealing-based algorithm is proposed, according to partial ranking lists given by domain experts.

Bipartite SimRank. The *SimRank* algorithm discussed in Section 2.1.5 can naturally be extended to bipartite networks such as, e.g., user-item rating networks. Intuitively, the similarity of users and the similarity of items are mutually reinforcing notions that can be formalized by two equations analytically similar to equation (2.28) and equation (2.29). More precisely, assuming that edges are directed from vertices of type 1 (e.g., users) to vertices of type 2 (e.g., items), and using superscripts (1) and (2) to denote the two types of vertices, respectively, we have the following equations:

$$S(u^{(1)}, v^{(1)}) = \alpha_1 \frac{1}{|R_{u^{(1)}}||R_{v^{(1)}}|} \sum_{i \in R_{u^{(1)}}} \sum_{j \in R_{v^{(1)}}} S(i,j), \tag{2.41}$$

and

$$S(u^{(2)}, v^{(2)}) = \alpha_2 \frac{1}{|B_{u^{(2)}}||B_{v^{(2)}}|} \sum_{i \in B_{u^{(2)}}} \sum_{j \in B_{v^{(2)}}} S(i,j), \tag{2.42}$$

where constants α_1, α_2 have the same semantics as α in equation (2.27). Equation (2.41) corresponds to the similarity between vertices of type-1 is the average similarity between the vertices (of type-2) that they refer to (e.g., items that the two users rated), whereas equation (2.42) corresponds to the similarity between vertices of type-2 is the average similarity between the vertices (of type-1) that they are referred to (e.g., the users who rated the two items).

2.1.10 Ranking-based clustering

Given the diversity of node and link types that characterizes HINs, it is also important to understand how the various nodes of different types can be grouped together. An effective solution is to "integrate" *ranking* and *clustering* tasks. This is the basic idea behind methods such as RankClus and NetClus, which will be discussed next.

RankClus. In [61], the RankClus algorithm is introduced, which integrates clustering and ranking on a bi-typed information network $G = (V, E)$, such that $V = V_0 \cup V_1$, with $V_0 \cap V_1 = \emptyset$. Hence, the nodes in the network belong to one of two predetermined types, hereinafter denoted as τ_0, τ_1. The authors use a bibliographic network as running example, which contains venues and authors as nodes. Two types of links are considered: author-venue publication links, with edge weights indicating the number of papers an author has published in a venue, and co-authorship links, with edge weights indicating the number of times two authors have collaborated. A formal definition of bi-typed information network is reported as follows.

A key issue in clustering tasks over network objects is that, unlike in traditional attribute based datasets, object features are not explicit. RankClus explores rank distribution for each cluster to generate new measures for target objects, which are low-dimensional. The clusters are improved under the new measure space. More importantly, this measure can be further enhanced during the iterations of the algorithm, so that the quality of clustering and ranking can be mutually enhanced in RankClus.

Two ranking functions over bi-typed bibliographic network are defined in [61]: *Simple Ranking* and *Authority Ranking*. *Simple Ranking* is based on the number of publications, which is proportional to the number of papers accepted by a venue or published by an author. Using this measure, authors publishing more papers will have higher rank score, even if these papers are all in junk venues. *Authority Ranking* is defined to give an object higher rank score if it has more authority. Iterative rank score formulas for authors and venues are defined based on two principles: (i) highly ranked authors publish many papers in highly ranked venues, and (ii) highly ranked venues attract many papers from highly ranked authors. When considering the co-author information, the rank of an author is enhanced if s/he co-authors with many highly ranked authors.

Differently from *Simple Ranking* (which takes into account only the neighborhood of a node), the score of an object with *Authority Ranking* is based on the score propagation over the whole network. Assuming to have an initial (e.g., random) partition of K clusters $\{C_k\}_{k=1}^{K}$ of nodes of target type τ_0 of a bi-typed information network, the conditional rank of τ_1-type nodes should be very different for each of the K clusters of τ_0-type nodes (e.g., in the bibliographic network case, the rank of authors should be different for each venue-cluster). The idea is that, for each cluster C_k, conditional rank of V_1, $\mathbf{r}_{V_1|C_k}$, can be viewed as a rank distribution of V_1 , which in fact is a measure for cluster C_k. Then, for each node $v \in V_0$, the distribution of object $u \in V_1$ can be viewed as a mixture model over K conditional ranks of V_1, and thus can be represented as a K dimensional vector in the new space [61]. The authors use an expectation-maximization algorithm to estimate parameters of the mixture model for each target object, and then define a cosine similarity based distance measure between an object and a cluster.

Given a bi-typed information network $G = (V_0 \cup V_1, E)$, the ranking functions for V_0 and V_1, and a number K of clusters, RankClus produces K clusters over V_0 with conditional rank scores for each $v \in V_0$, and conditional rank scores for each $u \in V_1$. The main steps of the RankClus algorithm are summarized as follows [47].

- Step 0 - Initialization: Assign each target node with a cluster label from 1 to K randomly.

- Step 1 - Ranking for each cluster: Calculate conditional ranks for nodes of type V_1 and V_0 and within-cluster ranks for nodes of type V_0. If any cluster is empty, the algorithm needs to restart in order to produce K clusters.

- Step 2 - Estimation of the cluster membership vectors for the target objects: Estimate the parameter of the mixture model, obtain new representations for each target object and centers for each target cluster: $\mathbf{s}_v$ and $\mathbf{s}_{C_k}$.
- Step 3 - Cluster adjustment: Calculate the distance from each object to each cluster center and assign it to the closest cluster.
- Repeat Steps 1, 2 and 3 until clusters are stable or change by a very small ratio ϵ, or until a predefined maximum number of iterations.

NetClus. NetClus [62] extends RankClus from bi-type information networks to multi-typed heterogeneous networks with a *star network schema*, where the objects of different types are connected via a unique "center" type. An information network, $G = (V, E, W)$, with $T + 1$ types of objects such that $V = \{V_t\}_{t=0}^{T}$, has a star network schema if $\forall\ e = (v_i, v_j) \in E, v_i \in V_0 \wedge v_j \in V_t (t \neq 0)$ or vice versa. Type τ_0 is called the center or target type, whereas $\tau_t (t \neq 0)$ are attribute types.

Examples of star networks are tagging networks, usually centered on a tagging event, and bibliographic networks, which are centered on papers. In general, a star network schema can be used to map any n-nary relation set (e.g., records in a relational database, with each tuple in the relation as the center object and all attribute entities linking to the center object).

NetClus aims to discover a set of sub-network clusters, and within each cluster a generative model for target objects is built given the ranking distributions of attribute objects in the network. This ranking distribution is calculated using an authority ranking process based on a power iteration method that combines the weight matrices defined between the various types and the center type. The clusters generated are not groups of single typed objects but a set of sub-networks with the same topology as the input network, called *net-clusters*. Each net-cluster is a sub-layer representing a concept of community of the network, which is an induced network from the clustered target objects, and attached with statistic information for each object in the network.

NetClus maps each target object, i.e., that from the center type, into a K-dimensional vector measure, where K is the number of clusters specified by the user. The probabilistic generative model for the target objects in each net-cluster is ranking-based, which factorizes a net-cluster into T independent components, where T is the number of attribute types. NetClus uses the same ranking functions defined for RankClus (*Simple Ranking* and *Authority Ranking*) adapted to the star network case. The core steps of the NetClus algorithm, given the desired number of clusters K, are summarized as follows [47].

- Step 0: Generate initial partitions for target objects and induce initial net-clusters from the original network according to these partitions, i.e., $\{C_k^0\}_{k=1}^{K}$.
- Step 1: For each net-cluster, build ranking-based probabilistic generative model, i.e., $\{P(v|C_k^t)\}_{k=1}^{K}$.

- Step 2: For each target object, calculate the posterior probabilities ($P(C_k^t|v)$) and update their cluster assignment according to the new measure defined by the posterior probabilities to each cluster.
- Step 3: Repeat Step 1 and 2 until the clusters do not change significantly, i.e., $\{C_k^*\}_{k=1}^K = \{C_k^t\}_{k=1}^K = \{C_k^{t-1}\}_{k=1}^K$.
- Step 4: Calculate the posterior probabilities for each attribute object ($P(C_k^*|v)$) in each net-cluster.

2.2 Trust inference in controversial conditions

In this section, we delve into the role of trust in the context of social interactions, with the final goal of developing an inference algorithm to deal with disambiguation of controversial situations i.e., cases where trust opinions widely differ, without a strong unanimous consensus. In this context, we suggest that a mindful use of local and global information can help the user decide whether to trust or not to trust the other user.

2.2.1 Introduction

Nowadays, online social networks (OSNs) and web communities have imposed their central role in the life of millions of people as the preferred mean to share information, experiences and opinions through continuous interactions. Social interactions happen without any geographical or temporal limit, and the information that anyone shares can easily reach a vast crowd, mainly composed by people which do not have a real direct knowledge of each other (e.g., unlike what usually happens in off-line small communities). In this scenario, it becomes necessary to introduce a mechanism for validating and filtering content, discerning the veracity of the information and selecting the appropriate people with whom to share opinions and preferences. A popular way to address this problem is to take into account the concept of *trust*, i.e., modeling information about how much trustworthy each user considers the other ones. OSNs containing information about trust relationships between pair of users are generally referred to as *trust networks.*

A large body of work exists which focuses on the definition of trust inference algorithms, i.e., methods capable of inferring the level of trust between users not directly connected to each other [63], [64]. Trust inference algorithms are typically classified into two main categories: *local* and *global* methods. Local methods infer the trust from a source node to a sink (i.e., target) node by modeling the trust propagation flow across the paths connecting the two nodes. This approach preserves the subjectivity of the trust statements, i.e., inferred trust values for the same target user can be completely different when coming from different source nodes. Conversely, global approaches are based on the calculation of a single trust evaluation for each user, which should reflect the opinion of the entire community; the notion of trust, which in this case can also be referred to as *reputation*, is considered as a global ranking of the users [65]–[67].

Controversial situations can easily arise in trust networks because of the discordance between personal (subjective) opinions [68]–[71]. These situations may occur when there is no agreement in the trust statements towards a user, leading to interpretation and reliability problems in the trust inference process. Local trust inference approaches have been proved to be more effective in presence of controversial conditions [69].

The aim of this work is to highlight the improvements that can be obtained in this context by combining information coming from both local and global trust aspects in the disambiguation of *controversial* situations, i.e., cases where the trust opinions widely differ from each other, without a strong unanimous consensus. Our key ideas and contributions are the following:

- We analyze several controversial cases, considering the use of both global and local methods to improve performances in terms of error prediction and accuracy.
- We propose a novel trust inference algorithm, namely *TrustZic*, which takes into consideration both local and global trust aspects, i.e., propagating trust evaluations proportionally to the global reputation of the trustor nodes.
- We provide explanatory examples and experimental results in a real dataset, proving the effectiveness and significance of the proposed *TrustZic* in solving controversial situations.

The remainder of the work is organized as follows. Section 2.2.2 introduces the scenario which motivates our proposal, Section 2.2.3 discusses related works, and in the following section we introduce the proposed algorithm *TrustZic*.

2.2.2 The motivating scenario

This section introduces a typical scenario that highlights the motivations underneath the proposed research work. Figure 2.2 shows a simple controversial condition that can happen in a social network context. Paul needs to hire a new employee for his company. After advertising the open position, Paul receives an application from Ken. Unfortunately, Paul does not know anything about Ken, but he is in touch with John and Mark, who had previous work experiences with Ken. Paul is happy because he can now take a decision based on the experiences reported by John and Mark. After asking them, he receives two controversial opinions about Ken. In fact, John recommends Ken as a good choice for filling the job position, while Mark reports negative experiences about the behavior of Ken at work. As Paul has the same consideration of John and Mark, he is unable to take a decision about Ken.

This is a typical situation of trust controversy, where trusted users report contrasting opinions. A reasonable decision about trusting or not a user in such a controversial situation is not possible, since the trusting paths connecting source and target nodes are propagating opposite opinions.

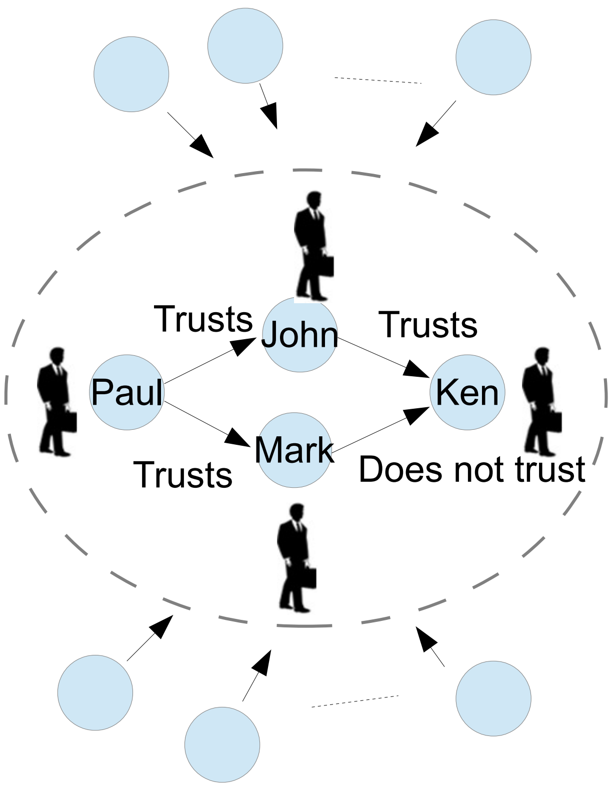


FIGURE 2.2: A typical trust controversy case: it is not possible to infer whether or not Paul should trust Ken.

The main idea of our proposed approach to solve this problem is to exploit information from the rest of the trust network in order to find a way for solving the controversial situation. To this purpose, it is necessary to introduce a criterion of discernment. The most natural way to differentiate the reported trust opinions is to give a different weight depending on the reputation of the trustors. In fact, each user has a different influence with respect to the reputation built in the specific context. The reputation influences the way the opinions, suggestions and recommendations are considered.

Figure 2.3 reports an extended version of the previous example, where the focus is broadened to include more the network. In this case, the global reputation of the nodes can be used to take a reasonable decision about whether Paul should or should not trust Ken. The extended scenario shows that John is trusted by all his in-neighbors, while Mark received both trust and no-trust statements: in this condition it is clear that John has a higher reputation than Mark. After considering the different reputations of John and Mark, Paul decides to take a final resolution about Ken. In fact, Paul finds that the reputation of John is much higher than that of Mark, so he decides to consider the opinion coming from John more reliable, and finally hires Ken. According to the reported considerations, in the proposed approach the reputation will be used as a discerning weighting factor in controversial situations.

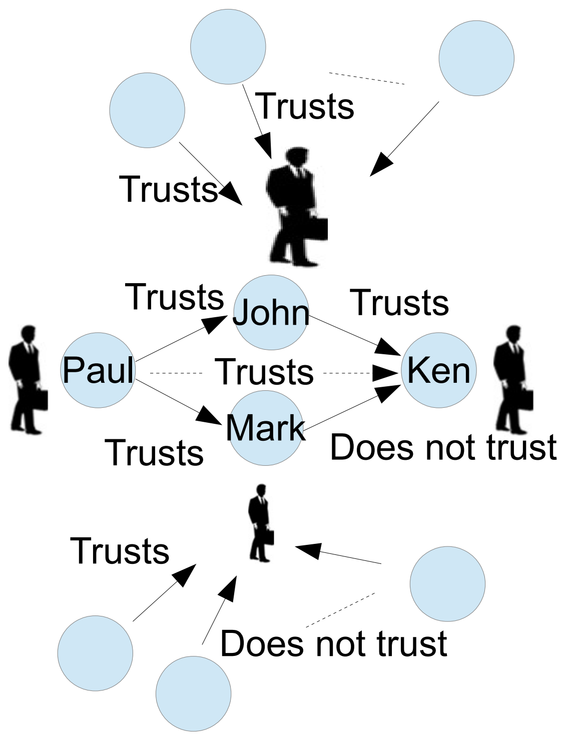


FIGURE 2.3: A controversy case addressed in a global scenario, where the size of the user figures is proportional to their reputation.

2.2.3 Related work

Trust inference

A trust network is usually modeled as a directed graph $G = \langle V, E, T \rangle$, consisting of a set V of nodes (users), a set of links (edges) that represent relations between users, and an edge weighting function T modeling the corresponding trust level of the edges.

The trust network is derived directly from a social network when users are requested to rate each other. *Epinions*[4] is a premier consumer reviews platform on the Web based on the recommendation and the trust among the users who review commercial products. In the *Epinions Web of Trust* technology, users can specify their trust in other users for receiving notifications when the latter ones review new products. In [72], the *Friend-Of-A-Friend* (FOAF) project[5] aimed at building a Web of acquaintances, was extended in order to include information of the level of trust on a scale of 1-9 that users can indicate with respect to the people they know.

Whenever there is no information about how much a user rates other users, the trust network can be derived by evaluating trust statements and interactions among users [73]–[75]. In [75], an algorithm based on the communication behavior in social communication networks is proposed in order to measure the

[4]http://www.epinions.com
[5]http://www.foaf-project.org/

trust among users. The trust is measured by quantifying the behavior in terms of length and frequency of communications, and propagation of information among users. Experiments were conducted on Twitter, where the mechanism of *retweet* was used for measuring the trust contribution coming from propagation.

The purpose of a generic trust inference algorithm IT is to compute a prediction of the trust from a source node u to a sink node v when there is no direct link between the two nodes, i.e., $(u, v) \notin E$. Obviously, there is no need to calculate a prediction when an explicit trust statement was issued, i.e., $IT(u, v) = T(u, v)$ if $(u, v) \in E$. Several trust inference algorithms have been proposed in literature for prediction purposes with different characteristics and performances. The trust inference algorithms can be classified into two main categories: *global* and *local* methods [76]. Global algorithms compute an overall evaluation, here commonly named *reputation*, for each user of the trust network; thus, the reputation can be considered as the synthesis of the trust evaluations from the entire network with respect to a single user. The well-known *PageRank* [77] can be considered as a basis for the development of global trust methods [65].

Nevertheless, global trust inference methods are not suited to preserve the characteristic of personalization in the subjective expression of an opinion, which can instead be achieved by using local trust inference algorithms. One of the most popular local trust inference algorithms is *TidalTrust* [76], which computes the trust between non-adjacent nodes by considering only shortest paths through trusted neighbors. Another local method is proposed in [78], namely *TISoN* (Trust Inference within online Social Networks). In this case a selection of the trust paths from a source node to a sink node is performed with respect to two criteria: the maximum allowed depth of the path, and the minimum threshold level allowed for the trust edges in the path. Then the inferred trust is computed selecting the *most trustable path*, based on the average of the trust edges in the path, the path variance of the trust edges w.r.t. the average, and a path weight based on its length. As regards other approaches, a detailed survey of trust inference algorithms in literature is presented in [64].

Trust controversy

As trust statements reflect subjective opinions built on personal experiences, it is often the case that users report contrasting and different trust opinions about the same target. When trust statements differ too much from each other, it becomes challenging to derive affordable inferred trust information. The presence of controversial conditions makes the trust inference problem more complex to address. Thus, the identification of controversial conditions requires to be explicitly taken into account when dealing with trust networks. The concept of controversy in trust networks has been widely studied in literature [68]–[71].

In [68], the controversy is referred to as single users that are trusted and distrusted by many, i.e., without a global agreement as regards their trustworthiness. The level of controversy for a generic user u is defined by two metrics, controversiality level (c_l) and controversiality percentage (c_p) as follows:

$$c_l(u) = \min(\#r_t(u), \#r_d(u)) \tag{2.43}$$

$$c_p(u) = \frac{\#r_t(u) - \#r_d(u)}{\#r_t(u) + \#r_d(u)} \tag{2.44}$$

where the received trust ($\#r_t$) and the received distrust ($\#r_d$) represent the number of trust and distrust edges to the node u, respectively.

Controversy was studied by [70] and [79] in the context of recommender systems, where *controversial reviews* are defined as those reviews that receive a variety of high and low scores, reflecting disagreement about reviews. Those studies introduced a measure of controversy in terms of standard deviation σ of the reviews of an item i and the *level of disagreement* α defined as follows:

$$\alpha(\Delta, i) = 1 - \max_{a \in \{1,\dots,m-\Delta+1\}} \left(\frac{\sum_{s=a}^{a+\Delta-1} f_i(s)}{\sum_{s=1}^{m} f_i(s)}\right) \tag{2.45}$$

where Δ is the level of a discrete rating scale from 1 to m, and $f_i(s)$ is the number of times that the item i received a rating of score s.

In this work, we will use the metric proposed in [71] in order to evaluate the controversy of the trust inferred from a source node A to a generic sink node Z. The main idea is to consider the controversy as the variance among the trust opinions of the predecessors of the source nodes. To this purpose, the mean trust $TM(A, Z)$ is calculated as follows:

$$TM(A, Z) = \frac{\sum_{P_i \in Pred(Z)} IT(A, P_i) \times T(P_i, Z)}{\sum_{P_i \in Pred(Z)} IT(A, P_i)} \tag{2.46}$$

where $Pred(Z)$ is the set of predecessors of the node Z, IT indicates the inferred trust values and T the original trust weights.

According to Equation 2.46, the mean trust is the weighted average of the trust values assigned by the predecessors P_i of node Z, where the weights are the inferred trusts from the source node A to the predecessor nodes P_i. As the controversy measure is related to the level of discordance among the different trust opinions reported to the source node through the trust chains up to the sink node, the controversy measure $TC(A, Z)$ is defined as the weighted variance of the inferred trust paths, as reported in the following:

$$TC(A, Z) = \frac{\sum\limits_{P_i \in Pred(Z)} IT(A, P_i) \times (T(P_i, Z) - TM(A, Z))^2}{\sum\limits_{P_i \in Pred(Z)} IT(A, P_i)} \tag{2.47}$$

The normalized trust controversy measure (NTC), introduced to bring TC into the range $[0, 1]$, can be computed as follows:

$$NTC(A, Z) = \frac{TC(A, Z)}{((Tmax - Tmin)/2)^2} \tag{2.48}$$

where $Tmin$ and $Tmax$ are the extremes of the trust range.

Local vs. global methods

How global and local methods should be used to solve different trust inference problems was widely explored in [80]. An interesting comparison of local and global trust metrics is reported in [68], where the characteristics of the two approaches are argued in detail. Results on real datasets proved that global methods show intrinsic limit when controversial users (i.e. users judged in different ways by other users) are analyzed, while local methods are more suitable in contexts where opinions are more subjective like in the controversial conditions. The study in [69] showed how local trust metrics outperform global ones when predicting the trustworthiness of users characterized by both trust and distrust evaluations.

The idea of improving the performance of prediction systems by exploiting both global and local trust metrics is not new in literature. A hybrid global-local method was introduced by [81] in the context of service recommendation systems. The scenario depicted in [81] is different from the one discussed in this work, as it considers a recommendation system consisting of different services rated by users integrated with a trust network. The trust network is used to support the computation of the personalized rating prediction $PT(u, s)$ of a service s from a user u as the weighted average reported in Equation 2.49 of the global $r(s)$ and local $T(u, s)$ rating evaluation of the service s, where w_l and w_r are the weights used for properly tuning the local and global contributions, respectively:

$$PT(u, s) = \frac{r(s) \times w_r + T(u, s) \times w_l}{w_r + w_l} \tag{2.49}$$

The promising results obtained by exploiting both global and local perspectives in recommendation systems have encouraged us to investigate how combining the two approaches can be beneficial for improving the performance of trust inference algorithms in controversial conditions.

2.2.4 The TrustZic algorithm

The proposed trust inference method, namely *TrustZic*, aims at solving problems arising when trying to infer trust values in controversial cases, by exploiting global and local trust information in a combined solution. In the proposed approach, the level of trust inferred from the source to the sink will still be personalized because it depends on the propagation of trust along the paths between the source and the sink across the trust graph. Nevertheless, the nodes in the traversed paths are not equally considered, but they exert a different bias on the trust propagation, i.e., proportional to their reputation which is obtained by applying a global trust method.

Figure 2.4 shows the main steps of the *TrustZic* approach: in the first step a global ranking is calculated from the trust graph in order to get a reputation value $r(i)$ for each node i. The *trust-reputation network* is obtained from the trust network by adding the reputation information on the nodes. In the second step, the trust between nodes not directly connected in the trust network is inferred. The trust inference method propagates the trust from the source node to the sink through the trust paths weighted based upon the reputation of each node traversed across the path. The resulting trust inference network is a complete graph in which nodes are connected by trust relations (solid edges) or inferred trust relations (dashed edges). In the following, we will give detailed information on how we perform these steps.

Step 1: computation of the global trust values. A *PageRank*-like algorithm is here used to implement the first step of the proposed method (i.e., obtaining the global ranking of all the nodes in the network). The stochastic matrix M is obtained as the transpose and normalized version of the trust matrix T, as shown in (2.50).

$$M[i,j] = \frac{T[j,i]}{\sum\limits_{k \in adj(j)} T[j,k]} \tag{2.50}$$

The vector r of reputations for all the N nodes is calculated as in (2.51), where α is a smoothing factor (by default set to 0.85) and p is the column vector with the initial reputation value (by default $p = \mathbf{1}$, i.e., the column vector of ones).

$$r = \alpha M r + \frac{1-\alpha}{N} p \tag{2.51}$$

We then adopt the final PageRank solution (i.e., vector of the stationary distribution of PageRank scores at convergence) to assign reputation values to the nodes in the network. However, this can be thought of as a general workflow, where different global ranking methods can be used in theory at this step.

Step 2: local trust inference propagation. The second step consists in propagating the trust from a source node to a sink node by taking into consideration not only the trust T but also the reputation r. The inferred trust $IT(i,j)$ from the source node i to the sink node j is computed as reported in (2.52), where $d \in (0,1]$ is the decay trust factor, introduced for modeling the fact that the trust generally decreases when increasing the distance from the source node.

$$IT(i,j) = \begin{cases} T(i,j) \text{ if there is a direct trust} \\ \frac{\sum\limits_{k \in neighbor(i)} r(k) \times d \times T(i,k) \times IT(k,j)}{\sum\limits_{k \in neighbour(i)} r(k) \times T(i,k)} \text{ otherwise} \end{cases} \tag{2.52}$$

In order to disambiguate controversy, trust is propagated from the source node through all the different paths reaching the sink node, without excluding the contribution of low trusted paths. Starting from the source node, the trust inference is calculated by averaging the trust (direct or inferred) for the sink node reported by all the directly connected trusted nodes (neighbors). More in detail, the aggregation of the trust coming from the different paths is obtained as a weighted average, where the reported trust is weighted by the reputation of the neighbors. The trust inference function is recursively computed along the paths in order to propagate the trust up to the target node.

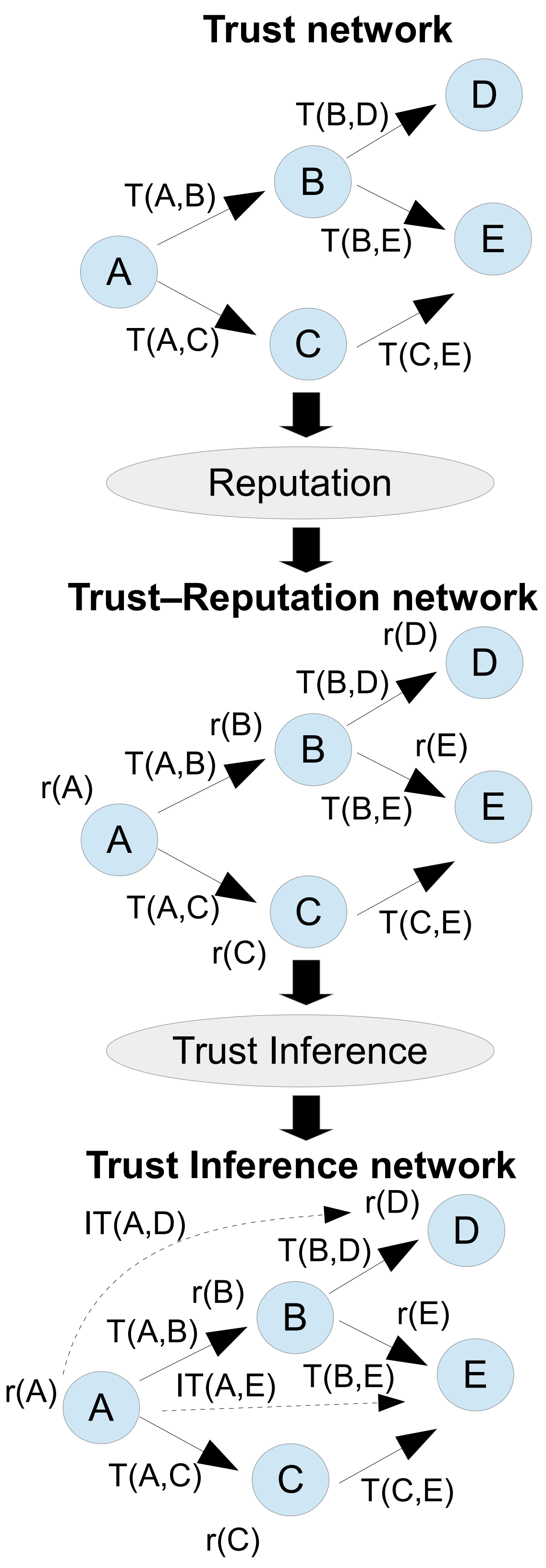


FIGURE 2.4: Main steps of the TrustZic algorithm.

Handling controversial cases with TrustZic

In this section we will give some preliminary numerical examples in order to show how *TrustZic* is able to address controversial cases by exploiting the reputation of the intermediate nodes. We consider a trust range $TR = [0, 1]$, so that $T(i, j) \in [0, 1]\ \forall(i, j) \in E$. $T(i, j) = 1$ means that user i fully trusts user j, while $T(i, j) = 0$ means lack of trust.

For the sake of simplicity and to isolate the reputation effect, only in this explanatory example, all the trust opinions will be considered equal to 1, while the lack of trust is uniformly considered equal to 0, and the effect of the trust decay with the path length is neglected ($d = 1$).

Figures 2.5, 2.6 and 2.7 show three different controversial cases. In the first case (Fig. 2.5), a node Z receives two opposite trust opinions from nodes B and C, propagated up to node A. In this case, the reputation of nodes B and C is the same due to the symmetric configuration, thus the resulting inferred trust is exactly the mean of the two reported trust opinions ($IT(A, Z) = 0.5$).

This is a critical situation when trust inference is used to make decisions. In fact, trust inference can be used as a decision support system by setting a trust threshold value for delimiting the trusting by the no-trusting zones. A generic node X decides to trust another node Y only if the inferred trust from the network is above a fixed threshold. In this specific example, if the threshold is reasonably set to the intermediate value ($Thres = 0.5$) of the $[0, 1]$ trust range, no decisions can be taken because the inferred trust is exactly equal to the threshold ($IT(A,\ Z) = 0.5 = Thres$).

The second and the third cases (Figures 2.6 and 2.7) consider the connections of the rest of the trust network with nodes B and C, and their effects on their reputation. In the second case, the reputation of node B is higher ($r(B) = 0.28$) than that of C ($r(C) = 0.13$) because of the two nodes (D and E) trusting B. In the third case, the reputation of node C is further decreased ($r(C) = 0.12$) as a consequence of the 0-trust inner edge from (F). The effects of the reputation changes become relevant for the resolution of the controversial case in the inferred trust between nodes A and Z. In fact, in the cases reported in Figures 2.6 and 2.7, the inferred trust allows to easily take decisions: node A will trust node Z because the reputation of node B, reporting a positive opinion, is higher than that of node C, reporting a lack of trust opinion.

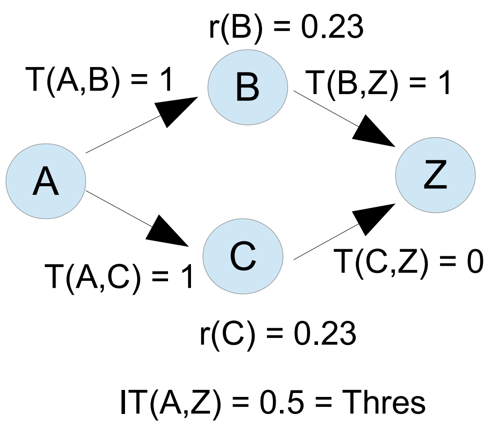


FIGURE 2.5: An undecidable controversial case.

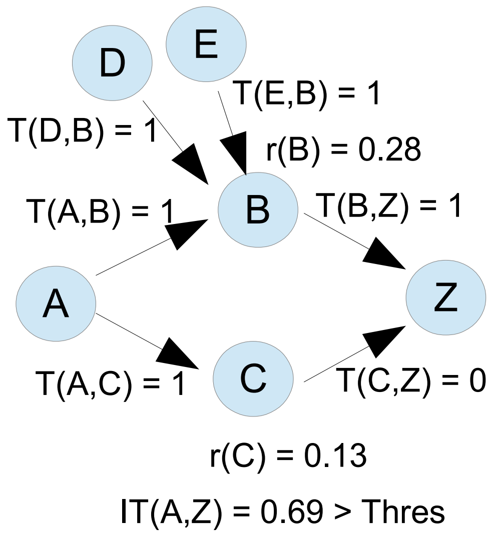


FIGURE 2.6: The controversial case resolved by *TrustZic* due to the increased reputation of node *B*.

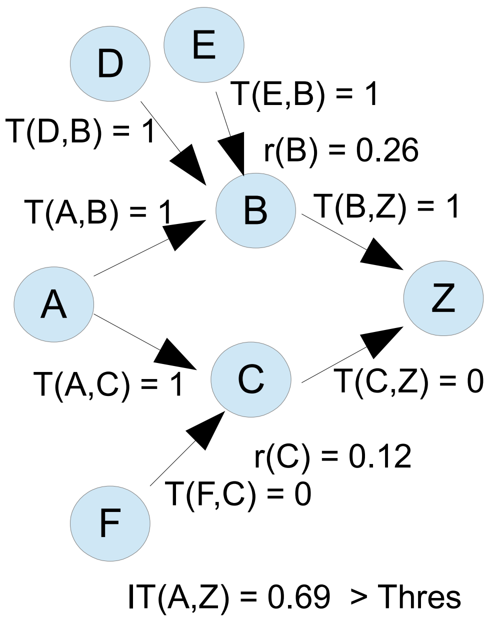


FIGURE 2.7: The controversial case resolved by *TrustZic* due to the decreased reputation of node *C*.

2.3 Conclusion

Trust regulates relations among people, in offline relations as well as in online social networks and web communities, where the interaction among users in the absence of any direct connection can easily happen. In this context, trust inference algorithms are a fundamental tool to estimate missing trust evaluations. Controversial situations caused by disagreement in the existing trust statements towards a user need to be specifically taken into account during the inference process. The trust inference algorithm *TrustZic*, in which global and local trust aspects are combined to infer reliable trust values also in presence of controversial situations, has shown significance and effectiveness of its approach in solving critical trust inference scenarios (experimental results can be found at [71]).

Chapter 3

Alternate behaviors in multilayer social networks

3.1 Introduction

Online user behaviors have been traditionally investigated in function of the role(s) that a user takes as a member of a *single* web-based social environment. This often corresponds to focusing on online social networks (OSNs) that are built on a single type of user relation type (e.g., followship, liking/rating, commenting) or platform context. However, nowadays it is naturally occurring that an individual has multiple accounts across different OSNs. This makes it important to link distributed user profiles belonging to the same user from multiple platforms, which can be beneficial for several applications; for instance, personalized Web services such as recommendation systems can take advantage of the aggregation of user profiles to alleviate the cold-start problem. A further scenario comes out by considering that real-life relations (e.g., working relationship, having lunch, etc.) can be available for the same population of an OSN, and hence they should be profitably exploited to enhance our understanding of the individuals and their relations in an OSN [82].

The above scenarios and furthermore others correspond to complex real-world network systems, which are indeed pervasive in many fields of science. Such systems can effectively be studied using a *multilayer network* model, which especially in social computing, has been recognized as a powerful tool to better (i.e., more realistically) understand the characteristics and dynamics of different, interconnected types of user relations [82]–[84].

3.1.1 Motivations

The dichotomy between production and consumption of resources in OSNs. Regardless of the peculiarities of the scenarios above discussed, the intrinsic complexity of a multilayer network system, in which users and their relations are represented, inevitably impacts on the motivational factors that ultimately determine the users' behavioral profiles. According to the unified model proposed in [85], there are essentially four categories of factors influencing user behaviors: environmental factors — which affect the user's feeling of the community and therefore influences the user's willingness to spend time or to contribute to the community; commitment factors and quality requirement factors — which are based on the relationship between the users and the community; and, of course, individual factors — which refer to the personalities of the users and their purpose of joining the community, such as needs and expectations, but also self-efficacy, extroversion/introversion, self-disclosure, etc. Through the combination of these factors, the resulting online behaviors can be analyzed from three broad perspectives: the development and spread of community norms, the contribution of valuable resources, and the consumption of resources.

Let us focus on the dichotomy between *contribution* and *consumption* of information resources. In this respect, depending on one or many of the influencing factors discussed above, *the same user may take different roles in terms of contribution and consumption of resources*. This is even more intuitive in a multilayer network context, where each layer corresponds to a distinct social environment that might significantly differ from that of the other layers. For instance, consider a multilayer network representing users that are members of many OSNs: it is not surprising that a user can frequently post contents and/or actively interact with other users in one or more layer OSNs, and on the contrary, s/he may be less involved in tangible participation and interactions with other members, thus assuming a *silent* behavior on other OSNs. Generally speaking, in the first case the user is producing resources, while in the other case s/he is rather consuming resources. Here we leave out of consideration the inactivity status, since users that are subscribers of an OSN without accessing it should not be considered as real members of the underlying online community.

The value of lurkers. Unlike inactive users, silent users actually gain benefit from the community, since they observe the user-generated communications though rarely visibly participating in them and contributing information themselves. For this reason, they are also called as *lurkers*. Lurking behaviors have been widely studied in social science, and the importance of analyzing it as a valuable form of online behavior has been definitively recognized [85], [86]. Moreover, lurking is strictly related to legitimate peripheral participation to learn the netiquette and social norms [87], individual information strategy of microlearning [88] and knowledge sharing barriers [89], individual motivation for interpersonal surveillance [90]. According to the model by [85] previously mentioned, reasons for lurking include environmental influence (e.g., bad usability/interaction design, information overload, low response rate, low reciprocity), personal aspects (e.g., introversion, bashfulness, lack of self-efficacy, no need to post – only seeking for information), lack of commitment to the community (e.g., low verbal and affective intimacy with others, unwillingness to spend too much time to maintain a commitment), or security concerns (e.g., worrying about the violation of private information, perception of poor quality requirements of security). The latter aspect is particularly crucial as related to the challenging design of defense strategies in the cyberspace, especially of social network platforms whose common habit is rather to encourage their users to self-disclose their personal information. In fact, as studied in [91], lurking can be explained as a conservative communication strategy adopted by users to protect their personal information while, in the form of passive participation, still connected in online communities; also, users having lack of computer self-efficacy may resort to lurking, thus keeping themselves as part of the online communications but without causing any harm to themselves or to members of their social network.

The coexistence of contributor and lurker behaviors. By getting information from other members of an OSN, lurkers become aware of the existence of different perspectives and may make use of these perspectives in order to form their own ideas, opinions, and even competence. Therefore, lurkers have the potential of exploiting the acquired knowledge outside their own OSN. On the other hand, when users decide to contribute to the community by performing a social action, often they may have formed their decision from exogenous sources of ubiquitous and/or specialist knowledge. In this regard, *social boundary spanning* theory [92], [93] explains how OSN users may naturally get knowledge from some of their social contacts and span it from one or more components to others in the network. A key role is played by those users that lay on the boundary of a component and act as *bridges* over other components. Relations have been studied not only between bridges and influential contributor users (e.g., [94]–[96]), but also between lurkers and bridges. For instance, Morales et al. [95] highlight the importance of bridges in spreading messages throughout a Twitter network organized into an event-driven community structure, where information-producers are found to create communities of information-consumers around them. In our previous work [97], we explain the relation between lurkers and community bridges identified through percolation analysis based on directed topological overlap. In another study [98], lurkers are found more likely to acquire information from outside their own community than to spread information towards other communities; this finding is also complementary to an important result of the application of an influence-maximization method of lurker engagement developed in [98], whereby active users who might help in engaging lurkers could be identified among members of communities external to those of lurkers. Major principles underlying the social boundary spanning theory can also be extended to a multilayer network, where the layers correspond to the boundaries in the complex system.

Following the above intuition, in this work we focus on the understanding of the knowledge flow across different layers of a complex network, from the perspective of both contributors and lurkers. Given the different contingencies occurring in the various layers and, consequently, the different behavioral profiles the same user can have therein, we leverage the importance of studying the behavior of users which may *alternately take the role of contributors and lurkers over different layer networks.*

Applications. The analysis of such alternate behaviors is of crucial importance in order to deeply understand how information and knowledge are produced, consumed, and shared through multiple OSNs. We believe that the development of effective methods for the problem under consideration can support several classes of applications. One of these concerns information filtering and propagation for *advertising*, *marketing campaigns* and related applications. For instance, in tasks of influence maximization [99], we can reasonably assume that contributors are candidates to be selected as initial influencers (i.e., seed users), and lurkers as potential targets of the information to be propagated; in a multilayer context, in order to invest on a more socially diversified set of promoters, it would be useful to learn the seeds not only among the (indirect) social contacts of a target in a layer network, but also among her/his contacts over other layers. Analogously, the identification of users that may consume information from a specific network and subsequently spread it across different platforms would also be beneficial in campaigns of *information containment*. This is also important for security reasons, e.g., to control that specific (e.g., sensitive) information be shielded from certain users, to cope with the diffusion of fake news through different web sources, etc.

Another class of applications concerns *user engagement*. OSNs encourage users to self-disclose their own opinions and actively contribute to the community life. This requires the development of so-called "delurking" strategies, such as providing rewards (i.e., grants to be offered to the users for their active contributions to the community) [100], enhancing users' commitment to the community [101], making lurkers understand the necessity of their contribution [102], learning or improving usability of the system [103]. Generalizing to a complex system, valuable indications might be provided from the platform(s) on which a user is an active contributor to tailor a suitable strategy for engaging her/him on other platforms.

From a broader perspective, identifying alternate behaviors in multiple OSN contexts can be a key tool to enhance the development of effective communication strategies through different online social environments, and ultimately to face challenges in today's knowledge management systems that concern users' motivations for online participation in different forms (i.e., active vs. passive participation).

3.1.2 Contributions

In this work, we address the novel problem of identifying and characterizing users who alternately behave as contributors and as lurkers over multiple layer networks in a complex system.

To the best of our knowledge, we are the first to study the dichotomy between information-producers (i.e., contributors) and information-consumers (i.e., lurkers) and their interplay over a multilayer network. To support this claim of novelty, note that while research on the identification and ranking of contributors and users with related roles (i.e., influencers, leaders, experts, etc.) in multilayer networks has recently gained momentum (e.g., [83], [84], [104]–[108]), none of these studies addresses problems of analysis of opposite and alternately occurring behaviors. Moreover, computational approaches to the analysis of lurking behaviors have been developed only for monoplex networks [97], [98], [109]–[112]. By contrast, in this work, we take into consideration both behavioral "opposites", leveraging on their duality and possible, "alternate" manifestations over many, interconnected layers of the same complex network system. This represents a major novelty aspect of our approach, which introduces a previously unexplored notion of centrality or prominence in multilayer network that can effectively be used to support emerging and challenging applications in complex network systems, as previously discussed in Sect. 3.1.1.

Our proposed approach is devised in terms of a twofold, mutually reinforcing ranking model, which can capture the interplay of the two behavioral roles over a multilayer network. The meaning of "alternate" stands for the contingency that the behavior of the same user can be regarded as lurking in one layer and as active participation in one or many of the other layers, or vice versa. We formally define the alternate lurker-contributor behavior status of any node in a multilayer network, conditionally to a given layer, following interrelated principles of cross-layer lurker behavior and cross-layer contributor behavior. Upon this, the proposed *Alternate Lurker-Contributor Ranking* method (mlALCR) is mathematically expressed as two mutually dependent equation systems, where one system determines the layer-specific lurking score and the other system determines the layer-specific contributor score for all nodes in the interconnected layers of the complex network.

Our mlALCR has been extensively evaluated on four real-world multilayer networks, specifically two cross-platform user multi-relational networks, a single-platform user multi-relational network, and a single-platform temporal multi-slice network. We assessed significance and meaningfulness of mlALCR both quantitatively and qualitatively. Experimentation also includes comparison of mlALCR with the basic (i.e., monoplex) lurker ranking method, its extension to multilayer networks — originally proposed in this work — ranking aggregation methods, and the Multiplex PageRank method. Results show the unique ability of mlALCR to effectively determine the two behavioral opposites in terms of information-production and information-consumption that a user may take over different layers of a complex network.

The remainder of this chapter is organized as follows. Section 3.2 briefly discusses related work, focusing on approaches to the measurement of node centrality in multilayer networks. Section 3.3 introduces to the identification of contributors in multilayer networks and the analysis of lurking behaviors. Section 3.4 is devoted to our proposed mlALCR method. Sections 3.5 and 3.6 present experimental methodology and results. Section 3.7 concludes the chapter.

3.2 Related Work

The notion of *centrality* commonly resembles the importance or prominence of a node in a network, i.e., the status of being located in strategic locations within the network. Concerning multilayer networks, we can distinguish two main approaches in the literature, which mainly differ from each other in the way they handle the additional complexity introduced by inter-layer edges. One approach is based on some strategy of aggregation, i.e., projection of the multilayer network to a single aggregated layer or aggregation of layer-wise results independently generated. The other approach comprises techniques which are designed to take into account both intra- and inter-layer connections.

Aggregation and projection-based approaches. Brodka et al. propose a definition of degree centrality for multilayer networks in [113]. The cross-layer degree centrality of a node v is measured as proportional to the weighted sum of edges connecting v with the members of its multilayered neighborhood, which is the set of nodes directly connected with v on at least a certain number of layers. In [106], De Domenico et al. study main factors influencing the navigability of multilayer networks, using random walks over a layer-aggregated network. The authors find that navigability is influenced by both structural characteristics and exploration strategies, and that a multilayer network is more resilient to random failures than the single layers separately. In general, when dealing with centrality scores produced at distinct layer networks, the use of different aggregation and normalization techniques is shown to strongly bias the final results [114].

Cross-layer approaches. Sole et al. [105] define a multilayer extension of the betweenness centrality by taking into account shortest paths which include inter-layer edges. This measure is shown to favor nodes which act as bridges between layers, allowing to connect individuals which are disconnected inside the layers. A cross-layer betweenness centrality is proposed in [115], also including applications of the proposed measure to multilayer community detection and message spreading tasks. Two classes of multilayer degree-biased random walks are proposed in [116], in order to analyze to what extent this kind of random walks can make the exploration of multilayer networks more efficient. The two classes refer to extensive walkers (i.e., biased w.r.t. node properties independently considered for each layer) and intensive walkers (i.e., the bias is proportional to intrinsically multilayer features, as the overlapping degree and the participation coefficient). Kuncheva and Montana [117] also employ multilayer random walks for community detection purposes. Their method utilizes inter-layer transition probabilities that depend on local topological similarity between corresponding nodes, in order to facilitate the discovery of cross-layer communities. A hierarchical clustering method is finally used to produce the communities based on the outcome of a random walk process. An important mention here regards the multilayer extension of the well-known PageRank algorithm proposed in [104], which we shall discuss in detail in Sect. 3.3.2 and that will be used as one of the competing methods in our experimental evaluation.

It should be noted that all the aforementioned approaches are either extension of classic centrality measures to a multilayer context [105], [113], [115], or approaches which exploit random walk models on multilayer networks to extend eigenvector centrality [104], [116] or to use it for specific tasks (e.g., community detection [115], [117]). By contrast, we propose a new ranking method explicitly designed to model opposite behaviors and their interplay over the multiple layers of a complex network system, leveraging information coming from user interactions through the different layers.

3.3 Lurker and contributor behaviors in multilayer networks

3.3.1 Multilayer network model

Let $\mathcal{L} = \{L_1, \ldots, L_\ell\}$ be a set of *layers*, with size equal to or greater than two. Each layer corresponds to a particular user relational context. Given a set $\mathcal{V}$

of *entities* (i.e., users), for each choice of user in $\mathcal{V}$ and layer in $\mathcal{L}$, we need to indicate whether the user is present in that layer. We denote with $V_{\mathcal{L}} \subseteq \mathcal{V} \times \mathcal{L}$ the set containing the user-layer combinations (i.e., tuples) in which a user is present in the corresponding layer. The set $E_{\mathcal{L}} \subseteq V_{\mathcal{L}} \times V_{\mathcal{L}}$ contains the *directed links* between user-layer tuples. Note that, when all layers refer to the same aspect or dimension (e.g., same temporal dimension, same group of interactions, etc.) like in our setting, the elements in $V_{\mathcal{L}}$ simply correspond to pairs $\langle$*node*, *layer*$\rangle$. We hence denote with $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$ the *multilayer network graph* with set of entities $\mathcal{V}$ and set of layers $\mathcal{L}$.

For every layer $L_i \in \mathcal{L}$, let $V_{L_i} = \bigcup_{v \in \mathcal{V}}(v, L_i)$ be the set of pairs expressing the occurrences of entities in the graph of L_i, and $E_{L_i} \subseteq V_{L_i} \times V_{L_i}$ be the set of edges in L_i; to simplify notations, we will also refer to V_{L_i} and E_{L_i} as V_i and E_i, respectively. Moreover, we will use notation $v \in V_i$ to indicate $v \in \mathcal{V}$ s.t. $\exists (v, L_i) \in V_i$. For any node $v \in V_i$, we denote with $N_i^{in}(v) = \{u \in V_i | ((u, L_i), (v, L_i)) \in E_i\}$ the set of nodes that are in-neighbors of v within the same layer of v, and with $N_{\neg i}^{in}(v) = \bigcup_{j \neq i}\{u \in V_j | (u, L_j), (v, L_j) \in E_j\}$ the set of nodes that are in-neighbors of v within any of the other layers. Analogous definitions hold for the out-neighbor sets $N_i^{out}(v)$ and $N_{\neg i}^{out}(v)$.

Note that while entities (i.e., elements of $\mathcal{V}$) are not required to participate in all layers, each entity has to appear in at least one layer. Moreover, the only inter-layer edges are regarded as "couplings" of nodes representing the same entity between different layers, according to a multiplex network representation. Table 3.1 summarizes main notations used throughout this chapter.

3.3.2 Identifying contributors in multilayer networks

When modeling asymmetric relations in OSNs, centrality is more intuitively used to refer to the social *endorsement* received by a user, which represents a statement of the way other users (e.g., followers) support one's activity in the network. In this regard, the well-known PageRank has been widely recognized as an effective method to rank users according to their influential activity (i.e., contribution) status.

As mentioned in Sect. 3.2, Multiplex PageRank (mpxPR) is an extension of PageRank to multilayer networks developed by Halu et al. in [104]. The basic idea underlying mpxPR is that, given two layers in a predetermined order, the PageRank score a node has in one layer affects the PageRank score of the node in another layer. The authors introduce two main definitions of multilayer PageRank. The first definition corresponds to a personalized PageRank in which the bias to one layer lies in the random teleportation towards any node in another layer. The method is dubbed "additive" as the effect of a layer L_i on another layer L_j is exerted by adding some value to the PageRank score the nodes have in L_j proportionally to the PageRank in L_i. The second method, dubbed "multiplicative", refers to the case in which the effect of layer L_i on layer L_j lies in multiplying the relevance a node receives from its in-neighbors in L_j by a factor proportional to the node's PageRank in L_i. The two types of bias can also be integrated to provide a "combined" variant of mpxPR.

notation	*description*
$\mathcal{V}$	Set of entities (i.e., users)
L; $\mathcal{L}$; ℓ	Layer; set of layers; number of layers
$V_{\mathcal{L}}$	Set of user-layer tuples (i.e., pairs)
$E_{\mathcal{L}}$	Set of directed links between user-layer tuples
$G_{\mathcal{L}}$	Multilayer network graph
V_{L_i} (or V_i)	Set of users in layer L_i
$N_i^{in}(v)$	Set of users that are in-neighbors of v within L_i
$N_{\neg i}^{in}(v)$	Set of users that are in-neighbors of v outside L_i
$N_i^{out}(v)$	Set of users that are out-neighbors of v within L_i
$N_{\neg i}^{out}(v)$	Set of users that are out-neighbors of v outside L_i
ω_i $(i = 1..\ell)$	Layer relevance coefficients
$\mathsf{mlLR}(v)$	*Multilayer LurkerRank* score of $v \in \mathcal{V}$
$\mathsf{mlALCR}(v)$	*Alternate Lurker-Contributor Ranking* score of $v \in \mathcal{V}_{\mathcal{L}}$
$\mathcal{R}_i^{lurk}(v)$ (or L-mlALCR)	Cross-layer lurking score of $v \in \mathcal{V}_{\mathcal{L}}$
$\mathcal{R}_i^{contrib}(v)$ (or C-mlALCR)	Cross-layer contributor score of $v \in \mathcal{V}_{\mathcal{L}}$
$\alpha_l, \alpha_c \in (0,1)$	Damping factors in the mlALCR method
$^{\alpha}\langle \mathrm{x} \rangle$	Setting $\alpha_l = \alpha_c = \mathrm{x}$, with $\mathrm{x} \in (0,1)$
(C, L)	Pair of C-mlALCR and L-mlALCR solutions
$name(L_i)$	Label of layer L_i
$C\_name(L_i)$	Projection of C-mlALCR to $name(L_i)$
$L\_name(L_i)$	Projection of L-mlALCR to $name(L_i)$
$\mathsf{mlALCR}\text{-Var}^L(v)$	Cross-layer variability of the deviations of L-mlALCR ranks of v from the median ranks
$\mathsf{mlALCR}\text{-Var}^C(v)$	Cross-layer variability of the deviations of C-mlALCR ranks of v from the median ranks
$\mathsf{mlALCR}_{max}(v)$, $\mathsf{mlALCR}_{min}(v)$, $\mathsf{mlALCR}_{med}(v)$	mlALCR score of v corresponding to its highest (max), lowest (min), or median (med) ranking position

TABLE 3.1: Main notations used in this paper

Although mpxPR was specifically designed for a duplex network, the authors also discuss an extension to the general case with more than two layers, by exploiting a cascade of single-layer PageRank instances (i.e., the PageRank on a layer contributes to the PageRank on the next layer in the ordering, in a recursive way).

3.3.3 LurkerRank and its extension to multilayer networks

In [97], [110], the authors have originally brought the concept of centrality in the context of lurking behavior analysis, with the goal of characterizing and ranking the users according to their degree of lurking in an OSN. Following an unsupervised learning paradigm, the lurker ranking models utilize only the topology information of an OSN (like PageRank and other classic ranking methods), which is seen as a directed graph where any edge (u, v) means that v is "consuming" or "receiving" information from u. Upon the assumption that lurking behaviors build on the *amount of information a node consumes*, the key intuition is that the strength of a user's lurking status can be determined based on three basic principles, which are here informally summarized as follows: *overconsumption* (i.e., the excess of information-consumption over information-production, estimated proportionally to the in/out-degree ratio of a node), the *authoritativeness of the information received from the in-neighbors*, and the *non-authoritativeness of the information produced, i.e., sent to the out-neighbors*. These principles form the basis for three ranking models, whereby a complete specification was provided in terms of PageRank and alpha-centrality based formulations. For the sake of brevity here, and throughout this chapter, we will refer to only one of the formulations described in [97], [110], which is that based on the *in-out-neighbors-driven lurker ranking*, hereinafter referred to as *LurkerRank* (LR).

Given a directed graph $G = (V, E)$, and a node $v \in V$, let $N^{in}(v)$ and $N^{out}(v)$ denote the set of in-neighbors and the set of out-neighbors of v, respectively. For any node v, the *LurkerRank* score $\mathsf{LR}(v)$ is defined as follows [97]:

$$\mathsf{LR}(v) = \alpha[\mathcal{R}^{\text{in}}(v)\ (1 + \mathcal{R}^{\text{out}}(v))] + (1 - \alpha)p(v) \tag{3.1}$$

where $\mathcal{R}^{\text{in}}(v)$ is the in-neighbors-driven lurking function:

$$\mathcal{R}^{\text{in}}(v) = \frac{1}{|N^{out}(v)|} \sum_{u \in N^{in}(v)} \frac{|N^{out}(u)|}{|N^{in}(u)|} \mathsf{LR}(u) \tag{3.2}$$

and $\mathcal{R}^{\text{out}}(v)$ is the out-neighbors-driven lurking function:

$$\mathcal{R}^{\text{out}}(v) = \frac{|N^{in}(v)|}{\sum_{u \in N^{out}(v)} |N^{in}(u)|} \sum_{u \in N^{out}(v)} \frac{|N^{in}(u)|}{|N^{out}(u)|} \mathsf{LR}(u) \tag{3.3}$$

Above, $p(v)$ is the value for v in the personalization vector, which is by default set to $1/|V|$, and α is a damping factor ranging within [0,1], usually set to 0.85. The ratio underlying the $\mathcal{R}^{\text{in}}(\cdot)$ model is that the score of a node increases with the number of its in-neighbors and with their likelihood of being non-lurkers (i.e., having relatively high out/in-degree); also, this model incorporates a factor that is inversely proportional to the node's out-degree, thus finally accounting for both the contribution of the node's in-neighbors and its own in/out-degree property. Concerning the $\mathcal{R}^{\text{out}}(\cdot)$ model, the lurking score of a node increases with the tendency of its out-neighbors of being lurkers; also this model incorporates a factor that penalizes the node's scores if it receives less than what its out-neighbors receive. Note also that the combination of the two models is such that the strength of non-lurking behavior of in-neighbors is dominant, which ensures a better fit of the hypothetical likelihood function for a given node, as demonstrated in [97]. The interested reader is referred to [97] for further details.

We propose here an adaptation of the original (i.e., monoplex) LurkerRank formulation to deal with multiple layers. Following the lead of [118], the key idea is to make the underlying random-walk model be expressed by as many transition probability matrices as the different layers. These matrices are then linearly combined, using a proper weighting scheme (e.g., weights following a probability distribution, or reflecting some user preference or knowledge on the relevance of the various layers).

Given a multilayer graph $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$, we define the *multilayer LurkerRank*, hereinafter denoted as mlLR, by the following system of equations, for all $v \in \mathcal{V}$:

$$\mathsf{mlLR}(v) = \alpha[\mathcal{R}^{\text{in}}(v)\ (1 + \mathcal{R}^{\text{out}}(v))] + (1-\alpha)p(v) \tag{3.4}$$

where $\mathcal{R}^{\text{in}}(v)$ is the multilayer in-neighbors-driven lurking function:

$$\mathcal{R}^{\text{in}}(v) = \sum_{L_i \in \mathcal{L}} \frac{w_i}{|N_i^{out}(v)|} \sum_{u \in N_i^{in}(v)} \frac{|N_i^{out}(u)|}{|N_i^{in}(u)|} \mathsf{mlLR}(u), \tag{3.5}$$

$\mathcal{R}^{\text{out}}(v)$ is the multilayer out-neighbors-driven lurking function:

$$\mathcal{R}^{\text{out}}(v) = \sum_{L_i \in \mathcal{L}} \frac{w_i |N_i^{in}(v)|}{\sum_{u \in N_i^{out}(v)} |N_i^{in}(u)|} \sum_{u \in N_i^{out}(v)} \frac{|N_i^{in}(u)|}{|N_i^{out}(u)|} \mathsf{mlLR}(u) \tag{3.6}$$

and the layer weighting scheme is specified by non-negative real-valued coefficients $\omega_1, \ldots, \omega_\ell$, such that $\sum_{L_i \in \mathcal{L}} \omega_i = 1$. Note that, with the exception of (i) the specification of layer in the summation terms and (ii) the layer-specific weighting coefficient (for the linear combination), the analytical forms Eq. 5 and Eq. 6 in mlLR are identical to those of the basic LurkerRank method, i.e., Eq. 2 and Eq. 3, respectively.

3.4 Multilayer Alternate Lurker-Contributor Ranking

The previously discussed methods offer proper solutions to problems of characterization of the user prominence, in terms of either lurking or contributor behaviors, that is somehow embedded in a multilayer network. These methods provide a useful opportunity for detecting properties related to the behavior of users that would otherwise remain unveiled if each of the layer networks was analyzed in isolation. However, they are not designed to identify and rank users according to opposite (i.e., lurking and contributor) behaviors they can alternately show across interconnected layer networks, which is the main problem under consideration in this work.

Motivating Example 1. *Consider the example multilayer social network shown in Fig. 3.1. We might suppose that layers correspond to three distinct OSNs to which users participate using their respective multiple accounts. We use symbols L_1, L_2, L_3 to denote the layers, from top to bottom. The meaning of edge (u, v) on any layer is that v endorses u, or consumes information produced by u (e.g., v follows u, or v likes/comments contents posted by u, etc.).*

At first, let us focus on simple topological characteristics, particularly on the ratio between in-neighbors and out-neighbors (hereinafter in/out-degree ratio) the nodes have on each of the layers. In layer L_1, users 2, 5 and, 6 appear to

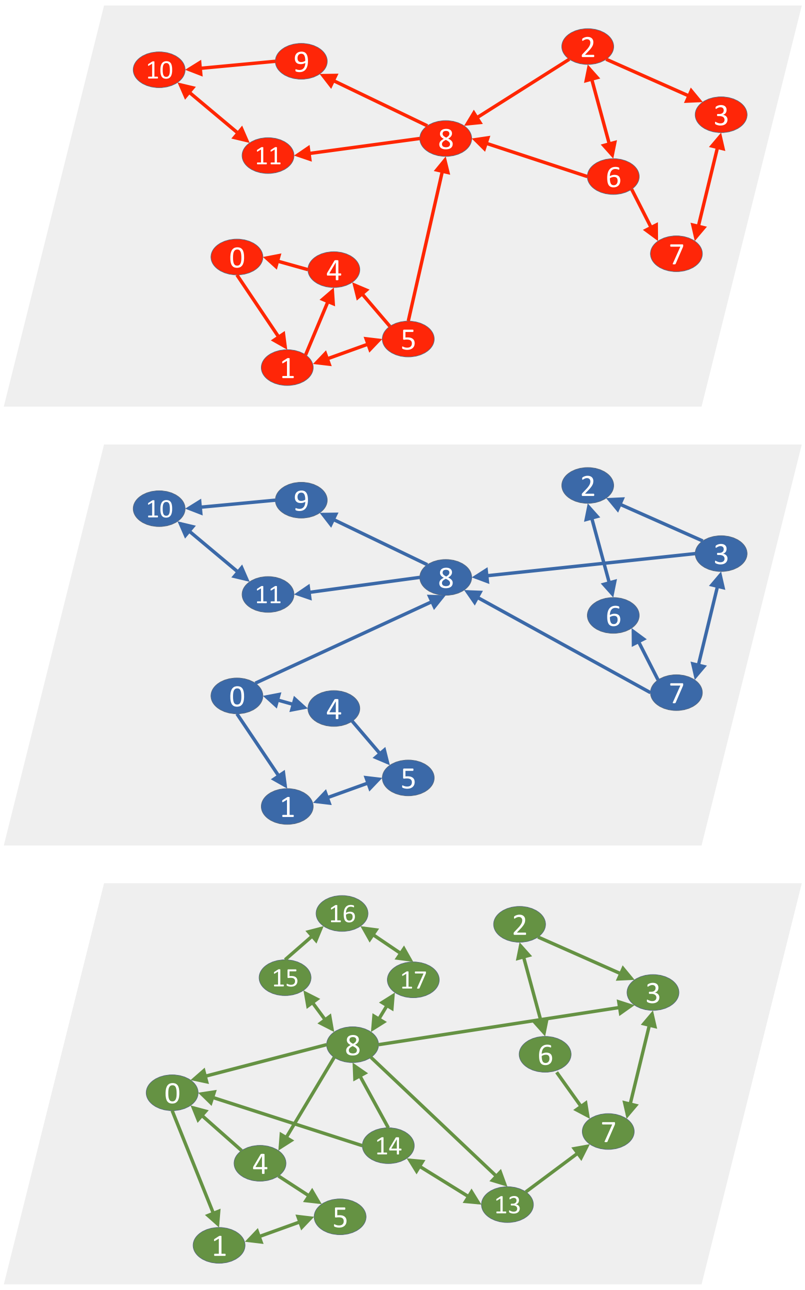


FIGURE 3.1: Example multilayer social network, which illustrates eighteen users located over three layer networks.

behave as contributors, since their in/out-degree ratio is lower than one (i.e., 0.33); by contrast, users 3, 4, 7, 10, 11 and 8 are potential lurkers since their in/out-degree is equal to 2. In layer L_2, users 1, 2, 5, 6, 10, 11 have the highest in/out-degree ratio (i.e., 2), so they are candidate lurkers in this specific layer. While users 0, 3, 7, 4 have the lowest ratio, and hence should be regarded as contributors. In layer L_3, candidate contributors are 13, 2, 4, 6, 14, 8 with in/out-degree ratio greater than one, while candidate lurkers are 0, 3, 7 with in/out-degree ratio equal to 3, and 1, 5, 15 with in/out-degree ratio equal to 2.

As previously discussed, there is however much more to be considered than a trivial in/out-degree ratio to effectively analyze lurking behaviors (and dually, *contributor behaviors), which further complicates in a multilayer network. For instance, user 8 receives information from two main components while providing limited information to a smaller component in L_1 and L_2, thus behaving as a lurker. By contrast, it is evident that the same user in L_3 acts as a major contributor by feeding various components; while not being the most active user in the layer (he is only the 6th if we consider the in/out degree ratio), user 8 might have the most distinctive alternate behavior in the multilayer network.*

Also, user 3 receives information from her/his membership component in layer L_1 (which includes a local top-contributor such as 2), while in layer L_3 is fed by users 2, 7, and above all by the top-contributor user 8. By contrast, in L_2, user 3 is top-contributor in her/his turn, as s/he receives information only from user 7 and feeds her/his component and user 8. Therefore, user 3 seems to show an important alternate behavior, which is strengthened by the effect of linkage to top-contributors/lurkers in all the three layers, and also nicely dovetails with the alternate behavior of user 8.

As another example, user 0 plays a marginal role in layer L_1, where she has only one outgoing link and one incoming link both limited to her/his component, and so s/he should not be identified either as a relevant lurker or contributor. Nevertheless, in the other layers s/he shows an alternate role: in layer L_3, s/he receives information from her/his own component, from user 8 and from previously unseen user 14; in layer L_2, s/he provides information to user 8 and towards her/his component.

We translate and generalize the requirements and intuitions underlying the above illustrated example to formulate a new ranking problem in multilayer networks, named *Alternate Lurker-Contributor Ranking.* In doing so, we adopt an *unsupervised learning* approach, based solely on structural information of the multilayer network. The reason behind this choice is that we do not want our method relies on any a-priori knowledge on influencing factors, topical expertise, or exogenous sources which, while contributing to some extent to forming the individual's behavior, they cannot easily be determined and measured.

Our basic assumption is that the lurker (resp. contributor) status of a user relies on the contributor (resp. lurker) status of some of her/his neighbors. Upon this, the key idea underlying the proposed approach is to determine the lurker and contributor behaviors of users in terms of two functions, each of which characterizes one role and is contextualized to a specific layer. The behavior to be identified is expected to be "alternated" over the various layers, in the sense that a user may act as a lurker (resp. contributor) in one layer while conversely behaving as a contributor (resp. lurker) in one or many of the other layers. Moreover, since the two behavioral properties are inter-related, we want that the scores determined by one of such functions depend on the scores by the other function.

Definition 1 (*Alternate Lurker-Contributor Ranking*)**.** *Let $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$ be a multilayer OSN graph, and let us consider any user in $\mathcal{V}$ that appears in a non-empty, non-singleton subset of $\mathcal{L}$. Let $v \in V_{\mathcal{L}}$ denote the particular instance of the user in a specific layer $L_i \in \mathcal{L}$. The alternate lurker-contributor behavior* *status of node v in $G_{\mathcal{L}}$ conditionally to its membership layer L_i follows two principles, each expressed by two mutually reinforcing terms:*

- *Principle 1 – Cross-layer lurker behavior, whose strength is proportional to the v's status as lurker in L_i, and to the v's status as contributor in a non-empty subset L' of layers other than L_i;*
- *Principle 2 – Cross-layer contributor behavior, whose strength is proportional to the the v's status as contributor in L_i, and to the v's status as lurker in a non-empty subset L'' of layers other than L_i;*

More formally, by denoting with ϕ^L and ϕ^C two scoring functions respectively for the lurker and contributor status of nodes conditionally to a given (set of) layers, we want to determine the ranking solutions $rank^L$ and $rank^C$, such that for any $v \in V_{\mathcal{L}}$ in $L_i \in \mathcal{L}$:

- *$rank^L(v)$ is higher by simultaneously increasing $\phi^L(v; L_i, \phi^C)$ and $\phi^C(v; L', \phi^L)$,*
- *$rank^C(v)$ is higher by simultaneously increasing $\phi^C(v; L_i, \phi^L)$ and $\phi^L(v; L'', \phi^C)$,*

where $L', L'' \subseteq \mathcal{L} \setminus \{L_i\}$.

Upon the above stated problem, we define the *Alternate Lurker-Contributor Ranking* method (mlALCR), which solves two mutually dependent systems of equations that are computed simultaneously (in the style of classic Kleinberg's HITS algorithm for the ranking of hubs and authorities in web pages [26]): one system determines the layer-specific lurking score and the other system determines the layer-specific contributor score for all nodes in the layers of the complex network. According to the principles stated in Def. 1, we devise each of the role scoring functions (i.e., equation systems) as a linear combination of two terms: the one measuring the user's behavior locally to his/her membership layer, and the other measuring the opposite behavior the same user might show externally to his/her membership layer.

Given a multilayer graph $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$, each layer is modeled as a directed graph according to the lurking-oriented direction of edges, i.e., (u, v) means that v endorses u by implicitly consuming/receiving information from u. For any node $v \in V_{\mathcal{L}}$ located in layer $L_i \in \mathcal{L}$, the *cross-layer lurking score* of v w.r.t. L_i, denoted as $\mathcal{R}_i^{lurk}(v)$, is determined by two terms: the one is (i) proportional to the number of v's in-neighbors and their status as cross-layer contributor in L_i, and (ii) inversely proportional to the number of v's out-neighbors in L_i; the second term is (i) proportional to the number of v's out-neighbors and their status as cross-layer lurker in each of the remaining layers L_j (with $j \neq i$), and (ii) inversely proportional to the number of v's in-neighbors in each L_j.

$$\mathcal{R}_i^{lurk}(v) = \underbrace{\alpha_l \frac{\omega_i}{|N_i^{out}(v)|} \sum_{u \in N_i^{in}(v)} \frac{\mathcal{R}_i^{contrib}(u)}{|N_{\neg i}^{out}(u)|}}_{v \text{ as } \textit{lurker} \text{ in } L_i} +$$

$$\underbrace{(1-\alpha_l) \sum_{L_j \in \mathcal{L}} \left[\frac{\omega_j}{|N_j^{in}(v)|} \sum_{u \in N_j^{out}(v)} \frac{\mathcal{R}_j^{lurk}(u)}{|N_i^{in}(u)|} \right]}_{v \text{ as } \textit{contributor} \text{ in } L_j, \text{ with } j \neq i} \quad (3.7)$$

Dually, the *cross-layer contributor score* of v w.r.t. L_i, denoted as $\mathcal{R}_i^{contrib}(v)$, is determined by two terms: the one is (i) proportional to the number of v's out-neighbors and their status as cross-layer lurker in L_i, and (ii) inversely proportional to the number of v's in-neighbors in L_i; the second term is (i) proportional to the number of v's in-neighbors and their status as cross-layer contributor in each of the remaining layers L_j (with $j \neq i$), and (ii) inversely proportional to the number of v's out-neighbors in each L_j.

$$\mathcal{R}_i^{contrib}(v) = \underbrace{\alpha_c \frac{\omega_i}{|N_i^{in}(v)|} \sum_{u \in N_i^{out}(v)} \frac{\mathcal{R}_i^{lurk}(u)}{|N_{\neg i}^{in}(u)|}}_{v \text{ as } \textit{contributor} \text{ in } L_i} +$$

$$\underbrace{(1-\alpha_c) \sum_{L_j \in \mathcal{L}} \left[\frac{\omega_j}{|N_j^{out}(v)|} \sum_{u \in N_j^{in}(v)} \frac{\mathcal{R}_j^{contrib}(u)}{|N_i^{out}(u)|} \right]}_{v \text{ as } \textit{lurker} \text{ in } L_j, \text{ with } j \neq i} \quad (3.8)$$

In both Eqs. 3.7–3.8, $\alpha_l, \alpha_c \in (0,1)$ are damping factors that control the contribution of the "within-layer" behavior against the "outside-layer" opposite behavior. Moreover, ω_i (with $L_i \in \mathcal{L}$) are non-negative coefficients, such that $\sum_{L_i \in \mathcal{L}} \omega_i = 1$, which embed some relevance scheme that might be assigned to the various layers in the network. Note that, in order to avoid divergence of values and hence guarantee convergence of the algorithm, both $\mathcal{R}^{lurk}$ and $\mathcal{R}^{contrib}$ score vectors are normalized after each iteration.

3.5 Evaluation

3.5.1 mlALCR settings and notations

The damping factors α_l and α_c will be varied in the set of values $\{0.5, 0.85\}$; specifically, we will present results that correspond to the following settings:

$\alpha_l = \alpha_c = 0.5$, balanced setting, used as default; $\alpha_l = \alpha_c = 0.85$, which weights more the user's behavior locally to the layer graph in which s/he is located. We will use shortcut ${}^{\alpha}\langle \mathrm{x} \rangle$ to indicate the setting $\alpha_l = \alpha_c = \mathrm{x}$; for example, ${}^{\alpha}\langle 0.85 \rangle$ corresponds to $\alpha_l = \alpha_c = 0.85$.

As concerns the layer weighting scheme in mlALCR, we consider two cases: *uniform weighting*, used as default, and proportional to the edge-set size of each layer, i.e., $\omega_i = |E_i|/|E_{\mathcal{L}}|$ (with $L_i \in \mathcal{L}$), also referred to as *layer-proportional weighting*.

We will use prefixes C- and L- to denote the mlALCR contributor ranking (Eq. 3.8) and the lurker ranking (Eq. 3.7) solution, respectively. To refer to a pair of C-solutions and L-solutions, for any given setting of mlALCR, we will also use short notation (C, L). Moreover, we will denote with $C\_name(L_i)$ (resp. $L\_name(L_i)$) the projection of the contributor (resp. lurker) mlALCR ranking solution to a particular layer L_i labeled as $name(L_i)$, i.e., the selection of a ranking solution corresponding to nodes that belong to L_i.

Our proposed mlALCR is computed using a power iteration method, for which we set up the termination threshold for score convergence as 1.0e-5.

3.5.2 Competing methods

We compare our proposed mlALCR with five methods: the Multiplex PageRank (mpxPR), the basic (i.e., monoplex) LurkerRank (LR), its extension to multilayer networks (mlLR) proposed in this work, and two ranking aggregation methods, originally presented here as well. Methods mpxPR, LR, and mlLR have been discussed in Sect. 3.3; as concerns their experimental setting, the damping factors α will be set to the default value, i.e., 0.85. In the following we describe the proposed ranking aggregation methods.

LR-aggregation methods. We define two ranking aggregation methods that leverage on the ranking solutions provided by the basic LR on every layer. These methods are designed to manipulate the rank assigned by LR to every node of each layer, in order to account for the heterogeneous distribution of the LR ranks of a node over the layers. We introduce notation $rank_i(v)$ to denote the rank (not the ranking score) assigned by LR to node v in layer L_i.

The first method, hereinafter denoted as $\mathsf{LRa1}_i$ (with $L_i \in \mathcal{L}$), computes a score for every node $v \in V_{\mathcal{L}}$ in L_i, by summing two terms: the first increasing for higher rank (i.e., lower values of *rank* function) of the node in L_i, and the second increasing for lower rank of the node in the other layers ($L_j \neq L_i$) in which it appears:

$$\mathsf{LRa1}_i(v) \quad = \quad \frac{|V_i| - rank_i(v) + 1}{|V_i|} \quad + \quad \frac{1}{|\mathcal{L}| - 1} \sum_{L_j,\ j \neq i} \frac{rank_j(v)}{|V_j|} \tag{3.9}$$

The second method, hereinafter denoted as LRa2, computes a global score for every $v \in \mathcal{V}$ appearing in two or more layers. It expresses the variability over the layers of the deviations of the LR rank of v from the median rank:

$$\mathsf{LRa2}(v) = \mathrm{Var}_{L_i \in \mathcal{L}} \left(\frac{rank_i(v) - |V_i|/2}{|V_i|} \right) \tag{3.10}$$

where Var is the variance operator.

3.5.3 Assessment criteria

Significance of mlALCR. We exploit the analytical form of Eq. 3.10 to also determine the variability of the L-mlALCR ranks and C-mlALCR ranks that a node has over the layers of a network. Specifically, for each $v \in \mathcal{V}$ appearing in two or more layers, we compute the cross-layer variability of the deviations of L-mlALCR ranks of v from the median ranks as:

$$\mathsf{mlALCR}\text{-Var}^L(v) = \mathrm{Var}_{L_i \in \mathcal{L}} \left(\frac{rank_i^L(v) - |V_i|/2}{|V_i|} \right) \tag{3.11}$$

where $rank_i^L(v)$ is the v's rank in the projection to layer L_i of the L-mlALCR solution; analogously, for the contributor role, we define function mlALCR-Var$^C(v)$.

We use the above measures to characterize the significance of the alternate lurker-contributor behavior of each user so that the higher the value of mlALCR-Var$^L(v)$ (resp. mlALCR-Var$^C(v)$), the stronger the cross-layer lurker (resp. contributor) behavior of the node.

Comparison with competing methods. To evaluate mlALCR against competing methods, we used two well-known ranking assessment criteria, namely *Kendall rank correlation coefficient* [119] and *Fagin's intersection metric* [120].

Kendall correlation evaluates the similarity between two rankings, expressed as sets of ordered pairs, based on the number of inversions of pairs which are needed to transform one ranking into the other. Formally: $\tau(\mathcal{R}', \mathcal{R}'') = 1 - \frac{2\Delta(\mathcal{P}(\mathcal{R}'),\mathcal{P}(\mathcal{R}''))}{M(M-1)}$, where $\mathcal{R}'$ and $\mathcal{R}''$ are the two rankings to be compared, $M = |\mathcal{R}'| = |\mathcal{R}''|$ and $\Delta(\mathcal{P}(\mathcal{R}'), \mathcal{P}(\mathcal{R}''))$ is the symmetric difference distance between the two rankings, calculated as the number of unshared pairs between the two lists. The score returned by τ is in the interval $[-1, 1]$, where a value of 1 means that the two rankings are identical and a value of -1 means that one ranking is the reverse of the other.

Fagin measure determines how well two ranking lists are in agreement with each other, by accounting for the problem of comparing "partial rankings", i.e., assuming that elements in one list may not be present in the other list. Moreover, it expresses top-weightedness, i.e., the top of the list gets higher weight than the tail. Given two k-size lists $\mathcal{R}', \mathcal{R}''$, the Fagin score is defined as: $F(\mathcal{R}', \mathcal{R}'', k) = (1/k) \sum_{q=1}^{k} (|\mathcal{R}'_{:q} \cap \mathcal{R}''_{:q}|)/q$, where $\mathcal{R}_{:q}$ denotes the sets of nodes from the 1st to the qth position in the ranking. Therefore, F is the average over the sum of the weighted overlaps based on the first k nodes in both rankings. F is within $[0, 1]$, whereby values closer to 1 correspond to better scores. For the experiments discussed in the following, we setup k to 100, for short denoted as Fagin-100.

Note that all methods under consideration in this work compute a *local* (i.e., layer-specific) score for any node in $V_{\mathcal{L}}$, whereas LRa2 and mlLR compute a *global* score for any element of $\mathcal{V}$. To enable comparison of the latter methods with mlALCR, we will evaluate the status of any $v \in \mathcal{V}$ in the ranking of LRa2 or mlLR through three cases: the status of v corresponding to its highest (max), lowest (min), or median (med) ranking position occurring in a solution provided by mlALCR. We will use notations mlALCR_{max}, mlALCR_{min}, mlALCR_{med} to denote the three different cases.

3.5.4 Datasets

We used four real-world multilayer OSNs, which correspond to three different categories, as described next.[1]

Cross-platform, user multi-relational networks. We used two datasets in which the graphs of different OSNs are modeled as layers: *FF-TW-YT* (stands for FriendFeed, Twitter, and YouTube) [82] and *GH-SO-TW* (stands for GitHub, StackOverflow and Twitter) [121]. The former was built by exploiting the feature of FriendFeed as social media aggregator to align registered users who were also members of Twitter and YouTube. The latter is a dataset originally built to characterize specialist expertise of users engaged in Web-mediated professional activities. Inter-layer edges link the same user across the different networks. Within-layer edges express followships (i.e., “who follows whom”), with the exception of StackOverflow in *GH-SO-TW* which is a help network built around questions and answers of the users, hence the meaning of edges is “who answers to whom”.

(Single-platform) User multi-relational network. *HiggsTW* is a Twitter dataset built through 2012 to collect user interactions about the event of the announcement of the discovery of the Higgs boson [122]. The meaning of the four layers is “who retweets/mentions/replies to/follows whom”, respectively.

(Single-platform) Temporal multi-slice network. *Flickr* refers to the Flickr dataset studied in [123], [124] which contains time information about the favorite markings assigned to the uploaded photos. We used the corresponding timestamped interaction network whose links express “who puts a favorite-marking to a photo of whom”. We extracted the layers on a month-basis and aggregated every six (or more) months to achieve a quite balanced multilayer network.[2]

Table 3.2 reports for each dataset, the size of set $\mathcal{V}$, the number of edges in all layers, the description of layers along with the percentage of nodes in each particular layer, the average coverage of node set (i.e., $1/|\mathcal{L}|\sum_{L_i \in \mathcal{L}}(|V_i|/|\mathcal{V}|)$), and the average coverage of edge set (i.e., $1/|\mathcal{L}|\sum_{L_i \in \mathcal{L}}(|E_i|/\sum_{L_i}|E_i|)$).

We also calculated basic, monoplex structural statistics, such as in-degree, average path length, and clustering coefficient, for the layer networks of each

[1] Our evaluation datasets can be retrieved from the web page supporting this manuscript, available at http://people.dimes.unical.it/andreatagarelli/mlalcr/.

[2] We point out that, though some interpretability issue may arise about mutual dependence underlying the time slices extracted from the same network, here we assume not to analyze the system as a time-evolving network, rather as comprised of time steps which are all observed as a whole at the last time available in the network.

dataset	**#nodes**	**#edges**	**layer notation and description (w/ % nodes)**	**node set coverage**	**edge set coverage**
GH-SO-TW	55 140	1 551 842	gh: Github (68.6%) so: StackOverflow (40.8%) tw: Twitter (94.5%)	0.68	0.36
FF-TW-YT	6 407	106 299	ff: FriendFeed (86.5%) tw: Twitter (89%) yt: YouTube (10.4%)	0.62	0.40
HiggsTW	456 626	15 367 315	fw: followship (100%) mt: mention (25.5%) rp: reply (8.5%) rt: retweet (56.2%)	0.48	0.26
Flickr	789 019	17 071 325	t1: Aug 2004-Mar 2005 (4.6%) t2: Apr-Sep 2005 (14.1%) t3: Oct 2005-Mar 2006 (26.9%) t4: Apr-Sep 2006 (49.3%) t5: Oct 2006-Mar 2007 (69.2%)	0.33	0.22

TABLE 3.2: Main characteristics of evaluation datasets

dataset (results not shown). Notably, according to all statistics, we observed much higher structural similarity between the layers in *Flickr* (e.g., clustering coefficient and average path length consistently around 0.03–0.04 and 3.8–3.9, respectively) than between the layers in each of the other datasets.

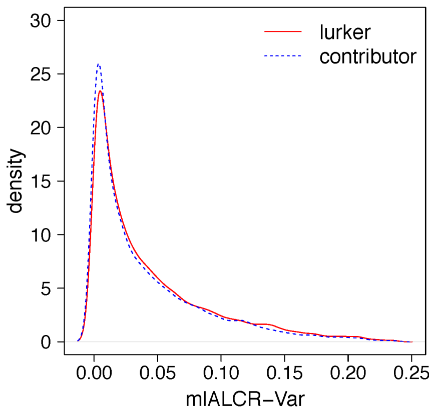
Graph representation. As discussed in Sect. 3.4, in order to directly input a multilayer network graph to our mlALCR, the orientation of edges is modeled according to the information that flows *from the producer to the consumer*. Therefore, we inverted the original orientation of the edges in each layer and dataset, so that an edge (u, v) means that v follows/likes/retweets u; one exception is for *HiggsTW* where we left the original orientation in layers *reply* and *mention*, i.e., (u, v) means u replies to/mentions v: the rationale here is that u is required to be actively involved in producing content for replying to/mentioning v, which in turn due to causality is stimulated to consume this content. We advocate that, in general, this may apply to any type of user interaction that is more likely to express both production and consumption of information in a bidirectional way, than least-effort relations/interactions like followships, liking, and retweeting.

3.6 Results

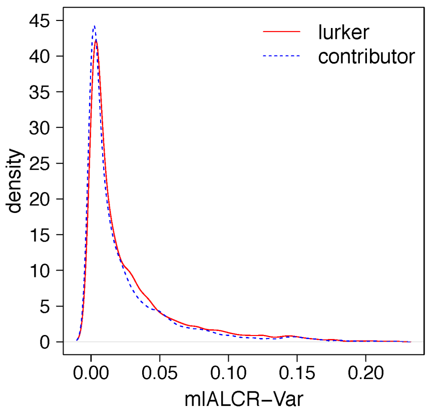
We organize the presentation of experimental results according to the following evaluation goals:

- *Significance of mlALCR.* We determine the strength of the alternate lurker-contributor behavior of users, and investigate how this is related to the different per-layer positions that users have in the ranking provided by mlALCR for each role (Sect. 3.6.1). We also investigate attachment distributions of lurkers versus contributors, and vice versa (Sect. 3.6.1).

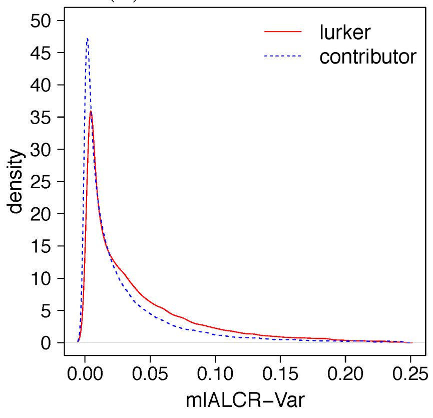
- *Analysis of the performance of mlALCR.* We study the dichotomy of the lurker and contributor ranking solutions provided by mlALCR, through an extensive analysis of correlation (Sect. 3.6.1). We also assess the effect of



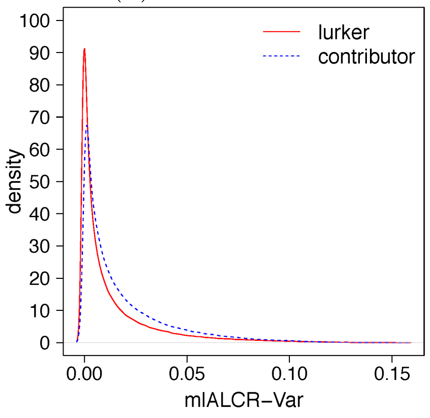
(a) *GH-SO-TW*



(b) *FF-TW-YT*



(c) *HiggsTW*



(d) *Flickr*

FIGURE 3.2: Density distributions of mlALCR-Var^L and mlALCR-Var^C on the various network datasets. *(Best viewed in color version, available in electronic format)*

the various configurations of mlALCR on its performance and convergence aspects (Sect. 3.6.1).

- *Meaningfulness of the solutions provided by mlALCR.* We investigate how the actual behavior of mlALCR reflects the theoretically expected one by manually inspecting the online profile of the top-ranked users with alternate lurker-contributor behavior (Sect. 3.6.1).

- *Uniqueness of mlALCR.* We present different stages of comparative evaluation with other methods (i.e., LR, LR-aggregation methods, mlLR, and mpxPR), in order to demonstrate the unique ability of mlALCR to determine alternate behaviors of lurking and active participation in multilayer networks (Sect. 3.6.2).

3.6.1 Evaluation of mlALCR

Distribution of cross-layer rank variability

We analyzed the density distributions of mlALCR-Var^L and mlALCR-Var^C, which are shown in Fig. 3.2. (Results correspond to the default setting of mlALCR, i.e.,

${}^{\alpha}\langle 0.5\rangle$ and uniform weighting). As we expected, all distributions are positively skewed, meaning that highest density corresponds to users which show small or negligible variability in their cross-layer lurker/contributor behavior. While no particular difference in shape occurs between per-dataset mlALCR-Var^L and mlALCR-Var^C distributions, the densities are quite different over the datasets, with *GH-SO-TW* and *Flickr* corresponding to the least and most skewness/peakedness, respectively. More interestingly, all distributions are also found to be heavy-tailed (leptokurtic), with range up to around 0.25, which confirms the presence of users with significant variability in their cross-layer lurker/contributor behavior.

Moreover, we gained an insight into the users corresponding to the head and tail of the distributions, by analyzing the quartile position that a user has in each *layer-specific projection* of the L-mlALCR and C-mlALCR solutions. The rationale here is that the higher the distance between the layer-specific quartiles the same user has in the L-mlALCR (resp. C-mlALCR) solution, the stronger the cross-layer lurker (resp. contributor) behavior of the user. In particular, for each role and corresponding mlALCR-Var distribution, we expect that users in the tail are likely to alternate their rank in distant positions (e.g., the 3rd and 1st quartiles) of the layer-specific projections of mlALCR; at the same time, we expect that users with low value of mlALCR-Var^L (resp. mlALCR-Var^C) tend to be ranked more uniformly, possibly over the same *and* lower quartile of L-mlALCR (resp. C-mlALCR).

Results have fully confirmed our above hypothesis, since in all datasets the top-ranked users (located in the 3rd quartile) in both mlALCR-Var^L and mlALCR-Var^C have always an alternate behavior; in particular, considering the fraction of those users who are ranked alternately among the 3rd and 1st quartiles of the layer-specific projections of mlALCR, *GH-SO-TW* offers the highest percentage of users with such strong alternate behavior (14.5%), followed by *HiggsTW* (8.2%), *FF-TW-YT* (6.1%), and *Flickr* (4.9%). Moreover, the lower quartile of mlALCR-Var^L (resp. mlALCR-Var^C) only contains users who are located over either the 1st or 2nd quartile of L-mlALCR (resp. C-mlALCR).

Attachment distributions

We discuss here our evaluation of the *attachment* between lurkers and contributors. We conducted two stages of analysis of the attachment distributions of lurkers and contributors based on the solutions provided by mlALCR. The two analyses mainly differ from each other in the way we selected the set of lurkers and the set of contributors under examination.

- In the first stage of analysis, we have kept one layer fixed at a time, thus considering *within-layer attachment*. To this purpose, we selected the same portion (i.e., 25%) of the head from the layer-specific projections of L-mlALCR and C-mlALCR in order to choose the set of top-ranked lurkers and the set of top-ranked contributors, respectively. Then, we calculated two distributions for every layer: (i) complementary cumulative distribution of the number of contributors followed by lurkers, and (ii)

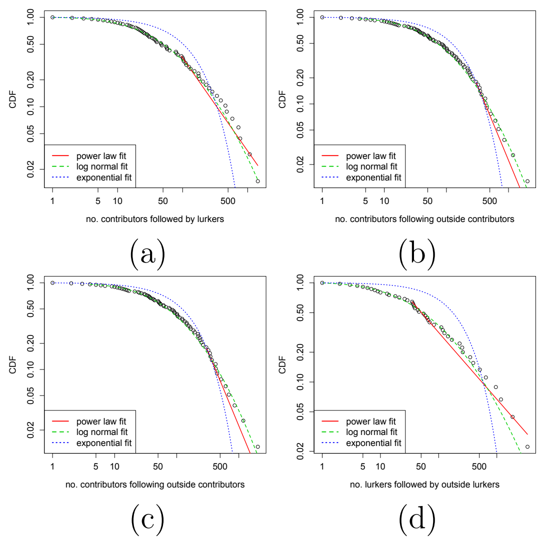


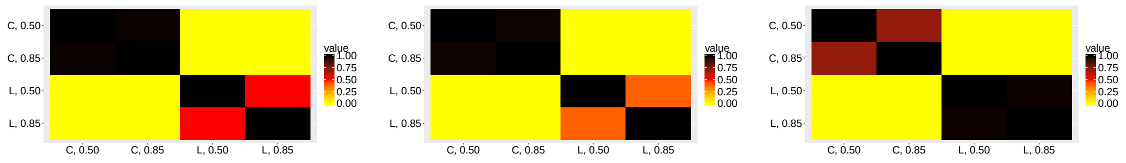
FIGURE 3.3: Complementary cumulative distributions of lurker-contributor attachment, on *GH-SO-TW*: (a)-(b) within-layer attachment in *Twitter* and (c)-(d) cross-layer attachment in *GitHub*. *(Best viewed in color version, available in electronic format)*

complementary cumulative distribution of the number of lurkers following contributors within-layer the same layer.

- In the second stage of analysis, we have instead considered *cross-layer attachment*. For a given layer L_i, we selected the top-25% of users ranked in the projection of L-mlALCR and considered their linkage, in the graphs of each of the remaining layers L_j (with $i \neq j$), with users appearing in the top-25% of the projection of L-mlALCR for each of layers L_j. We repeated the same selection process for the C-mlALCR solutions. Then, we calculated two distributions for every layer: (i) complementary cumulative distribution of the number of contributors following outside contributors, and (ii) complementary cumulative distribution of the number of lurkers followed by outside lurkers. Note that the orientation of the followship relation is here explained since, according to the logic of mlALCR, a top-ranked contributor in a layer is expected to behave as a lurker in the other layers, and vice versa.

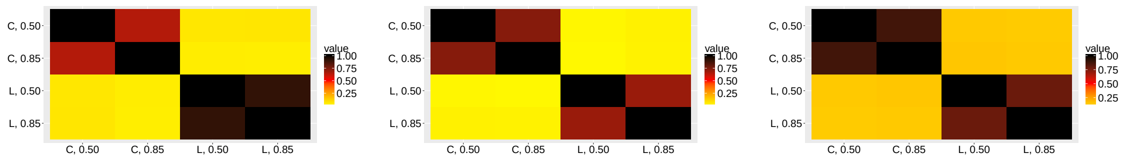
Afterwards, for each of the two stages, we tried to fit (discrete) power-law, log-normal and exponential distributions to each of the observed data, via maximum likelihood estimation.

Figure 3.3 shows part of the results obtained on *GH-SO-TW*, which are however indicative of the general trends we observed in both cases of attachment, for all layers and datasets. A first evident result is that the exponential distribution is definitely not appropriate for the analyzed data. The power-law distribution can fairly fit the data in many cases, although only for relatively high regimes of the values; the Kolmogorov-Smirnov test indeed confirmed the statistical significance of fitting only for high values of the minimal threshold



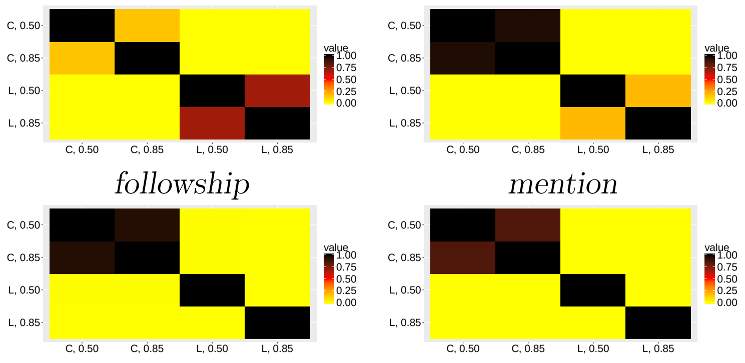
GitHub *StackOverflow* *Twitter*

(a) *GH-SO-TW*



FriendFeed *Twitter* *YouTube*

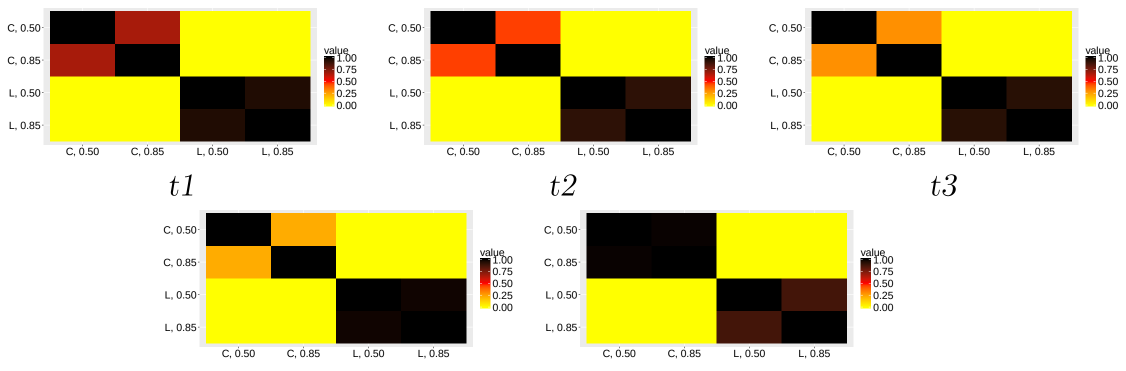
(b) *FF-TW-YT*



followship *mention*

reply *retweet*

(c) *HiggsTW*



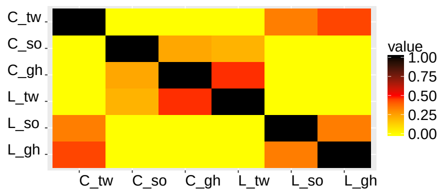
t1 *t2* *t3*

t4 *t5*

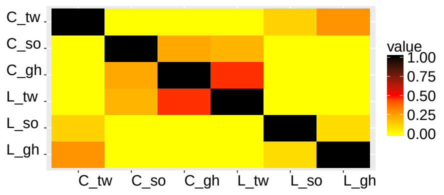
(d) *Flickr*

FIGURE 3.4: Per-layer evaluation of (C, L)-solutions, for each setting of αs, with uniform weighting scheme: Fagin-100 intersection heatmaps on the various layers of *GH-SO-TW*, *FF-TW-YT*, *HiggsTW*, and *Flickr* (from top to bottom). *(Best viewed in color version, available in electronic format)*

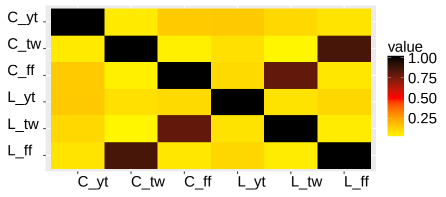
for fitting, x_{min}; for instance, considering the plots in Fig. 3.3, the following pairs (x_{min}, γ) were computed, with γ here denoting the exponent of the fitted power-law distribution: (a) (96, 2.04), (b) (74, 1.94), (c) (300, 2.73), and (d)



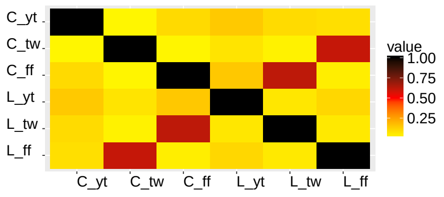
(a) *GH-SO-TW*



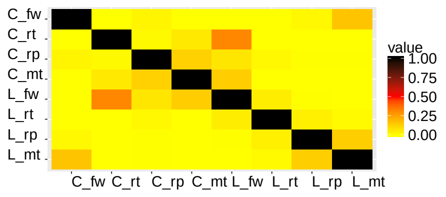
(b) *GH-SO-TW*



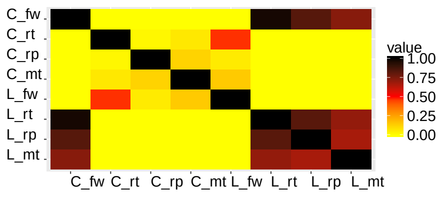
(c) *FF-TW-YT*



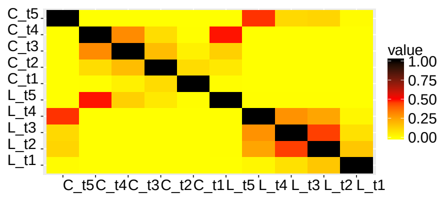
(d) *FF-TW-YT*



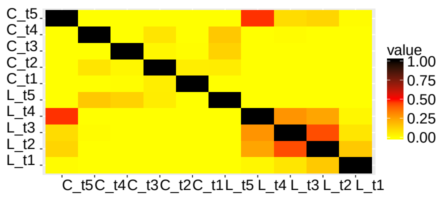
(e) *HiggsTW*



(f) *HiggsTW*



(g) *Flickr*



(h) *Flickr*

FIGURE 3.5: Per-αs-setting evaluation of (C, L)-solutions, for each layer, with uniform weighting scheme: Fagin-100 intersection heatmaps corresponding to (left) ${}^{\alpha}\langle 0.5\rangle$ and (right) ${}^{\alpha}\langle 0.85\rangle$. *(Best viewed in color version, available in electronic format)*

(35, 1.65). By contrast, as we observe from the plots, a much more reasonable fit was provided by log-normal distributions, which is also confirmed by the Shapiro-Wilk test of (log-)normality on our data.

Role correlation analysis

We analyzed how the 2-role mlALCR ranking solutions are correlated to each other. While varying the setting of damping factors, we first measured correlation at a layer level, i.e., between the *layer-specific projections* of the lurker and contributor mlALCR solutions obtained on each dataset; for this analysis, we present results corresponding to uniform weighting of layers. Then, we extended our analysis to the *whole* C-mlALCR ranking and L-mlALCR ranking solutions (i.e., ranking lists over all nodes and layers, ordered by decreasing score).

Fagin-100 / Kendall	L, ${}^{\alpha}\langle 0.5\rangle$ u-weighted	C, ${}^{\alpha}\langle 0.5\rangle$ u-weighted	L, ${}^{\alpha}\langle 0.85\rangle$ u-weighted	C, ${}^{\alpha}\langle 0.85\rangle$ u-weighted	L, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	C, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	L, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted	C, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted
L, ${}^{\alpha}\langle 0.5\rangle$, u-weighted	1.000	0.000	0.892	0.000	0.892	0.000	0.891	0.000
C, ${}^{\alpha}\langle 0.5\rangle$, u-weighted	-0.044	1.000	0.000	0.878	0.000	0.808	0.000	0.878
L, ${}^{\alpha}\langle 0.85\rangle$, u-weighted	0.731	-0.129	1.000	0.000	1.000	0.000	1.000	0.000
C, ${}^{\alpha}\langle 0.85\rangle$, u-weighted	-0.051	0.873	-0.120	1.000	0.000	0.691	0.000	0.990
L, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	0.535	-0.206	0.659	-0.172	1.000	0.000	1.000	0.000
C, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	-0.072	0.677	-0.210	0.638	-0.382	1.000	0.000	0.696
L, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted	0.637	-0.173	0.837	-0.159	0.671	-0.271	1.000	0.000
C, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted	-0.053	0.749	-0.117	0.865	-0.161	0.647	-0.155	1.000

Fagin-100 / Kendall	L, ${}^{\alpha}\langle 0.5\rangle$ u-weighted	C, ${}^{\alpha}\langle 0.5\rangle$ u-weighted	L, ${}^{\alpha}\langle 0.85\rangle$ u-weighted	C, ${}^{\alpha}\langle 0.85\rangle$ u-weighted	L, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	C, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	L, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted	C, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted
L, ${}^{\alpha}\langle 0.5\rangle$, u-weighted	1.000	0.000	0.653	0.000	0.672	0.000	0.665	0.000
C, ${}^{\alpha}\langle 0.5\rangle$, u-weighted	-0.361	1.000	0.000	0.413	0.000	0.934	0.000	0.434
L, ${}^{\alpha}\langle 0.85\rangle$, u-weighted	0.732	-0.354	1.000	0.000	0.808	0.000	0.901	0.000
C, ${}^{\alpha}\langle 0.85\rangle$, u-weighted	-0.320	0.652	-0.321	1.000	0.000	0.413	0.000	0.977
L, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	0.834	-0.466	0.745	-0.395	1.000	0.000	0.906	0.000
C, ${}^{\alpha}\langle 0.5\rangle$, lp-weighted	-0.372	0.942	-0.366	0.644	-0.480	1.000	0.000	0.434
L, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted	0.625	-0.419	0.858	-0.370	0.729	-0.434	1.000	0.000
C, ${}^{\alpha}\langle 0.85\rangle$, lp-weighted	-0.323	0.626	-0.334	0.942	-0.395	0.641	-0.383	1.000

TABLE 3.3: Kendall correlation and Fagin-100 intersection on whole mlALCR ranking solutions obtained on *GH-SO-TW* (top) and *HiggsTW* (bottom). (Prefixes "u" and "lp" stand respectively for uniform weighting scheme and layer-proportional weighting scheme.)

Per-layer correlation. By varying the setting of αs and considering one layer at a time, we first evaluated Kendall correlation between pairs of layer-projected (C, L)-solutions (results not shown). On all layers of *GH-SO-TW*, layers *mention* and *followship* of *HiggsTW*, and layer *t5* of *Flickr*, results are equal to or very close to zero, thus indicating total lack of correlation between the rankings of the two roles. In the remaining cases, Kendall correlation tends to be higher, up to values in the range 0.2–0.3 for *FF-TW-YT* and layers *t1*–*t4* in *Flickr*. Note however that Kendall correlation does not discriminate the head from the tail of the rankings being compared; furthermore, the above contingencies actually correspond to the presence of one or more layers that are clearly under-dimensioned and/or have topological similarity (e.g., *YouTube* in *FF-TW-YT*, early time slices in *Flickr*) with respect to the others in the network (cf. Sect. 3.5.4).

To support the above intuition and verify that any degree of correlation does not regard the head of rankings, we analyzed the top-ranked users from each of the layer-projected (C, L)-solutions in terms of Fagin intersection (Fig. 3.4). As we hypothesized, results show zero-correlation between any pair of (C, L)-solutions on *GH-SO-TW*, *HiggsTW* and *Flickr* for all layers and both settings of αs, while on *FF-TW-YT* Fagin-100 values are positive (0.14) only for the under-dimensioned layer *YouTube*.

Per-αs-setting correlation. In addition to the previous analysis, we studied the correlation between layer-projected (C, L)-solutions by varying the layers and considering one setting of the αs at a time. To indicate the layers of a dataset network, we will refer to their short notation (reported in Table 3.1). We discuss here Fagin-100 values, shown in Fig. 3.5.

At a first glance, by comparing the plots corresponding to ${}^{\alpha}\langle 0.5\rangle$ with those corresponding to ${}^{\alpha}\langle 0.85\rangle$, we observe few differences from C-solutions vs. L-solutions. More in detail, on *GH-SO-TW*, some differences occur between C_tw and L_gh (0.44 for ${}^{\alpha}\langle 0.5\rangle$ and 0.28 for ${}^{\alpha}\langle 0.85\rangle$), and between C_tw and L_so (0.36 for ${}^{\alpha}\langle 0.5\rangle$ and 0.13 for ${}^{\alpha}\langle 0.85\rangle$). Interestingly, the peak value (0.47) between C_gh and L_tw is yielded for both settings of αs. On the other cross-platform dataset, *FF-TW-YT*, high Fagin intersection corresponds to C_ff vs. L_tw (0.80 for ${}^{\alpha}\langle 0.5\rangle$ and 0.62 for ${}^{\alpha}\langle 0.85\rangle$) and to C_tw vs. L_ff (0.86 for ${}^{\alpha}\langle 0.5\rangle$ and 0.61 for ${}^{\alpha}\langle 0.85\rangle$). On *Flickr*, Fagin-100 values between (C, L)-pairs of different layers tend to be above zero mostly for temporally consecutive pairs of layers, with peaks between C_t4 and L_t5 (0.49, for ${}^{\alpha}\langle 0.5\rangle$), and between L_t4 and C_t5 (0.47). Considering *HiggsTW*, for ${}^{\alpha}\langle 0.5\rangle$, significant correlation occurs between L_fw and C_rt (0.31), and between C_fw and L_mt (0.16). These are further strengthened for ${}^{\alpha}\langle 0.85\rangle$: (i) 0.47 between L_fw and C_rt, whereas (ii) C_fw reveals to be highly correlated also with L_rt (0.97) and L_rp (0.82), besides with L_mt (0.74). The former correlation can reasonably be explained based on a *least-effort economy principle* of user interactional behavior: user tend to adopt the retweeting behavior more than producing contents (on which mentioning other users), in line with the intuition that a greater effort (in terms of time and communication economy) must be done to drive the social interaction towards active participation. The latter correlations instead would hint for a strong characterization of users that are mainly active as producers of social content (which explains their granted endorsement) from those who transfer others' information.

Whole ranking correlation. In a further stage of analysis, we studied ranking correlation by considering the whole solutions provided by mlALCR for each role and configuration of parameters. Table 3.3 summarizes Kendall and Fagin-100 results obtained on *GH-SO-TW* and *HiggsTW* (here chosen as representatives of a scenario that is substantially consistent over all datasets in general).

Considering pairs of (C, L)-solutions, we observed lack of positive correlation, for any setting of αs and weighting scheme. More specifically, Fagin-100 values are consistently equal to zero whereas Kendall values tend to be negative, with range -0.38–0 on *GH-SO-TW* and -0.48–0 on *HiggsTW*. This clearly indicates that, as expected, the lack or difference in correlation between the contributor and lurker rankings obtained by mlALCR are exacerbated when considering the entire solutions over all layers.

Impact of settings of αs

From the previous stage of evaluation of C-solutions vs. L-solutions, we observed no clear differences due to the setting of αs. In effect, looking at Table 3.3 and considering each role at a time, we find both high Kendall correlation and Fagin intersection between C-solutions as well as between L-solutions, for any fixed weighting scheme and varying the setting of αs. Interestingly, in most

role, αs config.	*GH-SO-TW*			*FF-TW-YT*			*HiggsTW*				*Flickr*				
	gh	*so*	*tw*	*ff*	*tw*	*yt*	*fw*	*mt*	*rp*	*rt*	*t1*	*t2*	*t3*	*t4*	*t5*
$L\ ^{\alpha}\langle 0.5\rangle$	0.728	0.589	0.799	0.917	0.928	0.750	0.887	0.880	0.815	0.911	0.928	0.945	0.951	0.955	0.951
$C\ ^{\alpha}\langle 0.5\rangle$	0.713	0.864	0.807	0.905	0.907	0.938	0.923	0.868	0.839	0.945	0.882	0.916	0.936	0.947	0.974
$L\ ^{\alpha}\langle 0.85\rangle$	0.771	0.714	0.945	0.947	0.926	0.799	0.976	0.833	0.588	0.620	0.936	0.951	0.958	0.963	0.959
$C\ ^{\alpha}\langle 0.85\rangle$	0.794	0.911	0.883	0.914	0.928	0.919	0.927	0.932	0.887	0.933	0.864	0.911	0.939	0.954	0.987

TABLE 3.4: Kendall correlation for comparison of the weighting schemes: uniform vs. proportional to layer edge-set size.

cases values corresponding to the comparison of L-solutions are significantly higher than those corresponding to the comparison of C-solutions, for both assessment criteria.

Considering layer-projected solutions, again it stands up the good correlation between any two C-solutions (resp. L-solutions) corresponding to $^{\alpha}\langle 0.85\rangle$ and $^{\alpha}\langle 0.5\rangle$, over all datasets. Let us next provide details concerning Kendall correlation. On *FF-TW-YT*, correlation values for L-solutions are quite uniform and range from 0.84 on *YouTube* to 0.88 on *Twitter* and *FriendFeed*, whereas C-solutions for different αs have correlation of 0.91 on *YouTube*, 0.83 on *Twitter*, and 0.68 on *FriendFeed*. On *GH-SO-TW*, there is a tendency of higher correlation between C-solutions than between L-solutions on *GitHub* (0.93 vs. 0.8) and *StackOverflow* (0.95 vs. 0.76), while similar values are obtained on *Twitter* (around 0.87). An opposite tendency occurs in *Flickr*, where correlation between L-solutions ranges from 0.93 to 0.96 whereas correlation between C-solutions ranges from 0.88 to 0.94. On *HiggsTW*, there is an interesting tendency in the difference between comparisons of L-solutions and comparisons of C-solutions, which relies on the *followship* layer. In fact, C-solutions for different αs are characterized by higher correlation values (0.91, 0.91, 0.86) on layers *retweet*, *mention*, and *reply* respectively, while layer *followship* corresponds to correlation of about 0.28; conversely, L-solutions have high correlation only for *followship*, while on *mention*, *retweet*, and *reply* the values are relatively lower (0.49, 0.48, 0.25, respectively).

Impact of layer weighting scheme

We also compared the solutions of mlALCR corresponding to the two layer weighting schemes, i.e., uniform and proportional to each layer edge-set size, in order to understand their impact on the ranking performance.

Considering whole ranking solutions, we generally observed very high values of both assessment criteria, for any fixed role and setting of αs; for instance, from Table 3.3, Kendall correlation is above 0.53 on *GH-SO-TW* and 0.83 on *HiggsTW*, whereas Fagin-100 intersection is above 0.9 in most cases. Analogously, correlation is moderate or very high also in the case of layer-projected solutions, as reported in Table 3.4 for Kendall correlation.

Let us next consider, for each layer, the mean and standard deviation over the two roles and setting of αs. On *FF-TW-YT*, Kendall (resp. Fagin-100) mean values are 0.92 (resp. 0.9) for *FriendFeed*, 0.85 (resp. 0.86) for *YouTube*, 0.92 (resp. 0.93) for *Twitter*, with standard deviation below 0.09 in all cases. Similar situation occurs for *Flickr*, whereas some exception is observed on *HiggsTW*:

while Kendall mean is 0.93 for *followship*, and 0.88 for *mention* with standard deviation always above 0.04, for layers *reply* and *retweet* we observe very high correlation (0.81 or above) for the C-mlALCR solutions and, with ${}^{\alpha}\langle 0.5\rangle$, also for the L-mlALCR solutions (quite lower values, between 0.59 and 0.62, correspond to L-mlALCR solutions with ${}^{\alpha}\langle 0.85\rangle$).

It should be noted that using one scheme relies on the choice of the proper weight to assign the size of any particular layer: the uniform weighting scheme considers the actual proportion of the layer to the complex system, while the layer-proportional scheme should be used to normalize the above effect. In Sect. 3.6.2, where we comparatively evaluate mlALCR with other methods, we will refer to the use of uniform weights, unless using the layer-proportional scheme would yield to significant difference in the relative performances of the methods.

Convergence aspects

mlALCR can achieve score convergence in few tens of iterations, in most of our evaluation cases (results not shown). In particular, on *FF-TW-YT*, the algorithm converged in 90 or fewer iterations, down below 40 with ${}^{\alpha}\langle 0.85\rangle$ and layer-proportional weighting scheme. While the weighting scheme, however, does not seem to impact on the convergence rate, we observed instead a significant reduction in the execution of mlALCR for the setting of αs to 0.85. This suggests that accounting more for the user's behavior locally to the membership layer would contribute to make mlALCR reach its equilibrium faster than in the balanced setting (i.e., ${}^{\alpha}\langle 0.5\rangle$).

mlALCR also showed to be relatively robust when the presence of users with strong alternate lurker-contributor behavior over the layers is less evident, like in the case of *Flickr*; on this dataset, mlALCR converged slowly, but within about 150 iterations with ${}^{\alpha}\langle 0.85\rangle$.

Qualitative evaluation

We investigated the meaningfulness of the ranking solutions produced by mlALCR in order to get evidence of that our proposed approach actually behaves as theoretically expected. For this purpose, we retrieved the publicly available web pages of top-ranked users, for both lurker and contributor mlALCR solutions, and manually inspected their profiles.

For this analysis we focused on *GH-SO-TW*, mainly because of three reasons: freshness, richness, and significance. Among our evaluation datasets, it is the most recently crawled one, which ensures better consistency with the user relations exploited to build the multilayer network. It is also the richest dataset in terms of variety and number of user features that can intuitively be seen as indicators of the users' online activities, both in terms of production and consumption of information. Specifically, from each network and user page, we selected and extracted (via Rest API) quantitative as well as qualitative information, which are reported next:

- from Twitter: number of tweets, number of lists the user belongs to, number of likes provided to posts of other users, membership date;
- from GitHub: number of commits during the last week, number of contributions during the last year, number of starred repositories, current and last streak (days-in-a-row) if any, membership date, time of last contribution, duration of the longest streak (in days), and date of the longest streak;
- from StackOverflow: number of the people reached, number of profile views, number of badges or medals (gold, silver, and bronze), date of the last access, top-overall percentage and reputation score, number of top posts, score relative to the top tags, and a description of the user profile.

Moreover, *GH-SO-TW* is particularly suitable for our analysis because it represents a prototype online social environment for sharing knowledge across different levels of expertise of the users, through various and sophisticated ways of endorsement and interaction among them, which opens to different user behavioral scenarios.

One of these scenarios concerns users who might want to exploit Twitter to create and spread information related to software development (e.g., for marketing reasons) but, perhaps due to the higher cost of contribution when specialist expertise is required, without contributing to the domain to which the expertise pertains. Indeed, we actually found several cases of users that are very active on Twitter, while behaving as lurkers on the other two platforms. Taking as case in point user S. K. which is a top-ranked user in the L-mlALCR solution over the *GitHub* layer, he joined GitHub on November 2012, counts a high number of followees (177) but a low number of followers (12), and has starred (in recent times) a good number (27) of repositories; the latter aspect indicates the willingness of the user of being recognized as member of the GitHub community, though as observer rather than as content producer (his last contribution is in fact dated October 2015). The same user shows an even more consumption-oriented behavior on StackOverflow, whereas he appears to be very active and popular on Twitter, with more than 10k tweets produced, only 185 followees and almost 8k followers.

Among the top-ranked users, we also identified several cases of people which are active on Twitter as well as they show a strong preference towards one particular developer community, i.e., they are active contributors on GitHub while showing lurking behaviors on StackOverflow, or vice versa. One example is given by user C. H., which is top-ranked in the C-mlALCR solution over *Twitter*. He joined the platform on April 2009 and he has only 64 followees, more than 10k followers and more than 4k tweets. He is definitely a contributor also on GitHub, with unbalanced counts of followers (2.8k) and followees (54). He has starred more than 300 repositories and his last contribution is quite recent (January 2016). It should be noted that his high score in the C-mlALCR ranking is explained not only by his active contribution on Twitter, but also from his lurking status on StackOverflow, where he shows low reputation (64), asked only 4 questions and answered none.

Other users might seek to gain benefits for their profession by actively participating in the StackOverflow and/or GitHub developer communities, while not being much interested in posting tweets. This is the case, for example, of user R. B., which appears to be a master on StackOverflow and GitHub, but relatively inactive on Twitter. He indeed shows an extremely active/influent behavior on StackOverflow, where he has numerous badges and more than 10 (resp., 100, 200) gold (resp. silver, bronze) medals. He shows extremely high reputation (above 40k), reaching about 2.7M people, answering almost 1k questions and asking more than 200 questions. However, his status is different on GitHub, where he provided a few contributions over the last year, and on Twitter, showing in both platforms quite a low level of socialization.

Summary of mlALCR evaluation

We summarize here our main findings from the evaluation of mlALCR.

- We have pointed out the meaningfulness of the problem under study on real-world multilayer networks, which indeed contain users that are characterized by evident variability in their cross-layer lurker/contributor behavior. Our mlALCR has shown to be able to identify users showing alternate behavior over the different layer networks.

- Both within-layer and cross-layer attachment distributions of lurkers vs. contributors, and vice versa, are better fitted by log-normal distributions than power-law ones.

- Lurker and contributor mlALCR solutions are clearly uncorrelated over the same network. When considering layer-specific projections of the solutions, there is a total disagreement in matching between the top-ranked users of C-mlALCR and L-mlALCR over the same layer, while correlation gets high for any pair of layers characterized by the presence of users with alternate lurker-contributor behavior.

- Neither the setting of damping factors α_l and α_c nor the weighting scheme has heavy effect on the ranking solutions of mlALCR; however, the setting of αs impacts on the convergence of mlALCR, whereby accounting more for the user's behavior locally to the membership layer (i.e., ${}^{\alpha}\langle 0.85\rangle$) leads mlALCR converge faster.

- Empirical evidence based on manual inspection of top-ranked users has confirmed the ability of mlALCR in detecting users that simultaneously exhibit lurking behavior on one layer (e.g., OSN platform) while they spread the acquired knowledge acting as contributor on other layers, or vice versa.

3.6.2 Comparative evaluations

Comparison with LurkerRank

In order to understand how mlALCR differs from the basic monoplex LR method, we analyzed the LR solution obtained on each layer of a network and the corresponding layer projections of mlALCR. Here we discuss Kendall correlation results (not shown).

In two datasets, namely *FF-TW-YT* and *Flickr*, correlation is very close to zero and does not significantly vary with the choice of role and layer, and with the setting of αs. Specifically, on *FF-TW-YT*, correlation values are mostly in the range ±1.0e-3 for *Twitter* and *FriendFeed*, and tend to be slightly negative for *YouTube* (around -3.0e-2). On *Flickr*, correlation with LR is in the same range (0.07) for both L- and C-solutions on layers *t1*, *t2*, *t3*, while is lower on layer *t4*. The most recent time slot (*t5*) presents a negative correlation with L-solutions and a positive correlation with C-solutions both in the range of 0.01.

In the other two datasets, we mostly observed a few cases of significant negative (resp. positive) correlation between LR and C-solutions (resp. L-solutions) of mlALCR. For instance, for *Twitter* on *GH-SO-TW*, correlation values range from -0.26 to -0.16 w.r.t. C-solutions, and from 0.44 to 0.56 w.r.t. L-solutions, which indicates a tendency of having lurkers in *Twitter* that are likely to be active in the other platforms; further analysis in terms of Fagin-100, with values below 0.06, however shows that such correlation is not determined by the top-ranked lurkers.

More interesting is the situation analyzed on *HiggsTW*. Here there is a significant negative correlation between LR and C-solutions over each layer; specifically, up to -0.26 on *followship*, -0.23 on *mention*, and -0.20 on *reply*. This clearly characterizes a strong mismatching between lurkers determined by LR according to a given user relation and the corresponding contributors determined by mlALCR for the same relation, which might behave as lurkers for other relations. Considering L-solutions, we observed relatively discordant results depending on the setting of αs: positive, up to 0.32 (resp. negative, up to -0.21) for solutions with ${}^{\alpha}\langle 0.5\rangle$ on *retweet* (resp. *reply*), and conversely slightly negative, up to -0.14 (resp. positive, up to 0.32) for solutions with ${}^{\alpha}\langle 0.85\rangle$. By contrast, on *followship* and *mention*, we observed good correlation (0.60 and 0.33, respectively); this is also strengthened by significant matching in terms of Fagin-100 results: 0.22–0.27 for *followship* and 0.12–0.39 for *mention*, which is in line with our analysis of results on *HiggsTW* presented in Sect. 3.6.1.

Comparison with LR-aggregation methods

We compared the mlALCR solutions (by varying the role, configuration of (α_c, α_l) and weighting scheme) with each of the two proposed ranking aggregation methods. For this evaluation, we discuss Fagin-100 results (not shown).

In general, we observe that the comparison with both ranking aggregation methods is characterized by near-zero or zero Fagin-100 with L-mlALCR as well as with C-mlALCR. More in detail, considering first LRa1, values are zero on

Flickr and *GH-SO-TW* (in all cases). On *FF-TW-YT*, Fagin-100 is 0.021 on average, with peak of 0.04, w.r.t. C-solutions, and 0.023 on average, with peak of 0.02, w.r.t. L-solutions. While on *HiggsTW*, Fagin-100 values range from 0 to 0.02 (the latter obtained w.r.t. C-solution with $^{\alpha}\langle 0.85\rangle$). Comparing mlALCR to LRa2, all of non-zero Fagin-100 values correspond to mlALCR$_{max}$, being around 0.03 for C-solutions and around 0.07 for L-solutions on *GH-SO-TW*. On *HiggsTW*, Fagin-100 is always zero w.r.t. mlALCR$_{min}$ for C-solutions, while non-zero values are equal to 0.02 (resp. 0.12) for mlALCR$_{med}$ (resp. mlALCR$_{max}$); concerning L-solutions there is no matching w.r.t. mlALCR$_{min}$ and mlALCR$_{med}$, while mlALCR$_{max}$ values are close to 0.18. On *FF-TW-YT*, in regard to C-solutions all non-zero values are below 0.1 except for mlALCR$_{med}$ w.r.t. $^{\alpha}\langle 0.5\rangle$ and non-uniform weighting with values equal to 0.16 (resp. 0.23) for C-solution (resp. L-solution). No matching at all is observed between mlALCR and LRa2 on *Flickr*.

Overall, we find out that the statistical manipulation of ranks over the layers provided by both LRa1 and LRa2 leads to no significant matching with top-ranked users by mlALCR.

Comparison with Multilayer LurkerRank

Comparing mlALCR with mlLR (results not shown), we observed some cases of positive Kendall correlation over the datasets. On *FF-TW-YT*, maximum correlation is obtained between L-mlALCR$_{max}$ (for any setting of αs) and mlLR ranging from 0.57 to 0.61, which decreases down below 0.5 when compared with mlALCR$_{med}$. On the other datasets, the maximum correlation corresponds always to comparison with L-mlALCR$_{max}$, but with lower values (on *GH-SO-TW*, 0.33 with $^{\alpha}\langle 0.85\rangle$, and 0.26 with $^{\alpha}\langle 0.5\rangle$; on *HiggsTW*, between 0.27 and 0.4 with $^{\alpha}\langle 0.5\rangle$). Minimum correlation is always obtained with mlALCR$_{min}$ (between 0.12 and 0.16 on *GH-SO-TW*, around 0.08 on *FF-TW-YT*, 0.04 on *HiggsTW*, while it varies from -0.08 to 0.1 considering C-solutions and L-solutions, respectively, on *Flickr*).

The observation of Fagin-100 results sheds light on the evidence that, over all datasets and regardless of the role and setting of αs, matching is always close to zero (below 0.07 on *HiggsTW*, 0.05 on *GH-SO-TW* and *FF-TW-YT*, 0.01 on *Flickr*). Coupled with the previously given observations about Kendall correlation, this indicates that mlALCR and mlLR might be relatively correlated though mainly along the tail.

Comparison with Multiplex PageRank

Our final stage of evaluation concerns a twofold comparison between mlALCR and the Multiplex PageRank method, in all of its three variants (Sect. 3.5.2). On the one hand, analogously to the previous analyses, we want to understand whether mpxPR solutions can be, to a certain extent, correlated to L-mlALCR, i.e., whether mpxPR can be able to discover users that behave as lurkers in one layer and as contributors in other layers; on the other hand, we want to capitalize on the actual nature of mpxPR (i.e., searching for high PageRank users in a multilayer context), and hence we compared it with C-mlALCR solutions to determine any affinity in the identification of contributors on individual layers. In the latter case, to properly set the input for mpxPR for this task, we reversed the orientation of the layer graphs, in the classic influence-oriented centrality model (i.e., edge (u, v) expresses endorsement of u to v).

We used *FF-TW-YT* and *GH-SO-TW* as cases in point. This implies that, being the number of layers equal to three in both datasets, there are six different layer configurations (two per layer) for the application of any of the variants of mpxPR. (Recall that mpxPR is dependent on the layer ordering.)

On *FF-TW-YT*, for both C- and L-solutions values are near-zero or zero for all three variants of mpxPR, for both assessment criteria. More variegated is instead the situation on *GH-SO-TW*. Considering L-solutions, Kendall correlation and Fagin-100 intersection range from -0.15 to 0.05 and from 0.16 to 0.18, respectively, with the exception of *GitHub* where Fagin-100 has a peak around 0.5 when comparing with the combined variant of mpxPR. Considering the comparison with C-solutions, all variants of mpxPR again tend to yield near-zero correlation with C-mlALCR by Kendall, and between 0.07 and 0.18 in terms of Fagin-100.

This can intuitively be explained since users that are highly central (i.e., contributors in the case of comparison with C-mlALCR, lurkers in the other case) on one layer L', and less central on the others, are more likely to receive a final ranking score from mpxPR which is substantially determined by the score on L'. Therefore, for all users determined by mlALCR as contributors (resp. lurkers) on L' but who do not show a strong alternate behavior elsewhere, there might be a relatively higher correlation with those users' ranking obtained by mpxPR.

It should also be noted that all variants of mpxPR are sensitive to the ordering of the layers: as an example, for the comparison with L-mlALCR and final ranking on *Twitter*, we observed two controversial results yielded by the additive variant: Kendall of 0.005 for the sequence *tw*←*so*←*gh* but -0.12 for *tw*←*gh*←*so*.

Summary of comparative evaluation

Comparing mlALCR with other methods allowed us to draw the following main remarks:

- Within-layer lurkers identified by LR have generally no correlation (resp. strong mismatching) with cross-layer lurkers (resp. contributors) identified by mlALCR in the same layer. However, some correlation might be observed when lurkers in a given layer are likely to behave actively on other layers (for example, in single-platform contexts like *HiggsTW* this might occur for *followship* vs. *retweet*, according to a least-effort economy principle of interactional behavior).

- No significant matching is found between mlALCR and any of the aggregation methods which statistically manipulate the layer-specific ranks obtained by LR.

- Top-ranked users in mlALCR and in the multilayer LR (mlLR) have no matching, while any positive correlation that may occur by comparing the entire ranking solutions depends on their respective tails.
- Upon testing in both lurking-oriented and contributor/influential-oriented user relation models, mpxPR can reward a user only according to the same role s/he consistently has over the layers of a network (though biased by the ordering of examination of the layers), while in any case it cannot detect cross-layer opposite roles.

Overall, the conducted comparative evaluation has shown the uniqueness of mlALCR in mining alternate lurker-contributor behaviors on a multilayer network.

3.7 Conclusions

In this work we focused on the dichotomy between contribution and consumption of information over multilayer OSNs. In this respect, we addressed the novel problem of identification and characterization of opposite behaviors that users may alternately exhibit over the multiple layers of a complex network. We proposed the first topology-driven ranking method for *alternate lurker-contributor behaviors* on a multilayer OSN, named mlALCR, which is designed to identify users that behave as lurkers (resp. contributors) in one layer while conversely acting as contributors (resp. lurker) in one or many of the other layers. Significance and uniqueness of mlALCR have been empirically demonstrated over four real-world multilayer networks. We have also discussed a number of applications that might benefit from the proposed approach, including user engagement, viral marketing and information containment.

Natural extensions of our mlALCR include the embedding of the temporal dimension and content information into the mlALCR ranking model. It would be interesting to study relations between the alternate lurker-contributor behaviors and properties of assortativity at network level [125] or at node level [126] which might be defined based on suitable notions of cross-layer centrality. Optimal-convergence analysis for specific settings of the α parameters would also represent an interesting theoretical investigation, which could be effectively addressed via gradient-based optimization (e.g., [127]). In addition, we believe it would be interesting to integrate mlALCR into a task of community detection in order to analyze knowledge transfer flows between cross-layer communities.

Chapter 4

A supervised approach to user behavioral problems

4.1 Introduction

Learning-to-rank (LTR), i.e., using machine learning to automatically build a ranking model, has become one of the key technologies for modern web search, retrieval and personalization [128], [129]. As in traditional ranking functions, a partial ordering (i.e., ranking) is provided for a set of objects, according to their degree of relevance to a given query; in LTR, the ranking function is learned from *training data* given in the form of $\langle$*query, object, relevance label*$\rangle$ tuples. In this chapter, we investigate whether learning-to-rank (LTR) can be successfully applied to ranking problems in OSNs. In particular we concentrate our effort to the detection of two specific types of online behavior: automated (i.e., bots) and lurking.

Automated behavior. Bots have often been regarded as harmless programs confined within the cyberspace. However, recent events in our society proved that they can have important effects on real life as well. Bots have in fact become one of the key tools for disseminating information through online social networks (OSNs), influencing their members and eventually changing their opinions. With a focus on classification, social bot detection has lately emerged as a major topic in OSN analysis; nevertheless more research is needed to enhance our understanding of such automated behaviors, particularly to unveil the characteristics that better differentiate legitimate accounts from bots. We argue that this demands for learning behavioral models that should be trained using a large and heterogeneous set of behavioral features, so to detect and characterize OSN accounts according to their status as bots. Within this view, in Sect. 4.3 we push forward research on bot analysis by proposing a machine-learning framework for identifying and ranking OSN accounts based on their degree of bot relevance.

Lurking behavior. While being long researched in social science and computer human interaction, lurking behaviors in online social networks (OSNs) have been computationally studied only in recent years. Remarkably, determining lurking behaviors has been modeled as an unsupervised, eigenvector-centrality-based ranking problem, and it has been shown that lurkers can effectively be ranked according to the link structure of an OSN graph. Although this approach has enabled researchers to overcome the lack of ground-truth data at a large scale, the complexity of the problem hints at the opportunity of learning from past lurking experiences as well as of using a variety of behavioral features, including any available, possibly platform-specific information on the activity and interaction of lurkers in an OSN. In this work, we leverage this opportunity in a principled way, by proposing a machine-learning framework which, once trained on lurking/non-lurking examples from multiple OSNs, allows us to predict the ranking of unseen lurking behaviors, ultimately enabling the prioritization of user engagement tasks.

4.2 Learning to rank methods

Learning-to-rank (LTR) is a supervised learning approach to build a ranking model. LTR methods are commonly organized into three categories, namely *pointwise*, *pairwise* and *listwise* [128], which correspond to different modeling of the input data and types of loss function to be minimized. Pointwise methods assume that every query-object pair is a learning instance, whose score has to be predicted. Ordinal scales are mapped into numeric values, therefore the ranking problem is seen as regression or ordinal classification. Object pairs are the learning instances in pairwise methods, which hence aim to learn which object in a pair precedes the other in the ranking. Optimization here consists in minimizing the number of switched/misclassified object pairs. In listwise methods, a learning instance is comprised of a query and its objects, and a quality criterion is optimized over all queries in the training data. Unlike the other two types of LTR, listwise methods take into consideration that the selection of features is not unbiased, since they depend on the queries, which in turn vary, so that some objects or object pairs might not be comparable with each other.

Several LTR methods have been developed since the last fifteen years [128], [130]. Here, for the purpose of this research, we focus on pairwise and listwise methods, which have shown to perform generally better than the earlier pointwise ones, and we briefly recall some of the most representative methods.

RankNet [131] is a pairwise method, which has become popular in commercial search engines (e.g., Microsoft Bing®). In RankNet, the training scheme is based on a neural network with two hidden layers and one output node, and back-propagation is used to minimize the following cost function, which represents the pair-wise cross entropy between the target probability $\bar{P}_{ij}$ and the modeled probability P_{ij}, for each pair objects i, j:

$$C_{ij} = -\bar{P}_{ij} \log(P_{ij}) - (1 - \bar{P}_{ij}) \log(1 - P_{ij}) \tag{4.1}$$

Coordinate Ascent [132] is a listwise method, where the scores of the query-object pairs are computed as weighted combination of the features values. The weights are tuned by using coordinate ascent optimization, a derivative-free optimization technique, in which the objective function can be any arbitrary IR evaluation criterion, and such that the optimization is performed in one dimension at a time while keeping the other dimensions fixed. If we denote with λ_i the i-th parameter to learn, then the update rule is given by:

$$\lambda'_i = \underset{\lambda_i}{arg\,max}\ E(R_\Delta, T) \tag{4.2}$$

where R_Δ is the set of rankings induced over all of the queries, T is the training data, and $E(\cdot,\cdot)$ denotes an evaluation function.

AdaRank [133] is a listwise boosting method inspired by the classification algorithm AdaBoost [134]. Through the use of a stepwise greedy optimization technique, it maximizes a chosen IR evaluation criterion by repeatedly building "weak rankers" on the basis of re-weighted training data. These weak rankers are finally linearly combined to produce the ranking predictions. One key aspect in AdaRank is that higher weights are assigned to queries whose relevant objects are more difficult to rank, whereas already learned queries are associated with lower weights on the basis of the forward stage-wise additive modeling paradigm. The objective function to be minimized has the form:

$$\min_{\substack{h_t \in \mathcal{H} \\ \alpha_t \in \mathbf{R}^+}} L(h_t, \alpha_t) = \sum_{i=1}^{m} \exp(-E(\pi(q_i, \mathbf{d_i}, f_{t-1} + \alpha_t h_t), \mathbf{y_i})) \tag{4.3}$$

where E is the quality criterion function, π represents the permutation for the i-th query q_i, the list $\mathbf{d_i}$ of retrieved documents for q_i, and the ranking model f. Moreover, $\mathbf{y_i}$ corresponds to the relevance labels (i.e., desired ranks), $\mathcal{H}$ denotes the set of possible weak rankers h_t, and α_t is a positive weight.

LambdaMART [135] is an ensemble method that combines LambdaRank [136] and MART [137]. The former performs optimization via the gradient of the loss function, while the latter is a boosted regression tree model in which the output is a linear combination of the outputs of a set of regression trees. LambdaRank is based on a neural network, like RankNet, and it stands out for one main aspect: instead of focusing on the definition of a less expensive cost function, it takes into account the gradient of the cost function directly, avoiding the additional computational cost introduced by sorting operations [138]. For each pair of objects i, j, the λ-gradient is defined as:

$$\lambda_{ij} = S_{ij} \left| \Delta Z_{ij} \frac{\partial C_{ij}}{\partial(s_i - s_j)} \right| \tag{4.4}$$

where $s_i - s_j$ is the difference of ranking scores of the two objects w.r.t. a query, $C_{ij} = s_i - s_j + \log(1 + \exp(s_i - s_j))$ is the cross-entropy cost, ΔZ_{ij} is the evaluation value gained by swapping the two objects, and S_{ij} is equal to $+1$ if object i is more relevant than object j, -1 otherwise.

4.3 Bot detection

Software robots, or bots, are computer programs designed to carry out one or more specific automated tasks. Social bots are a particular type of chat-bots employed in social media platforms to automatically interact with members and generate contents. Bots can be used to provide useful services such as customer support, meeting scheduling, tracking of product shippings, and so on. On the other hand, social bots often mimic a human being by controlling a social media account, and when acting as a group (i.e., botnet) they can also pursue malevolent intents, such as fostering fame [139], [140], biasing public opinion [141], [142], spamming or limiting free speech by submerging important messages with a deluge of automated bot messages [143].

In the last years, also thanks to minimal skill requirements, the use of automated accounts has seen an unprecedented rise [144]–[146]. Remarkably, groups of orchestrated bots are being used to steer public opinion and influence the electoral audience through the spread of (mis)information and fake news [147]; the most representative example is related to the latest US presidential election [141], [142], but several other cases can be listed [148]–[151].

Bot detection is a challenging problem, also due to the variety of strategies implemented by bots. For instance, they can mimic human behavior in order to blend in among legitimate accounts (i.e., users) and earn their trust, making bot detection more difficult [152], [153]. Bots can also achieve an amplifying effect by replicating contents related to a specific topic in order to distort the perception of its popularity, or create the appearance of grassroots support for a position (i.e., astroturfing) by over-promoting that point of view [154], [155].

The software behind social botnets is constantly evolving, introducing new expedients that result in sophisticated simulations of human behavior in OSNs [145], [146], [156], making the identification of automated accounts ever more complicated. Therefore, there is an emergence for the development of effective methods able to better understanding the behavioral patterns that characterize automated OSN accounts.

Research questions. Despite several approaches have been developed to detect bots in OSNs, a single method often covers only one specific pattern of automated behavior, leaving the identification challenge open. Taking inspiration from ensemble learning theory, whereby multiple weak learners are combined together in order to boost prediction performance, in this work we aim at developing a more general framework for the identification of bots that can exploit different sources of information and be capable of learning from multiple type of bot detection methods. Within this view, we raise the following questions:

- *How can we exploit previously identified automated accounts to tune a model for detecting and ranking bots?*
- *How can we successfully combine several bot identification methods to capture different behavioral patterns?*
- *How can we incorporate into a bot ranking model various "signals" that can be used as behavioral features, upon which the evaluation of any user-account w.r.t. a given context is performed?*

Motivations. We believe that LTR is particularly suitable for addressing the aforementioned problem, helping us to improve our understanding of automated behaviors in online social environment. The inherent complexity of the bot detection problem hints at the exploitation of techniques capable of determining the status as bot of a (suspected) OSN account, rather than simply indicating whether or not an account is a bot. In fact, LTR adopts a supervised approach to learn from past user experiences, which might be annotated according to the degree of bot relevance. LTR training is accomplished according to a set of features which, being of different types, can be useful predictors capturing different aspects of automated behaviors. LTR also offers unprecedented opportunities for incremental scenarios: once trained a learning-to-bot-rank model, this can be used to assign any previously unobserved account with a bot relevance score.

Contributions. In this work, we propose the first LTR framework to detect and characterize social bots in OSNs. We develop a *learning-to-rank-social-bot* methodology based on state-of-the-art LTR methods that exploits, in the process of feature extraction, information of different type (such as, user profile, user's activity rate, media content) and is supported by three state-of-the-art bot-detection methods, namely *DeBot*, *BotWalk*, and *BotOrNot*. Results obtained on four state-of-art datasets have shown the significance of our approach, confirming our initial hypothesis that LTR can be effective for unveiling automated behaviors.

The remainder of this chapter is organized as follows. Section 4.3.1 briefly discusses related work, whereas Section 4.2 describes LTR and bot-detection methods utilized in our framework. Section 4.3.2 is devoted to the proposed LTRSB framework. Section 4.3.3 presents experimental results and Section 4.3.4 concludes the chapter.

4.3.1 Related work

Bot detection Currently adopted approaches for bot detection belong to three main categories: graph-based, time-series-analysis-based, and hybrid methods. The former group utilizes network information (e.g., contacts, interactions, (re)tweets) in order to discern between legitimate accounts and bots [157]–[159]. The second group of methods focuses on identifying distinctive temporal behavioral patterns of bots (e.g., burst of interactions, non-stop activity, etc.) [160]–[164]. The third group includes software systems that are designed to use features of different type to train one classifier [165]–[167] or an ensemble of classifiers [168]. A different perspective is taken from *BotWalk* [169], which computes an aggregated anomaly score based on an ensemble of unsupervised anomaly detection methods.

Remarkably, most of existing methods aim to detect bots either focusing on a specific aspect of automated behavior, and for this reason they cannot be considered general enough, or by exploiting features of different type while, however, focusing on binary classification problems.

Learning-to-rank in OSNs Originally developed to meet the rising needs of modern web retrieval systems, LTR techniques have also been exploited in OSN analysis contexts. In particular, LTR frameworks have been developed to

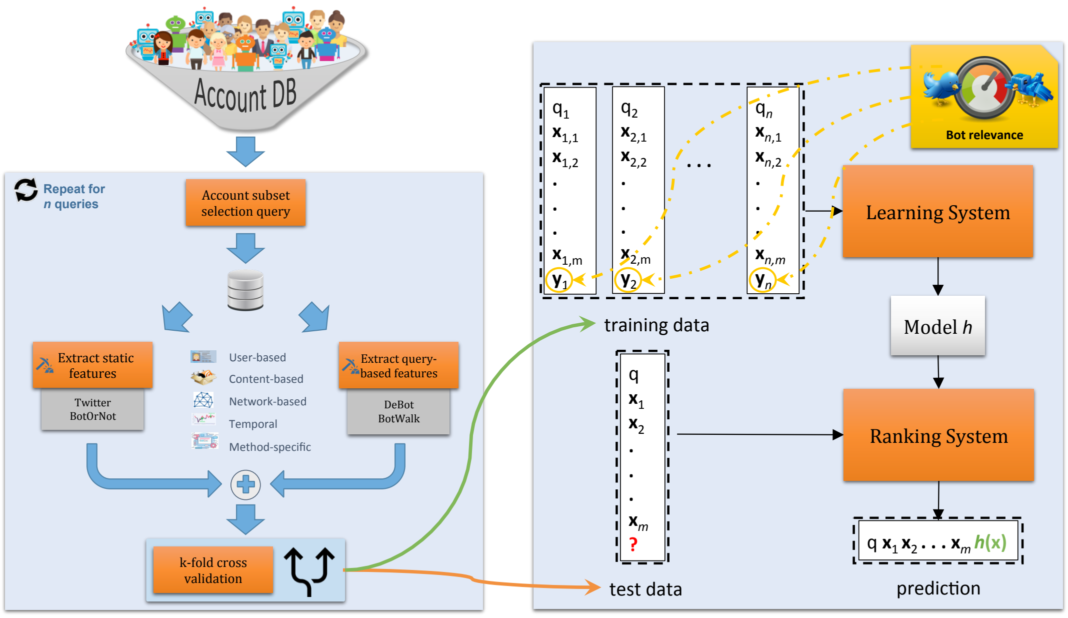


FIGURE 4.1: Main modules and data flows of our proposed learning-to-rank-social-bot (LTRSB) framework

tackle problems such as hashtag recommendation [170], credibility assessment of tweet content [171], ranking answers in large online question/answer collections [172], and also to address behavioral problems as user engagement [173], [174]. Nevertheless, to the best of our knowledge, LTR has never been used so far to address bot detection problems.

Bot detection methods

We focus our attention on three of the most relevant and recent methods for bot classification, namely *BotOrNot* [166], [167], *DeBot* [164], and *BotWalk* [169].

Given a Twitter screen-name as input, *BotOrNot* [166], [167] retrieves information about the activity of the account to generate a fairly extensive set of features. BotOrNot resorts to a classifier to compute a score describing the likelihood that an account is a bot. Various standard classification models were tested in [166], including AdaBoost, logistic regression, decision trees, and Random Forest; the latter models, which relies on an ensemble learning approach to combine many decision trees, was found the most accurate to produce bot-likelihood scores.

DeBot [164] approach is to detect accounts characterized by high temporally correlated activities. Each account timeline is represented by a time series of the tweet/retweet actions performed by the account at each time step. To compare time series, DeBot defines a correlation measure based on *dynamic-time-warping* (DTW) [175], a widely used method to compute the distance between two sequences by warping them locally to the same length (i.e., it allows one-to-many mappings between series to stretch a sequence, or many-to-one mappings to compress a sequence). The DTW-based correlation between two (normalized) time series $\hat{x}$ and $\hat{y}$ is defined as:

$$wC(x,y) = 1 - \frac{DTW^2(\hat{x},\hat{y})}{2P} \tag{4.5}$$

where P is the number of squared errors that are added to obtain a distance (i.e., the path length) [164]. DeBot performs a 4-stage workflow: first, tweets that contain selected keywords are gathered for a period of T hours, then it assembles the time series for each user. These series are processed based on a hash index in order to identify correlated activity patterns between two or multiple accounts. Third, the activity of suspected accounts is closely monitored through the stream Twitter API. The last step computes a pairwise warped correlation matrix, over the newly generated time series, and generates clusters of highly correlated users.

BotWalk [169] is a near real-time adaptive bot-identification method based on an ensemble of anomaly detection methods. Like DeBot, BotWalk adopts an unsupervised approach and focuses on specific distinguish patterns of automated behavior. In addition, BotWalk utilizes 130 features extracted from network, content, temporal and metadata information. Starting from a seed-bot and a set of random accounts, BotWalk retrieves each account's details, timeline, and one-hop follower neighborhood, with the goal of maximizing the likelihood of reaching other bots across the OSN. The output of the method is represented by an aggregated score, obtained by combining four different anomaly detection scores.

4.3.2 Proposed Framework

Overview

Figure 4.1 shows a schematic illustration of the proposed **L**earning-**T**o-**R**ank-**S**ocial-**B**ots (**LTRSB**) framework.

We are given a database storing information about legitimate accounts and bots gathered from heterogeneous sources. This database feeds information to a component that is in charge of (i) repeatedly selecting a subset of accounts to define queries for the LTR module, and (ii) extracting static as well as dynamic features of different types, as we shall describe later.

In this work we use Twitter as case in point, although our framework is in principle designed to be versatile and applicable to other OSN platforms. The feature extraction step is performed through Twitter API as well as through retrieving functionalities provided by BotOrNot, BotWalk and DeBot methods. In particular, Twitter API and BotOrNot are used to retrieve static and aggregate features, whereas BotWalk and DeBot are used to compute query-based features. More in detail, each DeBot run is characterized by a time-window of fixed length, in which all the users belonging to the query are being listened in order to compute clusters. Clusters are then examined to compute the warped correlation matrix and, finally, the DeBot features.

bots	non-bots	source	annotations
452	970	[166]	human generated
1254	1595	[177]	human generated
5937	2819	[143]	human generated
2371	2567	[176]	honeypot-based
1420	-	DeBot runs	DTW-based
11434	7951	Total	

TABLE 4.1: Composition of the Account DB

Account DB

Table 3.2 shows main characteristics of our evaluation data, for a total of about 19K accounts, 11K of which correspond to bots and the remaining to non-bots. Most of the account instances are from datasets originally built and analyzed in recent studies [143], [166], [176], [177], although from each of these datasets we removed accounts that are suspended by Twitter or no longer active; moreover, we gathered additional bot instances from Twitter through several DeBot runs.

The account instances stored in the database were originally annotated with a binary ground-truth label (i.e., bot or non-bot) according to three different approaches. The majority of instances were labeled by human experts. Accounts from [176] were instead labeled depending on whether an account is a follower of a target honeypot account. (Since honeypots are designed in such a way that a human can immediately tell if they are bots, any user in the network that connects to a honeypot will be considered as a bot). In addition to the datasets available from other studies, we exploited DeBot, which has shown to provide almost null false-positive rate, to further collect bot instances having correlation score of 0.995 or above.

Training Data

LTR training data consists of triplets $\langle$*query, object, relevance label*$\rangle$. We are given n queries, each corresponding to a subset of m. Each object is an account's feature vector, and the relevance label denotes one or several grades of *bot status*, so that higher grades correspond to more likely bots.

Relevance labeling We devised three approaches for bot relevance labeling, depending on the selection of relevant/non-relevant instances, and the type of relevance label:

- In the first configuration, dubbed BB, we considered *binary* relevance, i.e., bot and non-bot, and performed a *balanced* selection of relevant and non-relevant accounts.

- In the second configuration, dubbed UB, we again considered *binary* relevance, and performed unbalanced selection of relevant (30%) and non-relevant (70%) accounts.

- In the third configuration, dubbed Grad, we considered balanced selection of relevant and non-relevant accounts, but seven grades of *bot status*.

The first two configurations of relevance labeling correspond to the use the original binary-annotation information in the datasets reported in Table 4.1. For the Grad configuration, we computed the relevance labels as follows. First, we uniformed the feature representation for all the accounts in our database to the BotOrNot model, by retrieving from Twitter missing information for all the accounts not previously processed by BotOrNot. Second, we applied the BotOrNot-RandomForest classifier to produce the bot likelihood score for each instance, which varies within $[0, 1]$. Third, we discretized the bot scores in seven intervals, with the interval $[0, 0.3]$ corresponding to legitimate accounts, subsequent intervals with increment of 0.1, and $[0.8, 1]$ corresponding to definitely a bot account.

Feature set We organize the set of features extracted from our evaluation data into five categories: *aggregate*, *user-based*, *network-based*, *content-based*, and *temporal*.

Table 4.4 provides a concise description of the features. The first group is comprised of nine aggregate indicators provided by BotOrNot (i.e., from A.1 to A.9) and a query-wise score produced by the BotWalk anomaly detection task. User-based features describe Twitter account settings and basic information, such as account lifetime, time-zone and language, and whether the account was verified or not. Network-based features describe node-wise properties of a user in the social network graph, and include indicators of the user's centrality, popularity and role. Content-based features represent the largest group and include several indicators illustrating the diversity and alleged quality of the content created by a user. Considering normal human-limitations in terms of time spent on social-media platforms, we defined several activity-rate features as indicators of overproduction. We also included basic statistics on the inter-arrival time of consecutive tweets, and on user activity distributions to gain insights into content production patterns of social bots.

Note that, to ensure comparability across queries, all feature values were scaled between 0 and 1, through min-max normalization.

Feature informativeness We investigated about the usefulness or informativeness of the features, with a twofold goal: to discover possible correlation between features (possibly w.r.t. the relevance class attribute), and to estimate their impact on the ranking prediction performance. For this purpose, we resorted to five standard methods used in feature selection tasks: principal component analysis (PCA), information gain, Pearson's correlation coefficient, correlation-based feature selection (CFS) [178], and learner-based feature selection exploiting J48 decision tree model.

The outcome of the analysis confirmed the significance of most of the feature categories, namely content-based, aggregate, network-based and temporal features. However, it also highlighted the necessity of diversifying features in such a context. As we shall declare in our evaluation goals (cf. Section 4.3.2),

	id	description
User-based	U.1	Short length of user description
	U.2	Number of links appearing in the description
	U.3	URL website (Boolean value)
	U.4	Verified account (Boolean value)
	U.5	Number of lists created by the user
	U.6	Number of posts (tweets)
	U.7	Account lifetime
	U.8	Geo enabled (Boolean value)
	U.9	Time zone (ID number)
	U.10	Language (ID number)
	U.11	Default profile & background (Boolean value)
	U.12	Default profile image (Boolean value)
Network-based	N.1	Number of followers
	N.2	Number of friends (followees)
	N.3	Number of tweets with at least one mention
	N.4	Number of mentions
	N.5	Number of duplicate mentions
	N.6	Follower / followee ratio
	N.7	Mention / tweet ratio
Temporal	T.1	Duration of the longest tweet session without breaks longer than 10 minutes
	T.2	Mean (μ) of the inter-arrival time of consecutive tweets
	T.3	Minimum of the inter-arrival time of consecutive tweets
	T.4	Maximum of the inter-arrival time of consecutive tweets
	T.5	Standard deviation (σ) of the inter-arrival time of consecutive tweets
	T.6	Burstiness, i.e., $(\sigma - \mu)/(\sigma + \mu)$
	T.7	χ^2 second-of-minute, refers to tweet distributions across time
	T.8	χ^2 minute-of-hour, refers to tweet distributions across time
	T.9	χ^2 hour-of-day, refers to tweet distributions across time
	T.10	Average number of tweets per day
	T.11	Maximum number of tweets per day
	T.12	Minimum number of tweets per day
	T.13	Standard deviation of number of tweets per day
	T.14	Entropy inter-arrival time
	T.15	Average number of likes per day
	T.16	Average number of tweets per day
	T.17	Query-wise average DTW-based Correlation [DeBot]
	T.18	Query-wise DTW-based Correlation offset [DeBot]
	T.19	Query-wise activity rate [DeBot]

	id	description
Content-based$^{(*)}$	C.1	Number of favorites/likes
	C.2	Number of tweets
	C.3	Number of tweets with at least one hashtag
	C.4	Number of tweets with at least one URL
	C.5	Number of retweets performed
	C.6	Number of URLs
	C.7	Number of domains
	C.8	Number of hashtags
	C.9	Number of retweets received
	C.10	Number of duplicate URLs
	C.11	Number of duplicate domains
	C.12	Number of duplicate hashtags
	C.13	Mean of the Jaccard similarity of inter-tweet bag-of-words
	C.14	Minimum of the Jaccard similarity of inter-tweet bag-of-words
	C.15	Maximum of the Jaccard similarity of inter-tweet bag-of-words
	C.16	Standard deviation of the Jaccard similarity of inter-tweet bag-of-words
	C.17	Average number of special characters
	C.18	Minimum number of special characters
	C.19	Maximum number of special characters
	C.20	Standard deviation of number of special characters
	C.21	Average of tweet lengths
	C.22	Minimum of tweet lengths
	C.23	Maximum of tweet lengths
	C.24	Standard deviation of tweet lengths
	C.25	Retweets / tweets ratio
	C.26	Number of URLs on tweets
	C.27	Number of hashtags on tweets
	C.28	Number of media files (photos or videos) on tweets
	C.29	Average number of retweets received per tweet
	C.30	Average number of likes/favorites received per tweet
	A.1	Content indicator (based on statistics about length and entropy of shared text, POS tagging, etc.) [BotOrNot]
Aggregate	A.2	Friend indicator (based on follower-friend relations, number of replies, etc.) [BotOrNot]
	A.3	Network indicator (based on mentions, retweets, and hashtags) [BotOrNot]
	A.4	Sentiment indicator (based on several sentiment extraction techniques) [BotOrNot]
	A.5	Temporal indicator [BotOrNot]
	A.6	User's basic information [BotOrNot]
	A.7	Universal score (overall score for universal language) [BotOrNot]
	A.8	English score (overall score for English language) [BotOrNot]
	A.9	Overall score (average value between universal and English score) [BotOrNot]
	A.10	Query-wise aggregated anomaly score [BotWalk]

TABLE 4.2: Extracted features and their description.
$^{(*)}$ Statistics about content-based features refer to the *recent* activity of a user.

this prompted us to focus the evaluation of our framework on using either the full space of features or different subsets of feature categories.

Quality criteria

Most LTR methods minimize a *loss function*, which is related to standard IR measures for ranking evaluation [128], [179]. These are used as quality criteria in the optimization and/or evaluation of an LTR model. We will consider the most widely used criteria, namely $P@k$, MAP and $nDCG@k$.

Given a rank threshold k, *precision on the top-k* objects ($P@k$) is defined as the fraction of objects in the first k positions that are relevant to the query. Unlike $P@k$, *mean average precision* (MAP) considers also objects ranked after the k-th position: for each query q from a set Q of queries, it first computes $P@k$ with k corresponding to the rank position of each relevant object, then these precision values are averaged, and average precision values are further averaged over all queries:

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{4.6}$$

with $AP = \frac{1}{m}\sum_{k=1}^{n} P@k \cdot rel(k)$, where n and m are the number of retrieved objects and relevant objects, respectively, and $rel(k)$ denotes an indicator function that yields 1 if the object at rank k is relevant, zero otherwise.

Normalized discounted cumulative gain ($nDCG$) uses graded relevance labels to measure the goodness of the ranking of each object. It is defined as the ratio between the discounted cumulative gain (DCG) [180] to its ideal ranking, taking into account the top-k objects in the two rankings.

$$nDCG@k = \frac{DCG@k}{DCG_{max}@k} \tag{4.7}$$

with $DCG@k$ equal to:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel(i)} - 1}{\log_2(i+1)} \tag{4.8}$$

DCG is based on the assumption that highly relevant objects appearing in lower positions should be more penalized as the graded relevance value is reduced logarithmically proportional to the position in the ranking.

Framework setting

In our evaluation we included all the LTR methods discussed in Section 4.2. We used the Java implementations of *RankLib* under the *Lemur Project*, with default settings.[1] We trained and tested through 5-cross-validation each of the LTR methods using MAP, $P@k$, and $nDCG@k$, with $k \in \{10, 100\}$, both for optimization (except for RankNet) and evaluation. We defined $n = 20$ queries,

[1]http://www.lemurproject.org

each of which corresponding to $m = 1000$ instances. Also, we set to 15 minutes the width of the listening time-window in DeBot executions.

Evaluation goals

We pursue two main evaluation goals.

- *Performance on feature subsets:* We are interested in evaluating the usefulness and informative richness of *aggregate*, *user-based*, *network-based*, *content-based*, and *temporal* features sets w.r.t. the ability to characterize automated behavior in OSNs. We tested each group of features separately in order to evaluate the support provided in the characterization of automated behavior.

- *Performance on the whole feature space:* We want to assess the capability of our LTRSB framework to detect several automated behavioral patterns and its ability to leverage the whole feature space. Therefore, we evaluated its performance on all available data, using all the methods described in Section 4.2.

4.3.3 Experimental Results

We organize the presentation of experimental results according to the previously discussed evaluation goals. For each stage of evaluation, we will summarize results through *heatmaps*, with numerical detail on the performance scores corresponding to the various assessment criteria, by varying LTR method and setting. Note that, for the sake of readability, particularly to avoid cluttering of the figures, we have intentionally hidden entries in every heatmap that refer to optimal performance values (i.e., equal to 1).

Evaluation w.r.t. feature subsets

At a first glance looking at Figures 4.2–4.4, we observe a variegate situation, whereby the status of bot can be successfully determined only for some particular configurations.

Results corresponding to the balanced binary relevance setting (Fig. 4.2) show that Coordinate Ascent and LambdaMART can achieve good or near-optimal performance for most of the configurations of the assessment criteria and feature subsets. This would suggest their ability in the BB scenario to learn enough information for the construction of the ranking model from every subset of features. By contrast, there is an evident emergence for a more comprehensive feature space for methods like RankNet and AdaRank.

In general, it is not straightforward to recognize the most informative subset of features, already for the BB scenario which is expected to be the simplest among the three we devised. The variability of performance across feature groups seems to suggest that, due to the inherent difficulty of the detection task at hand, a single subset (i.e., type) of features is not sufficient to learn a proper ranking function capable of identifying different behavioral patterns.

Under the UB, we aimed to stress the LTRSB ability to learn from fewer examples of bots, thus making the ranking task more difficult. Results shown in Fig. 4.3 confirm the above-mentioned difficulty. Overall, quite poor performance characterizes RankNet, especially when using aggregate features only, and AdaRank, with user-based and network-based features only. LambdaMART and Coordinate Ascent show to be able to learn a ranking model that can successfully rank the top-10 instances, using any subset of features, though their performance in terms of all assessment criteria significantly decrease when considering top-100 or all instances.

As previously stated, the third scenario (Grad) corresponds to a balanced selection of relevant and non-relevant instances, with graded relevance labels derived from the bot probabilities computed by BotOrNot. We observe some difficulty in discerning between different levels of bot status; in particular, in terms of $nDCG$@10 and $nDCG$@100 criteria, almost all methods are affected by poor performance when only one subset of features is used. The only significant exception is represented by LambdaMART and Coordinate Ascent which, when trained on the subspace of aggregate features, are able to achieve optimal or very good ranking prediction accuracy. This might be explained since aggregate features includes highly descriptive features derived from BotOrNot as well as from BotWalk.

Evaluation w.r.t. all features

Figure 4.5 reports on LTRSB performance results on all datasets, exploiting the whole space of features, with different selections of relevant/non-relevant instances, and both graded and binary relevance labels.

One general remark is that, in most cases, LTRSB can achieve good performance, with any LTR method, especially in the case of balanced settings with both binary and graded relevance labels. Under the balanced binary relevance setting, small variations in terms of performance are observed between the methods. Performance results vary from 0.883 to 1, with two exceptions corresponding to AdaRank (0.578) and LambdaMART (0.67) according to MAP. For Grad, Coordinate Ascent and LambdaMART show to be robust w.r.t. the assessment criteria, with scores above 0.88 and, in most case, close or equal to 1. RankNet (resp. AdaRank) can also achieve optimal (resp. near-optimal) performance, according to P@10 and P@100, but perform poorly in terms of $nDCG$@10.

The unbalanced setting is characterized by a high variance across all assessment criteria and methods, with evidence of some difficulty in ordering instances belonging to the tail of the ranking solution; in particular, according to MAP and P@100, AdaRank, LambdaMART, and RankNet perform quite poor (i.e., < 0.5). Coordinate Ascent achieves best overall performance ranging from a minimum value of about 0.6 to the near optimal 0.957.

Discussion

Detecting and characterizing bot accounts in OSNs is a non-trivial learning problem. On the one hand, OSN platforms like Twitter only recently took more serious actions in order to ban, and better regulate bots within their boundaries. On the other hand, bot-masters are developing more sophisticated tools in order to better mimic human behaviors, thus increasing their potential to influence people, bias public opinion, and even pursue malevolent intents. For those reasons, it is strongly recommended to develop software systems that can exploit all the information available, and leverage useful "signals" of different type, in order to unveil suspicious bot activities.

Answering the research questions stated in the Introduction:

- Our proposed LTRSB framework is designed to learn from examples of bots and non-bots in order to detect and rank previously unseen bot accounts; when equipped with Coordinate Ascent or LambdaMART, LTRSB can achieve optimal ranking performance.

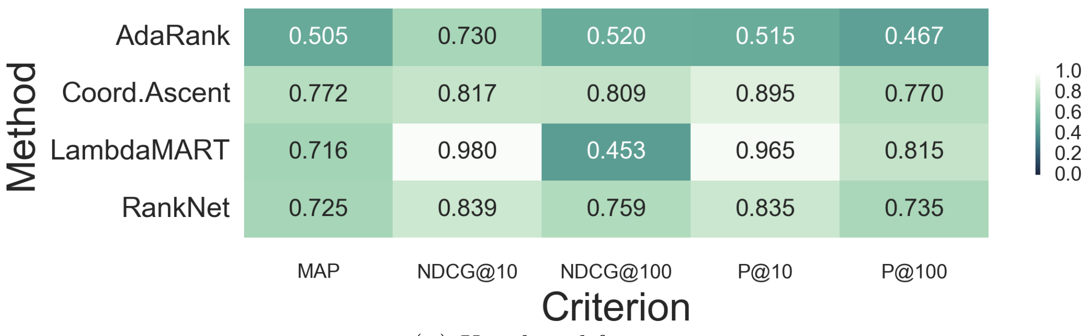
- LTRSB demonstrates that different methods for bot detection, developed upon different criteria and models, can be incorporated into a unifying machine-learning framework for determining the status of OSN accounts at various levels of bot status.

- LTRSB evaluation has shown that leveraging a large and heterogeneous space of features is beneficial and, in most cases, essential to effectively detect and rank bots, whose traits might in general refer to different behavioral patterns.

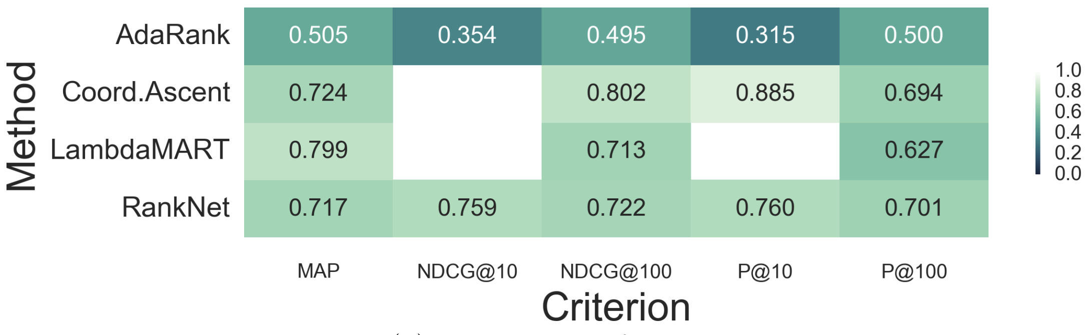
4.3.4 Conclusion

In this work, we advanced research on bot behavior analysis by developing a robust, supervised ranking model, leveraging different behavioral signals of bot activity. Our learning-to-rank framework, named LTRSB, exploits the most relevant existing methods on bot detection for enhanced feature extraction, and state-of-the-art learning-to-rank methods for the optimization. LTRSB was evaluated using ground-truth data, according to different assessment criteria.

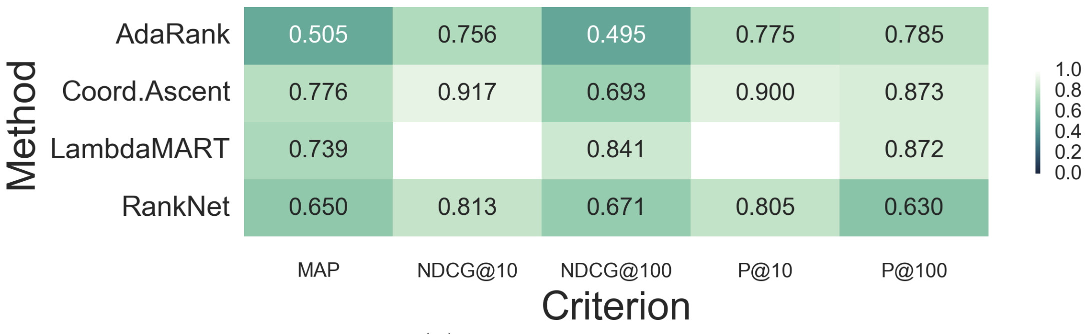
As future work, it would be interesting to evaluate LTRSB using queries and ground-truth data that correspond to multiple classes of bots, from harmless (e.g., advertisement bots, entertainment bots) to malicious bots (e.g., spambots, scraper bots).



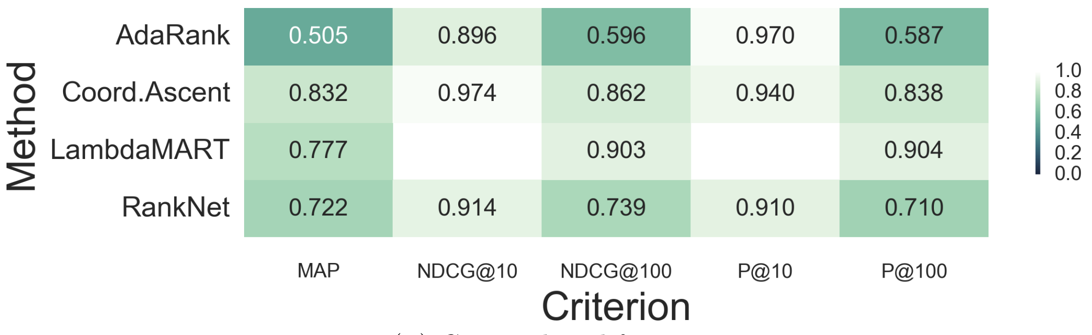
(A) User-based features



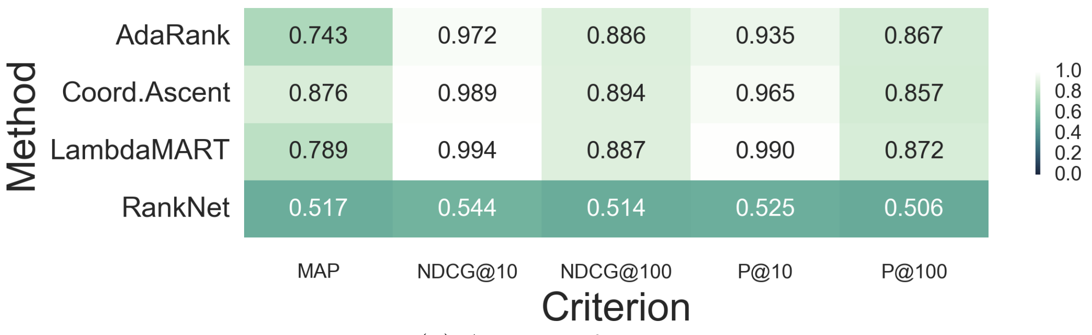
(B) Network-based features



(C) Temporal features

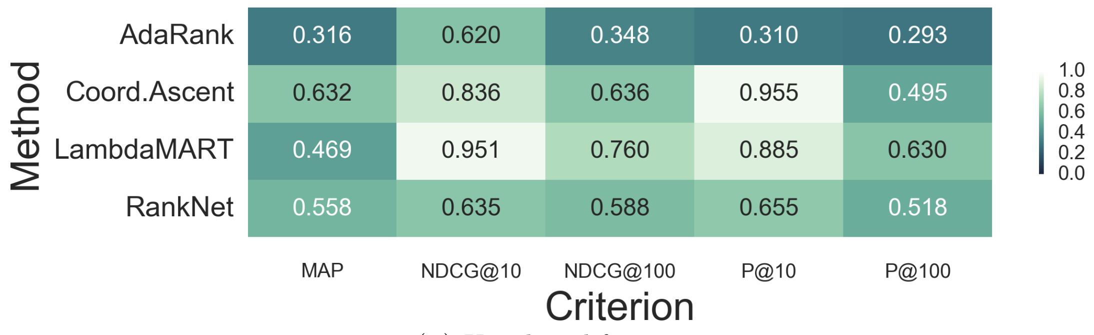


(D) Content-based features

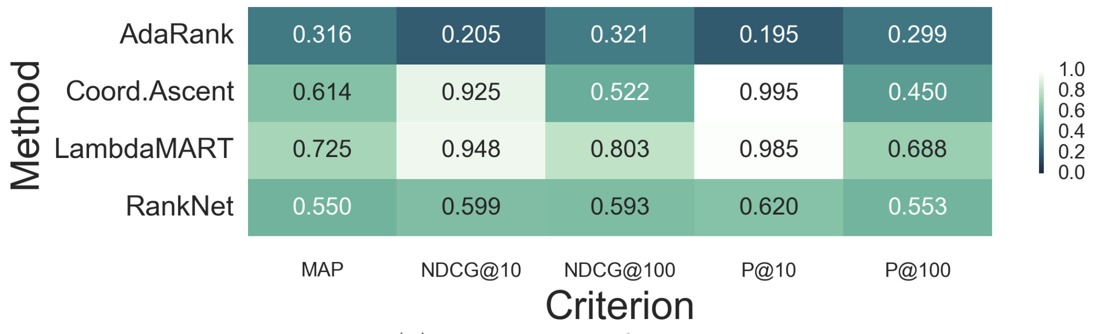


(E) Aggregate features

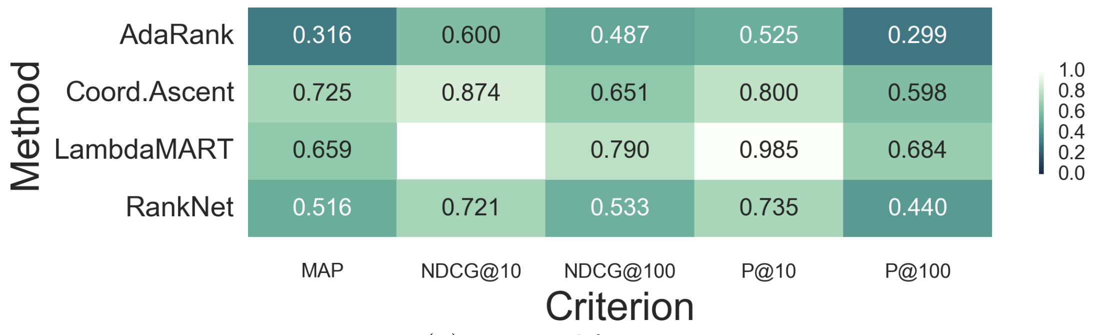
FIGURE 4.2: LTRSB performance on all data, with balanced binary relevance (BB), and using different feature subsets.



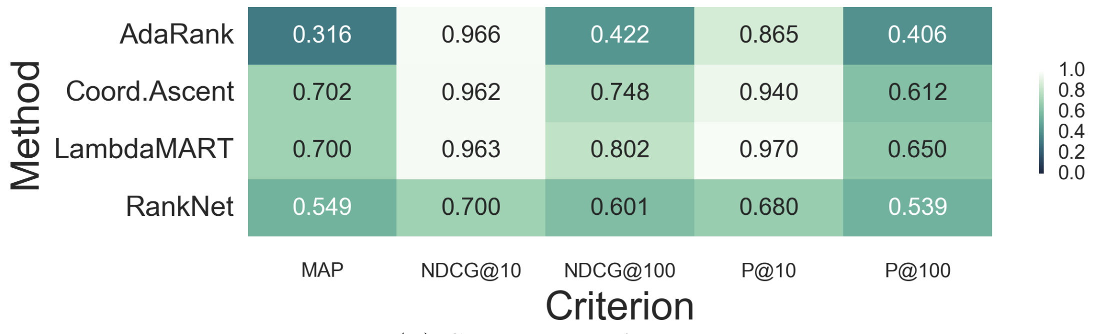
(A) User-based features



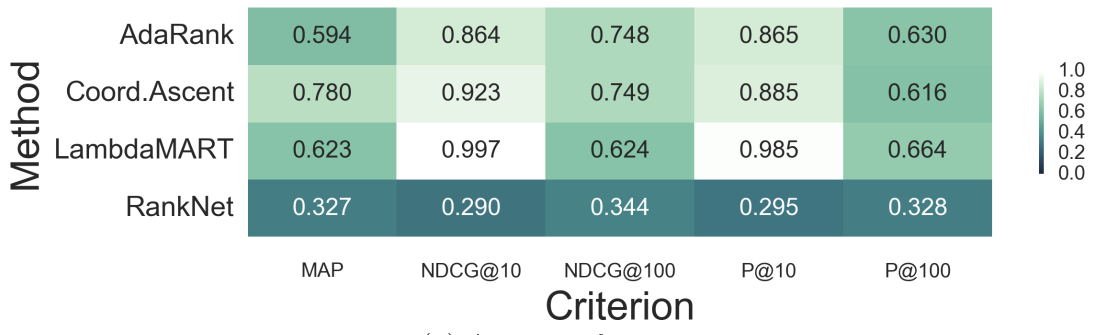
(B) Network-based features



(C) Temporal features

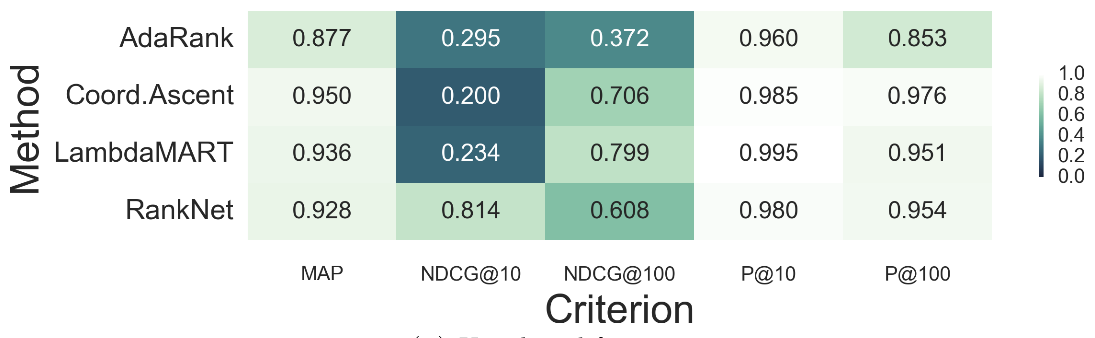


(D) Content-based features

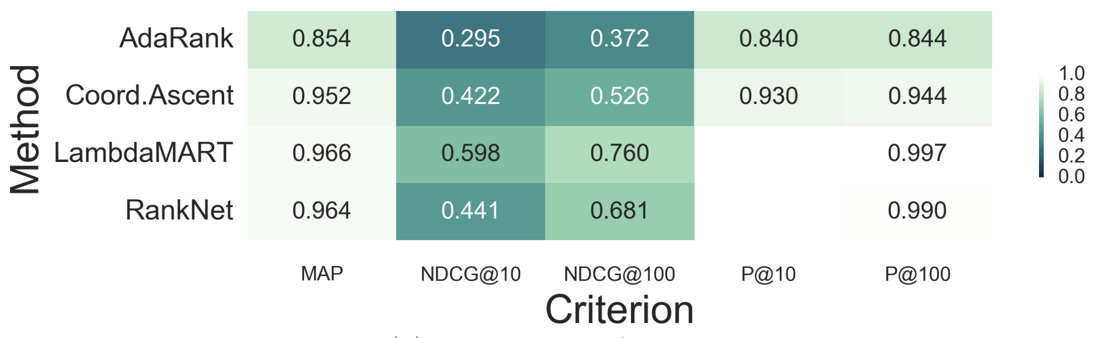


(E) Aggregate features

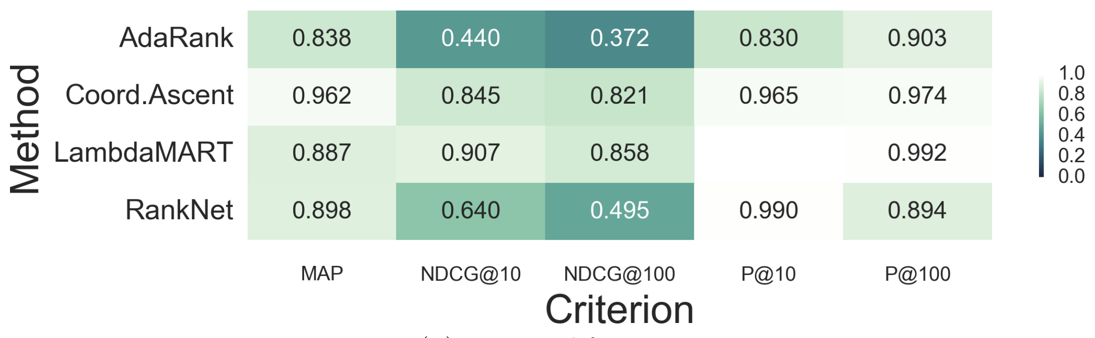
FIGURE 4.3: LTRSB performance on all data, with unbalanced binary relevance (UB), and using different feature subsets.



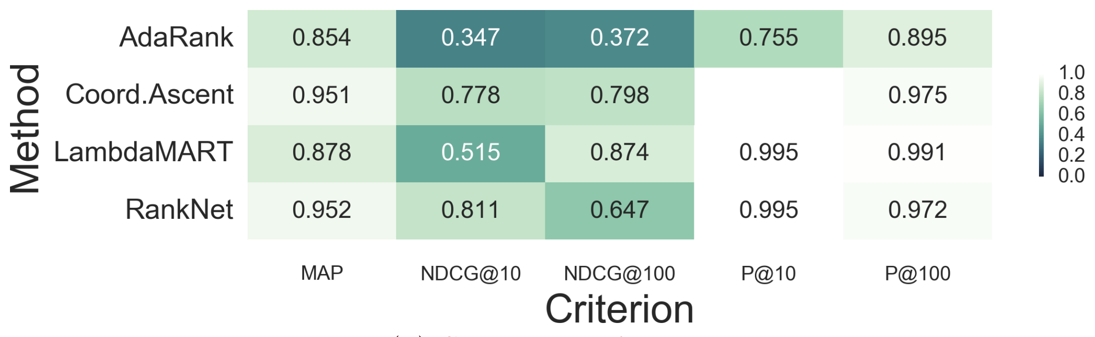
(A) User-based features



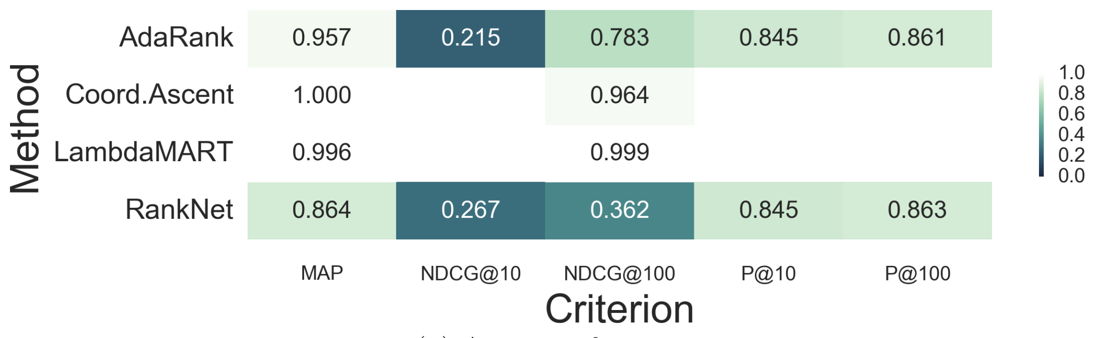
(B) Network-based features



(C) Temporal features

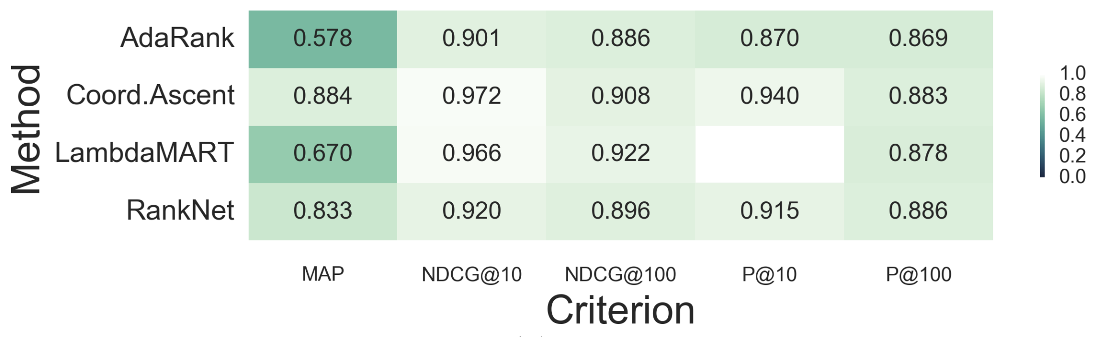


(D) Content-based features



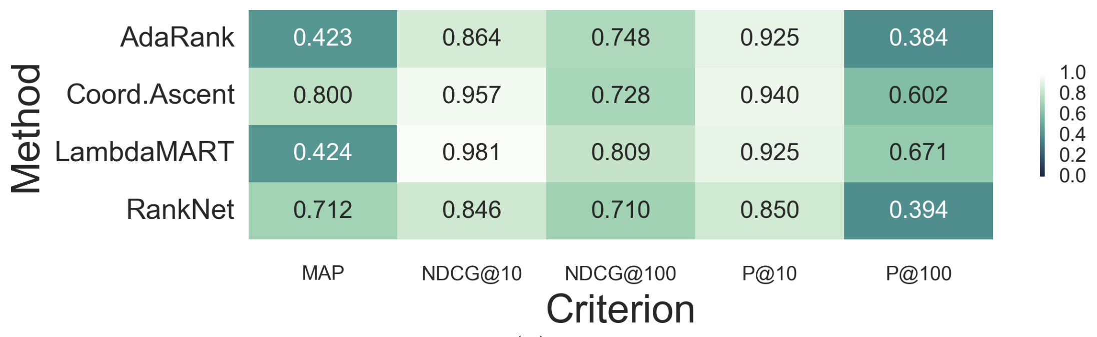
(E) Aggregate features

FIGURE 4.4: LTRSB performance on all data, with graded relevance setting (Grad), and using different feature subsets.



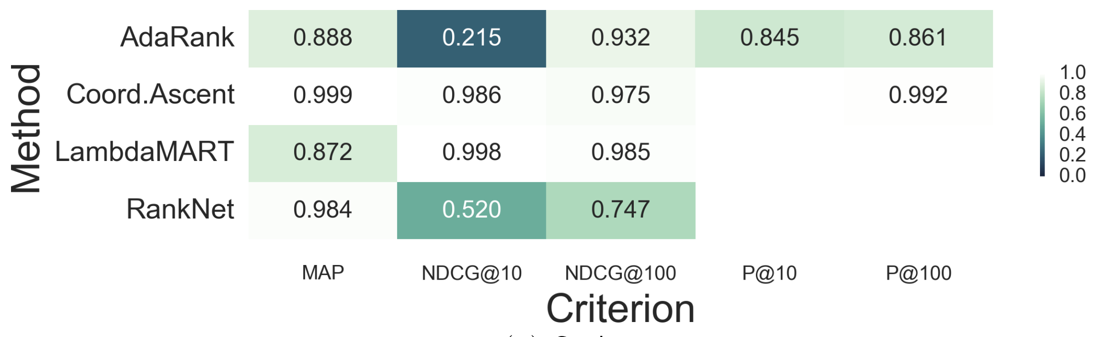
Method	MAP	NDCG@10	NDCG@100	P@10	P@100
AdaRank	0.578	0.901	0.886	0.870	0.869
Coord.Ascent	0.884	0.972	0.908	0.940	0.883
LambdaMART	0.670	0.966	0.922		0.878
RankNet	0.833	0.920	0.896	0.915	0.886

(A) BB



Method	MAP	NDCG@10	NDCG@100	P@10	P@100
AdaRank	0.423	0.864	0.748	0.925	0.384
Coord.Ascent	0.800	0.957	0.728	0.940	0.602
LambdaMART	0.424	0.981	0.809	0.925	0.671
RankNet	0.712	0.846	0.710	0.850	0.394

(B) UB



Method	MAP	NDCG@10	NDCG@100	P@10	P@100
AdaRank	0.888	0.215	0.932	0.845	0.861
Coord.Ascent	0.999	0.986	0.975		0.992
LambdaMART	0.872	0.998	0.985		
RankNet	0.984	0.520	0.747		

(C) Grad

FIGURE 4.5: LTRSB performance on all data and whole feature space, using different relevance labeling settings.

4.4 Learning to rank Lurkers

Lurking in an online social network (OSN) characterizes those users in the crowd who do not significantly take an active and tangible role in the interaction with other members [85], [86], [181], [182]. Such users are referred to as *lurkers*, since they gain benefit from information produced by others, though their presence is legitimated [87], expected and even welcome [101], [183]. More importantly, lurkers might hold potential *social capital*, because they acquire knowledge from the OSN: by observing user-generated communications, they can form their own opinions and even expertise, though they rarely will let other people know their "value". Therefore, it might be desirable to make lurkers' social capital available to other users [184]. This can be accomplished through mechanisms of *engagement* [102], [103], [185]–[187], ultimately encouraging lurkers to more actively participate in the OSN life.

In [97], [110], the authors presented the first tool, namely *LurkerRank*, for automatically bringing order into the crowd of users that may show lurking behaviors at varying degrees. LurkerRank adopts a query-independent, eigenvector-centrality based approach that utilizes the link graph structure underlying user relationships (e.g., followships, like/comment interactions). Its ranking solution can enable a way to prioritize the engagement of (top-ranked) lurkers. LurkerRank has also been adapted as query-dependent (e.g., trust-biased) ranking method [97], [112], extended to handle time-evolving networks [111], or exploited for targeting users in an influence maximization task [188].

Research questions. Nevertheless, several challenges remain open, since the complexity of lurking behaviors hints at the opportunity of using any available, possibly platform-specific information on the activity and interaction of lurkers in an OSN. In particular, the following questions may arise:

- *Can we incorporate into a lurker ranking model various "signals" that can be used as behavioral features, upon which the evaluation of the degree of lurking of any user w.r.t. a given context (i.e., online environment) is performed?*
- *Can we enable a lurker ranking model to accurately be tuned by exploiting past lurking experiences?*
- *What if information on lurking behaviors is available from populations of different networks?*
- *Can we predict the lurking behavior of a previously unexamined user by avoiding to rank from scratch all users?*

The LTR opportunity. In this work we aim at answering the above questions by developing a principled, machine-learning-based ranking framework for the analysis of lurking behaviors. *Learning-to-rank* (LTR), i.e., using machine learning to automatically build a ranking model, has become one of the key technologies for modern web search, retrieval and personalization [128], [129]. As in traditional ranking functions, a partial ordering (ranking) is provided for a set of objects, according to their degree of relevance to a given query; in LTR, the ranking function is learned from *training data* given in the form of ⟨*query, object, relevance label*⟩ tuples.

Motivations. We believe that LTR is particularly suitable for addressing the aforementioned research questions, thus helping us to improve our understanding of a behavioral analysis problem that is inherently dynamic as well as sensitive to many aspects. In fact, LTR adopts a supervised approach to learn from past user experiences, which might be annotated according to the degree of lurking behavior. LTR training is accomplished according to a set of features which, being possibly of different types, would capture different aspects that can be useful predictors of lurking behaviors.

Due to learning from different, properly engineered features, LTR is a robust approach in cases where it would be difficult to gather sufficient and significant lurking information from the graph of user relationships: this would lead to a weak or sparse link structure, which negatively affects the effectiveness of any graph-based lurker ranking method. In addition, by inheriting robustness to outliers of some machine learning methods, LTR models can effectively handle inconsistent or false lurking behaviors (e.g., those shown by users that are totally inactive in the online platform). To a more general extent, by treating multiple OSN platforms as different queries and building an LTR model from instances (users) that belong to the various platforms, lurker ranking results can be compared through different OSNs, thus generalizing the concept of lurking to multiple networks.

Last, but not least, LTR offers unprecedented opportunities for incremental lurking scenarios: once trained a learning-to-lurker-rank model, this can be used to assign any previously unobserved user with a lurking score, thus avoiding the calculation from scratch of a ranking scheme over the entire OSN graph.

Contributions. In this work, we propose the first LTR framework to analyze lurkers in OSNs. Note that LTR has been previously used to address behavioral problems, in particular user engagement. For instance, the study in [173] aims at ranking tweets by the amount of user participation in them; however, identifying and ranking lurkers is a more complex problem, which requires proper modeling of users and their behavioral features.

By exploiting state-of-the-art LTR methods, we develop a *learning-to-lurker-rank* methodology which exploits the unsupervised LurkerRank method to derive binary or graded relevance labels, and a feature engineering phase that accounts for relational (i.e., topology-based), media-based, activity rate, and platform-specific information on the commitment and interaction of users from different OSNs. Results obtained on 23 network datasets, which were built from 7 OSNs, have shown the significance of our approach, confirming our initial hypothesis that LTR can be effective for ranking lurking behaviors.

4.4.1 Lurker ranking

In [97], [110], the authors have originally brought the concept of centrality in the context of lurking behavior analysis, with the goal of characterizing and ranking

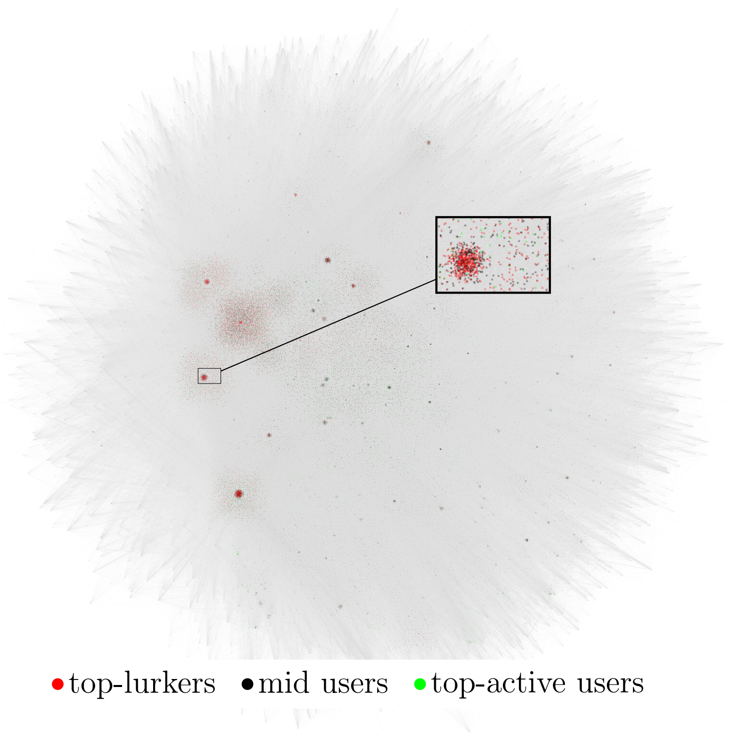


FIGURE 4.6: Localization of lurkers and non-lurkers in Twitter professional-oriented followship graph. *(Best viewed in color)*

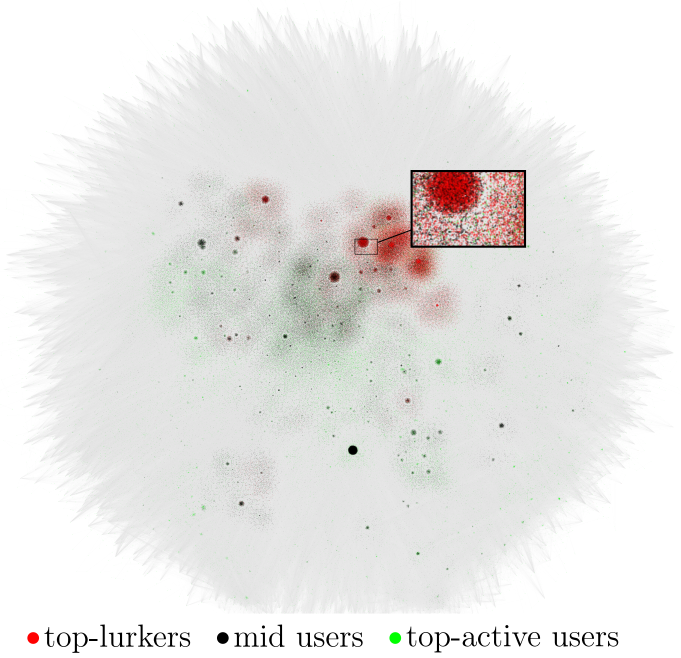


FIGURE 4.7: Localization of lurkers and non-lurkers in Twitter-Higgs followship graph. *(Best viewed in color)*

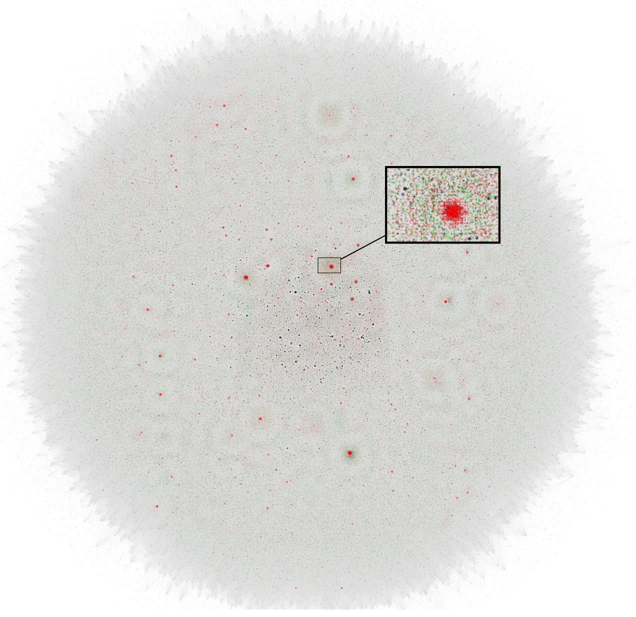


FIGURE 4.8: Localization of lurkers and non-lurkers in Flickr like-based interaction graph. *(Best viewed in color)*

the users according to their degree of lurking in an OSN. Following an unsupervised learning paradigm, the lurker ranking models utilize only the topology information of an OSN (like PageRank and other classic ranking methods), which is seen as a directed graph where any edge (u, v) means that v is "consuming" or "receiving" information from u. Upon the assumption that lurking behaviors build on the *amount of information a node consumes*, the key intuition is that the strength of a user's lurking status can be determined based on three basic principles, which are here informally summarized as follows: *overconsumption* (i.e., the excess of information-consumption over information-production, estimated proportionally to the in/out-degree ratio of a node), the *authoritativeness of the information received from the in-neighbors*, and the *non-authoritativeness of the information produced, i.e., sent to the out-neighbors*. These principles form the basis for three ranking models, whereby a complete specification was provided in terms of PageRank and alpha-centrality based formulations. For the sake of brevity here, and throughout this chapter, we will refer to only one of the formulations described in [97], [110], which is that based on the *in-out-neighbors-driven lurker ranking*, hereinafter referred to as *LurkerRank* (LR).

Given a directed graph $G = (V, E)$, and a node $v \in V$, let $N^{in}(v)$ and $N^{out}(v)$ denote the set of in-neighbors and the set of out-neighbors of v, respectively. For any node v, the *LurkerRank* score $\mathsf{LR}(v)$ is defined as follows [97]:

$$\mathsf{LR}(v) = \alpha[\mathcal{R}^{\text{in}}(v)\ (1 + \mathcal{R}^{\text{out}}(v))] + (1 - \alpha)p(v) \tag{4.9}$$

where $\mathcal{R}^{\text{in}}(v)$ is the in-neighbors-driven lurking function:

$$\mathcal{R}^{\text{in}}(v) = \frac{1}{|N^{out}(v)|} \sum_{u \in N^{in}(v)} \frac{|N^{out}(u)|}{|N^{in}(u)|} \mathsf{LR}(u) \tag{4.10}$$

and $\mathcal{R}^{\text{out}}(v)$ is the out-neighbors-driven lurking function:

$$\mathcal{R}^{\text{out}}(v) = \frac{|N^{in}(v)|}{\sum_{u \in N^{out}(v)} |N^{in}(u)|} \sum_{u \in N^{out}(v)} \frac{|N^{in}(u)|}{|N^{out}(u)|} \mathsf{LR}(u) \tag{4.11}$$

Above, $p(v)$ is the value for v in the personalization vector, which is by default set to $1/|V|$, and α is a damping factor ranging within [0,1], usually set to 0.85.

It should be noted that, although the principles underlying LR are generic and applicable to any OSN, the particular type of social context managed by the online platform, the semantics of user relations therein, and the structural properties of the corresponding graph(s) modeling the user relations in the platform, can have different impact on the understanding of lurking behaviors, and consequently on how lurkers relate themselves to the other (i.e., active) users.

Figure 4.8 shows three examples of social network graphs, in which different colors are assigned to nodes in order to distinguish between lurkers and users that can be regarded as moderately or highly active in the corresponding network. For every network, we first computed the LurkerRank score for each node, then we determined the weight of each edge (u, v) proportionally to the fraction of the original lurking score of v given by its in-neighbor u, exploiting the notion of influence probability defined in [188]. Node colors are chosen according to the quartile of membership in the LurkerRank ranking solution: nodes belonging to the 3rd quartile, i.e., the top-25% lurkers, are colored in red, whereas nodes in the 1st quartile, i.e., the top-25% active users, are colored in green, and the remaining nodes (mid users) in black. We generated the graph plots using the force-directed layout OpenOrd [189], which is well-suited to display the formation of agglomerations around nodes that, in our setting, receive incoming connections with higher weights. Looking at the figure, lurkers can be differently influenced by few or many groups of active users, presumably depending on the popularity and appealingness of the topics discussed. The leftmost graph plot represents a Twitter subnetwork of developers (which also have an account on GitHub and StackOverflow; cf. Sect. 4.4.2). Here, the localization of lurkers tends to be around a few groups of the developers, depending on the attractiveness of the software development project the active users are working on. The middle graph plot corresponds to a portion of Twitter focused on a single main theme, i.e., the announcement of the Higgs boson. Compared to the previous example graph, lurkers tend even more to agglomerate with only some of the active users, the latter presumably being the most visible (i.e., popular) users that are active in the spreading of information about Higgs boson. The rightmost graph plot corresponds to favorite/like-based interactions of users in Flickr. Here we observe many micro-agglomerations that imply a quite diversified configuration in terms of localization of lurkers and active users, due

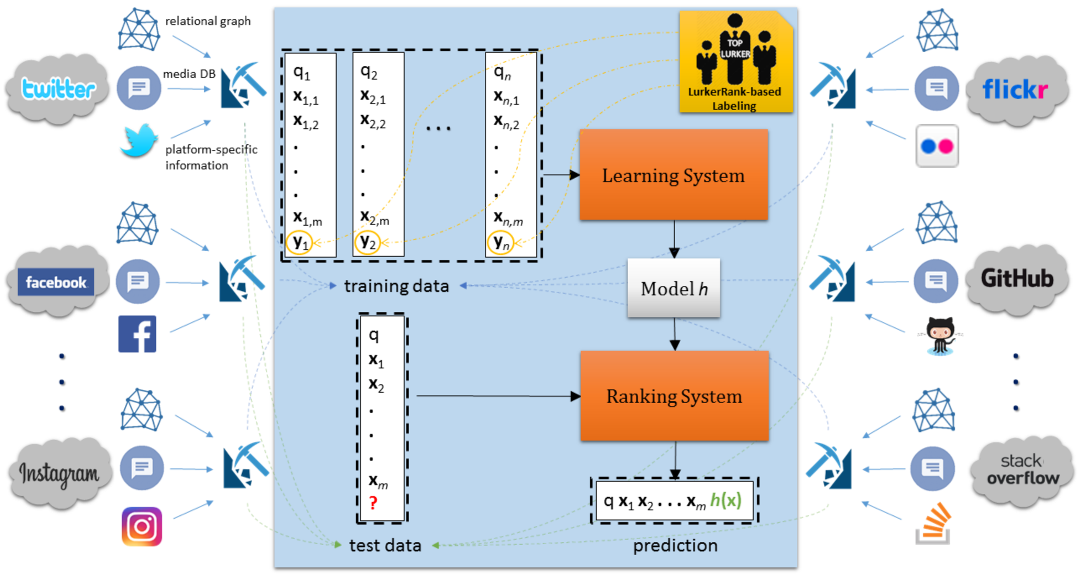

relational graph
media DB
platform-specific information
twitter
facebook
Instagram
flickr
GitHub
stack overflow
TOP LURKER
LurkerRank-based Labeling
Learning System
Model h
Ranking System
training data
test data
prediction
q_1 $x_{1,1}$ $x_{1,2}$ $x_{1,m}$ y_1
q_2 $x_{2,1}$ $x_{2,2}$ $x_{2,m}$ y_2
q_n $x_{n,1}$ $x_{n,2}$ $x_{n,m}$ y_n
q x_1 x_2 x_m ?
$q\ x_1\ x_2 \ldots x_m\ h(x)$

FIGURE 4.9: Illustration of our proposed learning-to-lurker-rank (LTLR) framework

to higher heterogeneity of the topics discussed in the network (i.e., photo-driven themes); moreover, according to previous studies on lurking behaviors in scenarios of least-effort types of interaction [111], the difference between lurkers and active users is much less evident than in two other graphs.

The above-presented example points out an emergence for enhancing a generic, topology-based ranking model for lurking behaviors by leveraging the peculiarities of user relations and media-based properties that are specific to the OSN under examination. Within this view, in this work we take an opportunity for the development of a new framework that is able to learn from user behavioral features and past pieces of evidence of lurking states, over multiple OSNs, to predict and rank previously unseen lurking states.

4.4.2 Learning to lurker rank framework

In this section we describe our proposed *learning-to-lurker-rank* (LTLR) framework. Figure 4.9 provides a schematic illustration, which sheds light on the main modules and data flows involved in the framework. In this regard, we assume that a number of social media sources are available (e.g., Twitter, Instagram, etc.), from which three types of information are extracted and organized into three main components, namely user-relational (i.e., topological) network data, media data, and platform-specific meta-data. These are processed to build the feature space for the representation of the training data. The learning-to-rank system is designed to work with training data instances, each of which is annotated with a relevance label expressing the degree of lurking of the corresponding user (module LurkerRank-based labeling, on the top-right corner of the internal box shown in the figure). In the following, we elaborate on each of the main aspects of the proposed LTLR framework.

network	**#users**	**#edges**	**types of user relation**
Flickr-s1	36 362	261 192	timestamped favorites/like
Flickr-s2	111 281	1 178 865	timestamped favorites/like
Flickr-s3	212 031	2 642 354	timestamped favorites/like
Flickr-s4	388 667	5 200 581	timestamped favorites/like
Flickr-s5	546 143	7 788 333	timestamped favorites/like
FriendFeed	493 020	19 153 367	like and comment
GitHub	37 829	443 792	induced followship
GooglePlus	13 673 251	13 673 251	followship
Instagram-t1	43 925	536 370	topic-biased comments relations
Instagram-t2	33 548	314 496	topic-biased comments relations
Instagram-t3	30 384	152 590	topic-biased comments relations
Instagram-t4	31 458	187 025	topic-biased comments relations
Instagram-t5	44 915	745 604	topic-biased comments relations
Instagram-t6	33 071	325 369	topic-biased comments relations
Instagram-t7	33 797	278 081	topic-biased comments relations
Instagram-t8	41 476	624 489	topic-biased comments relations
Instagram-t9	44 484	696 956	topic-biased comments relations
StackOverflow	22 507	88 277	Q&A help
Twitter	52 126	2 412 523	followship
Twitter-Higgs-soc	456 627	14 855 842	followship
Twitter-Higgs-mt	116 409	150 818	mention
Twitter-Higgs-rp	38 919	32 523	reply
Twitter-Higgs-rt	256 492	328 132	retweet

TABLE 4.3: User relational characteristics in our evaluation networks

Datasets

We built our evaluation datasets from 23 user-relation networks and media databases gathered from Twitter (5), Instagram (9), Flickr (5), FriendFeed, GooglePlus, StackOverflow, and GitHub. Table 4.3 summarizes main characteristics relating to the size of the networks and type of user relations. Note that our selection of network data is quite diversified both in terms of number of users involved (ranging from a few tens of thousands, in StackOverflow, to several million, in GooglePlus) and types of user relations, which vary from explicit or induced followship to least-effort interactions (e.g., favorites, likes) and most-effort actions like comments and answers. Moreover, our network data embed information about the users' participation, in terms of both production and consumption of social content, reputation, and influence. These will be exploited to generate features of different types, as we shall describe later in this section. It should also be noted that all of the selected networks are publicly available and have been used in early works on lurking analysis [97], [111], [190]; in addition, user behaviors in StackOverflow and GitHub datasets, and in the Twitter-Higgs networks, were originally studied in [121] and [122], respectively.

Training data

LTR training data consists of triplets ⟨*query, object, relevance label*⟩. In our setting, any query corresponds to one of the 23 networks above introduced, any object corresponds to a user's feature vector, and the relevance label denotes one or several grades of lurking, so that the higher grade a user has, the more lurker the user is.

Relevance labeling We used LurkerRank [97], [110] (cf. Section 4.4.1) to produce lurking scores upon which we derived the relevance labels in the training data. Basically, users at *top-p%* ranked and *bottom-q%* ranked portions of a LurkerRank solution were labeled as *relevant* (i.e., lurkers) and *non-relevant* (i.e., not-lurkers), respectively. Besides binary labeling, we also considered graded relevance by a finer-grain segmentation of the selected sublist of lurkers, as we shall discuss later in Section 4.4.2.

Feature engineering

Table 4.4 provides a concise description of the features extracted from our evaluation data. We organize them into four categories: *relational*, *media-based*, *activity-rate*, and *platform-specific*.

Relational features correspond to local properties of a node in a graph (i.e., user in a network), which we devise according to the three principles at the basis of the *topology-driven lurking model* proposed in [97], [110], previously discussed in Section 4.4.1. Within this view, relational features might include information about a node's in-neighbors, out-neighbors, and their ratio (which would reflect the lurking principle of overconsumption).

Media-based features describe actions that a user performed or underwent in relation to her/his media in the OSN (e.g., posts, images). We advise that one convenient way to organize such features is based on whether these actions are perceived as manifestations of a user's active participation or passive participation, i.e., we might distinguish between *latent* actions (e.g., views/clicks) and *visible* actions; the latter, in turn, can be divided into *least-effort* actions (e.g., likes, favorites, mentions), and *most-effort* actions (e.g., comments, replies). Accordingly, to extract media-based features, we might include the number of actions performed or underwent, for each of the action types. Moreover, it would be important to measure some notion of user's influence based on the amount of endorsement shown by other users in relation to the media contents produced.

Activity rate features are aimed to capture temporal aspects of a user's activity, in terms of frequency as well as latency [111]. Therefore, features of this type are expected to include information about the length of the time period between two temporally consecutive actions, or the average frequency of media created, or latent/least-/most-effort actions performed.

The fourth category of features we devise concerns *platform-specific* indicators of the degree of a user's commitment and engagement to the social life in the online platform. For instance, features falling into this category might include information relating badges or other forms of rewards for the user, the

	id	description
relational	R.1	#in-neighbors
	R.2	$(1 + \#\text{out-neighbors})^{-1}$
	R.3	#in-neighbors / #out-neighbors
	R.4	$(1 + \#\text{in-neighbors of a user's in-neighbors})^{-1}$
	R.5	#out-neighbors of a user's in-neighbors
	R.6	#out-neighbors / #in-neighbors of a user's in-neighbors
	R.7	#in-neighbors of a user's out-neighbors
	R.8	$(1 + \#\text{out-neighbors of a user's out-neighbors})^{-1}$
	R.9	#in-neighbors / #out-neighbors of a user's out-neighbors
media based	M.1	#latent actions performed
	M.2	$(1 + \#\text{least-effort actions performed})^{-1}$
	M.3	$(1 + \#\text{most-effort actions performed})^{-1}$
	M.4	(#latent actions performed)/(1 + #least-effort actions performed)
	M.5	(#latent actions performed)/(1 + #most-effort actions performed)
	M.6	Inv. exp. of *EI* defined on least-effort actions received for media created
	M.7	Inv. exp. of *EI* defined on most-effort actions received for media created
activity-rate	A.1	Mean per-day #latent actions performed
	A.2	$(1 + \text{mean per-day \#least-effort actions performed})^{-1}$
	A.3	$(1 + \text{mean per-day \#most-effort actions performed})^{-1}$
	A.4	A.1 * A.2
	A.5	A.1 * A.3
	A.6	Mean #days between two consecutive actions of new content production
	A.7	$(1 + \text{mean \#days between two consecutive actions of content consumption}))^{-1}$
platform-specific	P.1	$(1 + \#\text{times the user was tagged})^{-1}$ [Instagram]
	P.2	$(1 + \#\text{days in a row of current active contribution})^{-1}$ [GitHub]
	P.3	#days in a row of last active contribution [GitHub]
	P.4	$(1 + \#\text{days of the longest active contribution})^{-1}$ [GitHub]
	P.5	$(1 + \text{reputation score earned})^{-1}$ [StackOverflow]
	P.6	$(1 + \text{top-overall percentage})^{-1}$ [StackOverflow]
	P.7	$(1 + \#\text{people reached})^{-1}$ [StackOverflow]
	P.8	$(1 + \#\text{profile views})^{-1}$ [StackOverflow]
	P.9	$(1 + \#\text{gold badges received})^{-1}$ [StackOverflow]
	P.10	$(1 + \#\text{silver badges received})^{-1}$ [StackOverflow]
	P.11	$(1 + \#\text{bronze badges received})^{-1}$ [StackOverflow]
	P.12	$(1 + \#\text{top answers given})^{-1}$ [StackOverflow]
	P.13	$(1 + \#\text{lists the user belongs to})^{-1}$ [Twitter]
	P.14	$(1 + \#\text{tweets})^{-1}$ [Twitter]

TABLE 4.4: Extracted features and their description

number of lists a user belongs to, or the length of time period for intensively active contribution.

)

Framework setting

In our evaluation we included all the LTR methods discussed in Sect. 4.2. We used the Java implementations of *RankLib* under the *The Lemur Project*, with default settings.[2] We trained and tested through 5-cross-validation each of the LTR methods using MAP, $P@k$, and $nDCG@k$, with $k \in \{10, 100, 1000\}$, both for optimization (except for RankNet) and evaluation.

From our built evaluation datasets (cf. Table 4.3), we devised three cases depending on the selection of relevant/non-relevant instances:

- In the first case, dubbed BB, we considered *binary* relevance, i.e., lurking and not-lurking, and performed a *balanced* selection of relevant and not-relevant users.
- In the second case, dubbed UB, we performed unbalanced selection of relevant/not-relevant.
- In the third case, dubbed Grad, we considered m grades of relevance of lurking users. We will present results based on the setting of BB corresponding to top-5% and bottom-5% users by LurkerRank as relevant and not-relevant, resp., the setting of UB corresponding to top-5% and bottom-25% users by LR as relevant and not-relevant, resp., and the setting of Grad with $m = 3$ corresponding to top-5% (relevance 3), second 5% (relevance 2), third 5% (relevance 1) as relevant and bottom-25% as not-relevant.

Table 4.4 provides a concise description of the features extracted from our evaluation datasets. The first group of features (i.e., from R.1 to R.9) describes network-topology-based indicators of user relations, whose strength would proportionally increase with the lurking status of a user. Features from M.1 to M.7 capture the user's behavior from the perspective of the production and consumption of media. Note that, except for M.6 and M.7, we chose inverse linear function (for those features requiring inverse proportionality to a certain statistic count) for smoother numerical decrease. Symbol EI in features M.5–M.7 stands for *empirical influence*, a measure used to estimate a user's influence based on the amount of information s/he produced and that her/his followers have stated to endorse (e.g., liked, replied, etc.) [97], [191]. The third group of features contains measurements of the production and consumption activities of a user, including aggregate indicators such as the mean per-day ratio between latent and least-effort (resp. most-effort) actions. The last group of features refers to a number of aspects, including user's reputation, popularity, and activity, that are specific to a single-platform. Finally, it should be noted that to ensure comparability across queries, all feature values were scaled between 0 and 1, through min-max normalization.

Mapping the selected datasets (Table 4.3) with the features in Table 4.4, we derived the following subsets of data:

1. **D1**: Flickr networks are associated with relational features (i.e., from R.1 to R.9) plus M.2, M.6, A.2, and A.7.
2. **D2**: Flickr and Instagram networks are associated with relational features plus A.2 and A.7.

[2]http://www.lemurproject.org

3. **D3**: Flickr and FriendFeed networks are associated with relational features plus M.2 and M.6.

4. **D4**: GitHub network is associated with relational features plus P.2, P.3, and P.4.

5. **D5**: StackOverflow network is associated with relational features plus P.2, and from P.5 to P.12.

6. **D6**: Twitter networks are associated with relational features plus P.13.

Feature informativeness

We investigated about the usefulness or informativeness of the selected features, with a twofold goal: to discover possible correlations between features (possibly w.r.t. the lurking-relevance class attribute), and to estimate their impact on the ranking prediction performance. For this purpose, we resorted to five standard methods used in feature selection tasks: principal component analysis (PCA), information gain, Pearson's correlation coefficient, correlation-based feature selection (CFS) [178], and learner-based feature selection exploiting J48 decision tree model.

Using PCA, we found that in most cases the first principal component is a linear combination of relational features (among which, R.2, R.5, and R.8 are often characterized by the highest variance), however several exceptions are present. For instance, on StackOverflow data, the built-in reputation system (i.e., features P.5, P.9, P10, and P.11) corresponds to the first principal component, while the subsequent components consist of various linear combinations of mostly relational features. On Flickr data, the first principal component is a combination of non-relational (M.2 and M.6) and relational features (R.1–R.9).

By employing the information gain method, unlike in PCA, we could take into account the class of relevance, both for the binary and graded relevance settings. As expected, since that the constituent parts of the defining principles of LR are topology-driven, the most informative features w.r.t. to the lurking relevance labels are relational as well (in particular, from R.1 to R.8), for all the analyzed networks. Nevertheless, non-relational features such as M.6, A.6, and P.14 (which are of media-based, activity-rate, and platform-specific type, respectively) are the subsequent most informative features.

Similarly, an analysis based on Pearson's correlation recognizes relational features as the most relevant, followed by the number of tweets posted and the number of lists a user belongs to (P.13 and P.14) in Twitter, the reputation system and medals present in StackOverflow (P.5, P.9, P.10, P.11), the number of days in a row of active contribution and the number of least-effort actions performed (P.3, M.2) in GitHub. As concerns the CFS method, while being consistent with the other methods regarding the relational features, its selection of most informative non-relational features includes A.6, for the setting of graded relevance label, and P.5, M.3 for the binary settings. Finally, J48 confirms the overall scenario described by the other methods, where relational features (in particular, R.3, R.5, and R.6) strongly correlate with the lurking-relevance class, followed by M.6 and P.9.

Pearson's correlation coefficient also deals with both the class of relevance and the feature under examination, it treats them like random variables while measuring the linear correlation between them.

To sum up, relational (i.e., topological) features revealed to take an essential role in predicting the lurking status of users, although different subsets of non-relational were found as equally informative. This prompted us to focus our empirical evaluation of LTLR on two main settings: the one with relational features only, or non-relational features only, and the other one corresponding to the full space of features.

Evaluation goals

We organize the presentation of experimental results according to the following evaluation goals.

- *Performance on followship network data, with relational features:*

  With the intent of setting an initial benchmark we tested our LTLR framework on followship network data represented over the space of relational features only (Section 4.4.3), which are the primary features as they correspond to the constituent factors of the topology-driven lurking notion upon which the LurkerRank method is built.

- *Performance on heterogeneous data:*

  In order to assess the capability of the LTLR framework to generalize the lurking behavior across different domains and its ability to leverage the whole features space we evaluate its performance on all available data, corresponding to the various evaluation networks and represented in the full space of features (Section 4.4.3).

- *Relevance of non-relational features:*

  We were interested in evaluating the usefulness and informative richness of *media-based*, *activity-rate* and *platform-specific* features sets w.r.t. the ability to predict lurking behavior in OSNs. We focused on a comparison between the use of *non-relational features* only (which may vary from platform to platform) and *all features* together for the training data (Section 4.4.3).

- *Performance on distinct networks:*

  We also investigated whether the ranking model learned from a specific platform, could be successfully applied to another network sharing the same feature space. To verify and evaluate this ability, we selected the data subsets composed of at least two networks, and for each subset, we trained all the LTR algorithms on one network and tested on the other, and vice-versa (Section 4.4.3).

4.4.3 Experimental results

We organize the presentation of experimental results according to the previously discussed evaluation goals. For each stage of evaluation, we will summarize results through heatmaps with numerical detail on the performance scores corresponding to the various assessment criteria, by varying LTLR method and setting. Note that, for the sake of readability, particularly to avoid cluttering of the figures, we have intentionally hidden entries in every heatmap that refer to optimal performance values (i.e., equal or very close to 1).

Evaluation on followship network data with relational features only

In the first stage of analysis, we tested our framework on followship networks only, i.e., GitHub, GooglePlus, and Twitter networks, focusing on relational features. Results are shown in Fig. 4.10. One general remark is that in many cases LTLR can achieve optimal or near-optimal performance. More specifically, in line with the current literature, LambdaMART and Coordinate Ascent reveal to be highly effective in learning to rank lurkers, regardless of the assessment criterion (except for P@1000) and setting. RankNet shows good performance as well, except for P@1000 and for $nDCG$ with Grad setting. By contrast, AdaRank achieves quality results around 0.5 for the BB setting (with a peak of 0.875 for $nDCG$@1000), but much lower scores for UB (all criteria) and Grad ($nDCG$).

In general, LTLR methods tend to achieve the highest scores according to $nDCG$@1000 and the worst ones with P@1000. This would suggest that the heads of the ranking solutions are effectively detected in most cases, while the methods exhibit some difficulty in ranking most of the relevant users in the top-k positions, especially in the UB case.

Evaluation on heterogeneous network data

Figure 4.11 reports on LTLR performance results on all network data, which are divided into two representation settings: the one corresponding to relational features only (first three columns in the heatmap) and the other one corresponding to all features. At a first glance, we observe a variegate situation, whereby the lurking status of users can be successfully determined only for some of the methods and settings. Among the LTLR methods, the highest scores are again achieved by LambdaMART and Coordinate Ascent, which are equally able to perform nearly optimally with all settings of distribution of relevant/non-relevant instances, when only relational features are used. Also, these methods seem not to gain any particular benefit from the use of all the features, with few positive exceptions (e.g., Coordinate Ascent with $nDCG$) and negative exceptions (e.g., LambdaMART with MAP and P@1000). Competitive behavior is also shown by the representative pairwise approach, RankNet, which appears to benefit from the use of all networks and features, especially for the binary relevance cases (BB and UB) and with MAP and $nDCG$ criteria; for the graded relevance case, however, despite the improvement w.r.t. the previous evaluation scenario with followship networks and relational features, the

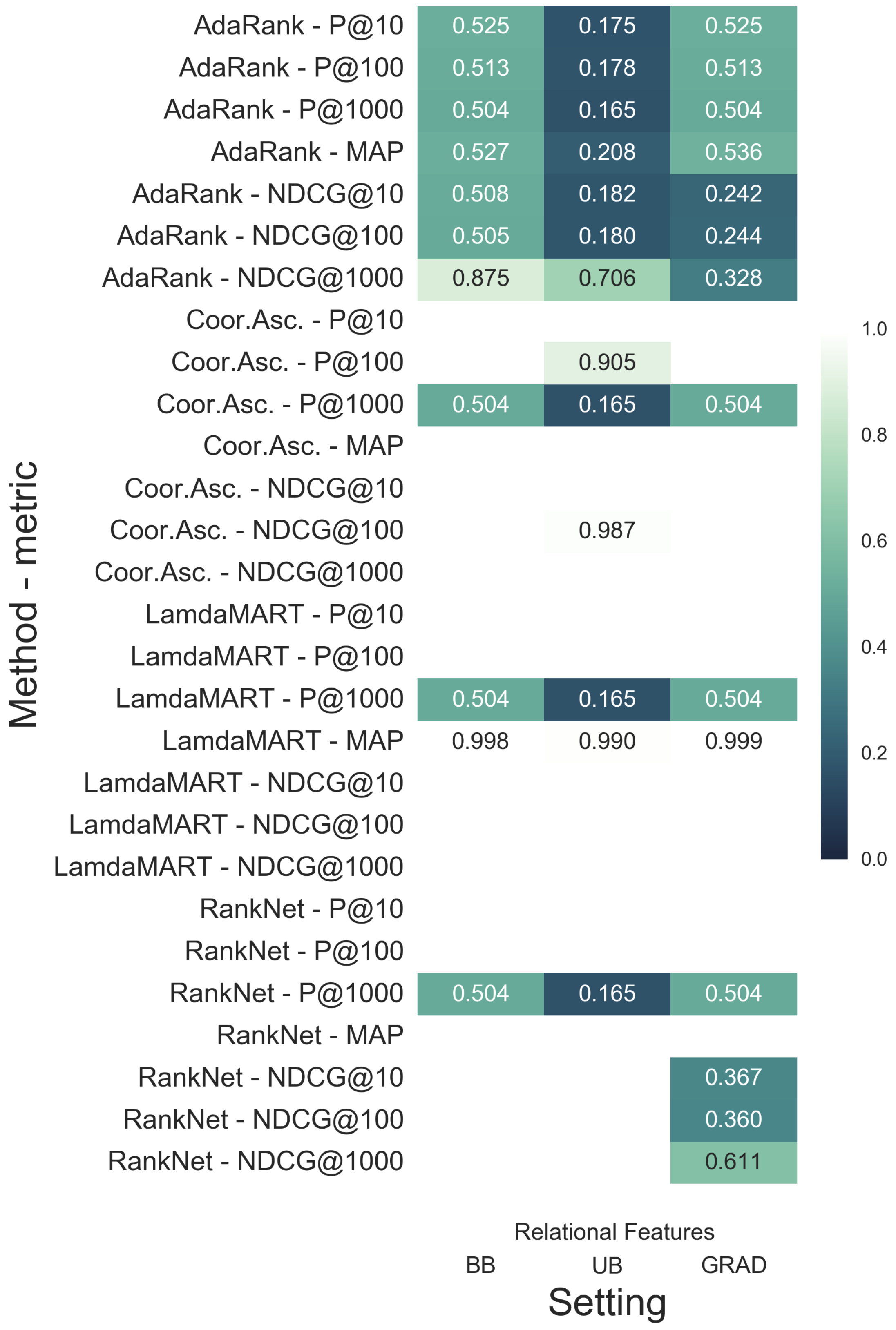
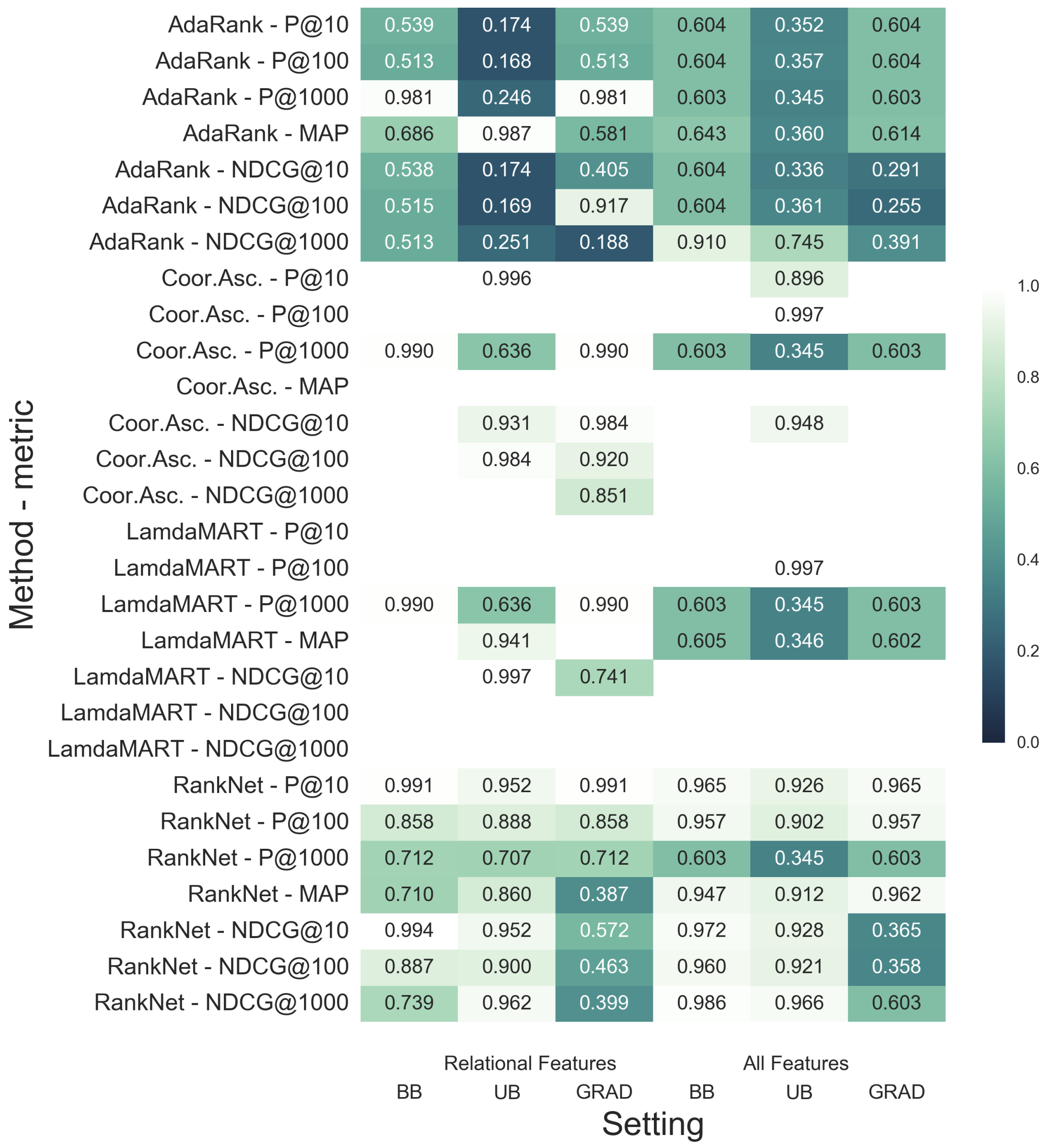


FIGURE 4.10: LTLR performance on followship network data with relational features only



Method - metric	Relational Features BB	Relational Features UB	Relational Features GRAD	All Features BB	All Features UB	All Features GRAD
AdaRank - P@10	0.539	0.174	0.539	0.604	0.352	0.604
AdaRank - P@100	0.513	0.168	0.513	0.604	0.357	0.604
AdaRank - P@1000	0.981	0.246	0.981	0.603	0.345	0.603
AdaRank - MAP	0.686	0.987	0.581	0.643	0.360	0.614
AdaRank - NDCG@10	0.538	0.174	0.405	0.604	0.336	0.291
AdaRank - NDCG@100	0.515	0.169	0.917	0.604	0.361	0.255
AdaRank - NDCG@1000	0.513	0.251	0.188	0.910	0.745	0.391
Coor.Asc. - P@10		0.996			0.896	
Coor.Asc. - P@100					0.997	
Coor.Asc. - P@1000	0.990	0.636	0.990	0.603	0.345	0.603
Coor.Asc. - MAP						
Coor.Asc. - NDCG@10		0.931	0.984		0.948	
Coor.Asc. - NDCG@100		0.984	0.920			
Coor.Asc. - NDCG@1000			0.851			
LamdaMART - P@10						
LamdaMART - P@100					0.997	
LamdaMART - P@1000	0.990	0.636	0.990	0.603	0.345	0.603
LamdaMART - MAP		0.941		0.605	0.346	0.602
LamdaMART - NDCG@10		0.997	0.741			
LamdaMART - NDCG@100						
LamdaMART - NDCG@1000						
RankNet - P@10	0.991	0.952	0.991	0.965	0.926	0.965
RankNet - P@100	0.858	0.888	0.858	0.957	0.902	0.957
RankNet - P@1000	0.712	0.707	0.712	0.603	0.345	0.603
RankNet - MAP	0.710	0.860	0.387	0.947	0.912	0.962
RankNet - NDCG@10	0.994	0.952	0.572	0.972	0.928	0.365
RankNet - NDCG@100	0.887	0.900	0.463	0.960	0.921	0.358
RankNet - NDCG@1000	0.739	0.962	0.399	0.986	0.966	0.603

FIGURE 4.11: LTLR performance on all network data, using either relational features only (leftmost 3-column group) or all features (rightmost 3-column group)

RankNet scores remain far lower than LambdaMART and Coordinate Ascent. The other listwise method, AdaRank, again is found to perform quite poorly in most cases (below 0.7), with a minimum score of 0.168 (UB, P@100). We tend to ascribe this to an inability of the method of selecting good enough subspaces of features for the building of weak learners and their subsequent combination.

Overall, comparing the performances of LTLR methods on heterogeneous data against those obtained on followship networks and relational features only, we conclude that training and testing from heterogeneous data, by exploiting various types of features, does not undermine the significance of the LTLR framework. Moreover, since the general performance trends by the methods are similar to the restricted case of followship networks with relational features, the basic topology-driven principles underlying LurkerRank (and hence the definition of the relevance labels in the training data) are applicable in different platform contexts.

Relevance of non-relational features

Figures 4.12–4.17 show the prediction capabilities obtained with the use of media-based, activity-rate, and platform-specific features, with and without the addition of relational features, on specific subsets of network data (D1–D6, cf. Section 4.4.2). As a general remark, LambdaMART and, to some extent, Coordinate Ascent do not gain any benefit from the extension of the feature space, while AdaRank and RankNet show significant improvement in most cases. More specifically, concerning subsets D1–D3 (Figs. 4.12–4.14), all the LTLR methods show none or negligible variation in performances when including or not the relational features, in most cases. This would suggest that, for relatively homogeneous sets of network data, non-relational features can induce most or all of the lurker-ranking prediction ability exhibited by LTLR, i.e., topological information could become redundant. On the other hand, it also indicates that the topology-driven formulation of LurkerRank is able to capture user's behavioral patterns that are consistent with the production and consumption of contents, and relating activity-rate.

On D4–D6 (Figs. 4.15–4.17), we observe a more diversified situation. While LambdaMART and Coordinate Ascent are still able to extract all the needed information from few non-relational features and induce a well-performing ranking model (with the usual exception regarding P@1000), AdaRank and RankNet offer poor performance.

Figures 4.18-a)–4.18-d) provide further insights into the behavior of AdaRank and RankNet, showing the percentage increase in performance when using all features compared to a representation of network data based on non-relational features only.

As anticipated before, most of the percentage values are actually nearly zero or negative. For instance, in Fig. 4.18-a), RankNet shows a relatively small decrease in performance for all settings, while AdaRank takes little advantage in using all features for the UB setting;

in Fig. 4.18-c), with the exception of RankNet $nDCG$@10 in the Grad setting, whereby using all the features is extremely advantageous (200% of increase), the variations turn out to be positive but negligible for AdaRank, while

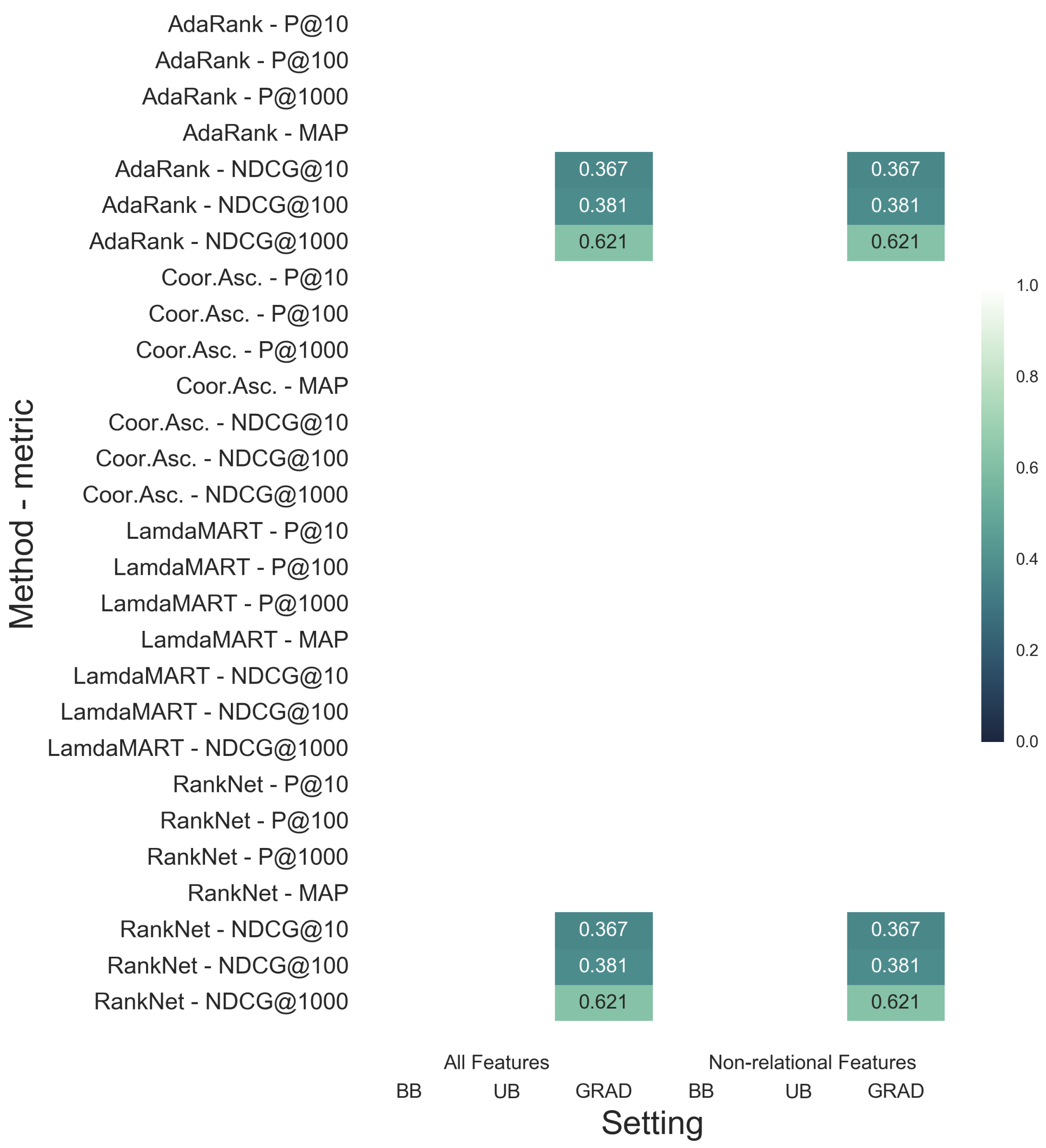
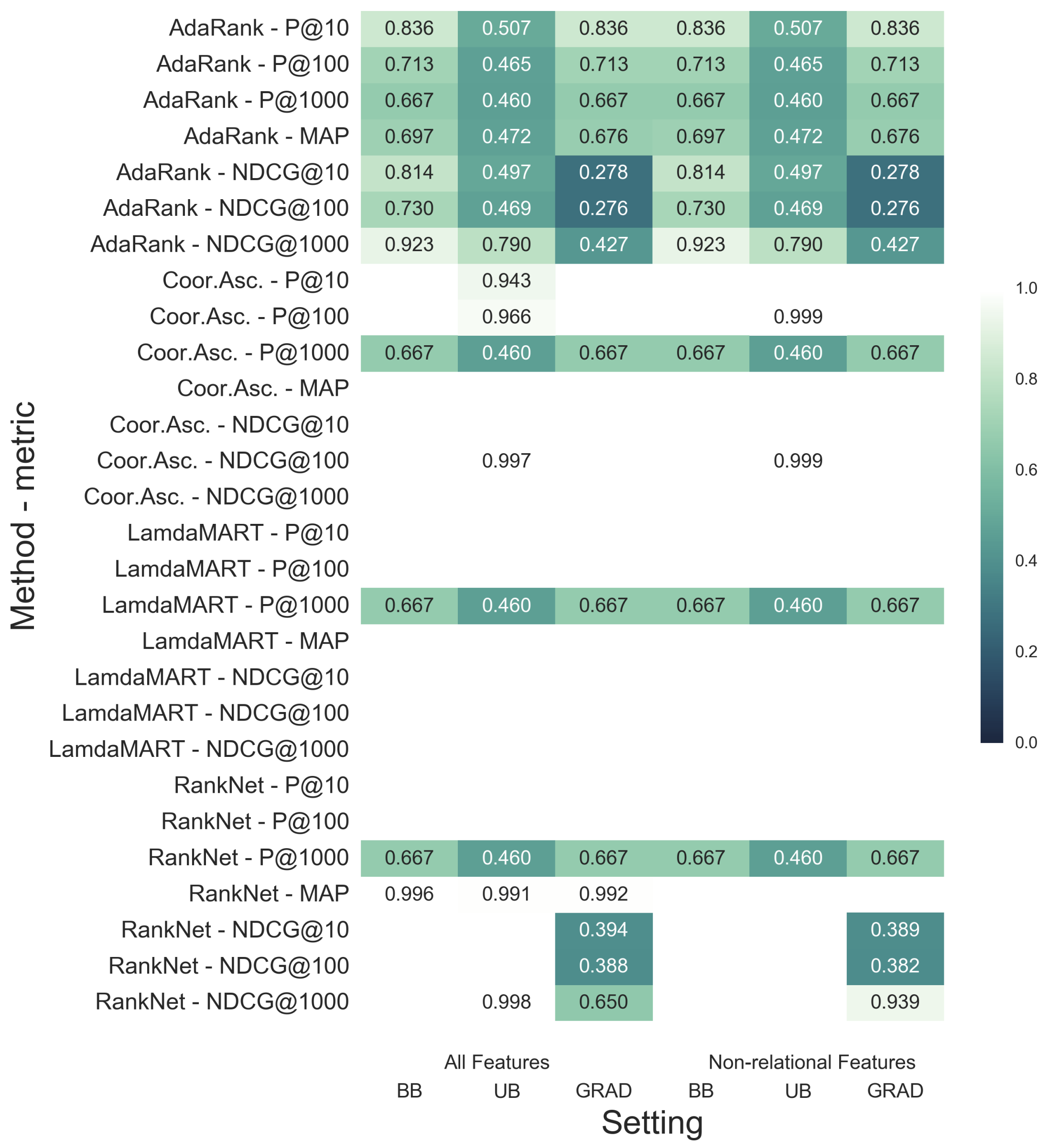
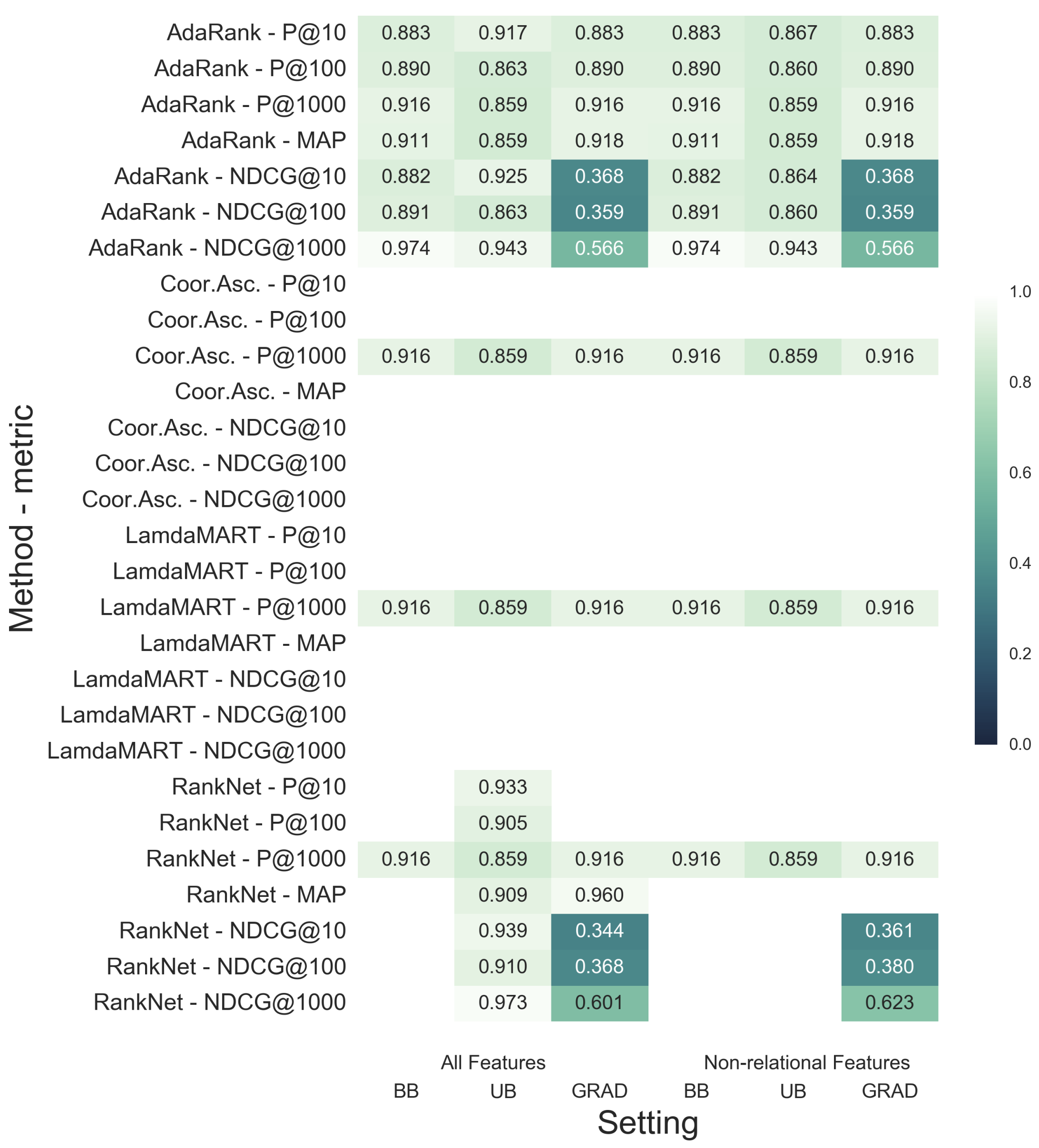


FIGURE 4.12: LTLR performance on subset D1 (Flickr networks)



	All Features			Non-relational Features		
Method - metric	BB	UB	GRAD	BB	UB	GRAD
AdaRank - P@10	0.836	0.507	0.836	0.836	0.507	0.836
AdaRank - P@100	0.713	0.465	0.713	0.713	0.465	0.713
AdaRank - P@1000	0.667	0.460	0.667	0.667	0.460	0.667
AdaRank - MAP	0.697	0.472	0.676	0.697	0.472	0.676
AdaRank - NDCG@10	0.814	0.497	0.278	0.814	0.497	0.278
AdaRank - NDCG@100	0.730	0.469	0.276	0.730	0.469	0.276
AdaRank - NDCG@1000	0.923	0.790	0.427	0.923	0.790	0.427
Coor.Asc. - P@10		0.943				
Coor.Asc. - P@100		0.966			0.999	
Coor.Asc. - P@1000	0.667	0.460	0.667	0.667	0.460	0.667
Coor.Asc. - MAP						
Coor.Asc. - NDCG@10						
Coor.Asc. - NDCG@100		0.997			0.999	
Coor.Asc. - NDCG@1000						
LamdaMART - P@10						
LamdaMART - P@100						
LamdaMART - P@1000	0.667	0.460	0.667	0.667	0.460	0.667
LamdaMART - MAP						
LamdaMART - NDCG@10						
LamdaMART - NDCG@100						
LamdaMART - NDCG@1000						
RankNet - P@10						
RankNet - P@100						
RankNet - P@1000	0.667	0.460	0.667	0.667	0.460	0.667
RankNet - MAP	0.996	0.991	0.992			
RankNet - NDCG@10			0.394			0.389
RankNet - NDCG@100			0.388			0.382
RankNet - NDCG@1000		0.998	0.650			0.939

FIGURE 4.13: LTLR performance on subset D2 (Flickr and Instagram networks)



Method - metric	All Features BB	All Features UB	All Features GRAD	Non-relational Features BB	Non-relational Features UB	Non-relational Features GRAD
AdaRank - P@10	0.883	0.917	0.883	0.883	0.867	0.883
AdaRank - P@100	0.890	0.863	0.890	0.890	0.860	0.890
AdaRank - P@1000	0.916	0.859	0.916	0.916	0.859	0.916
AdaRank - MAP	0.911	0.859	0.918	0.911	0.859	0.918
AdaRank - NDCG@10	0.882	0.925	0.368	0.882	0.864	0.368
AdaRank - NDCG@100	0.891	0.863	0.359	0.891	0.860	0.359
AdaRank - NDCG@1000	0.974	0.943	0.566	0.974	0.943	0.566
Coor.Asc. - P@10						
Coor.Asc. - P@100						
Coor.Asc. - P@1000	0.916	0.859	0.916	0.916	0.859	0.916
Coor.Asc. - MAP						
Coor.Asc. - NDCG@10						
Coor.Asc. - NDCG@100						
Coor.Asc. - NDCG@1000						
LamdaMART - P@10						
LamdaMART - P@100						
LamdaMART - P@1000	0.916	0.859	0.916	0.916	0.859	0.916
LamdaMART - MAP						
LamdaMART - NDCG@10						
LamdaMART - NDCG@100						
LamdaMART - NDCG@1000						
RankNet - P@10		0.933				
RankNet - P@100		0.905				
RankNet - P@1000	0.916	0.859	0.916	0.916	0.859	0.916
RankNet - MAP		0.909	0.960			
RankNet - NDCG@10		0.939	0.344			0.361
RankNet - NDCG@100		0.910	0.368			0.380
RankNet - NDCG@1000		0.973	0.601			0.623

FIGURE 4.14: LTLR performance on subset D3 (Flickr and FriendFeed networks)

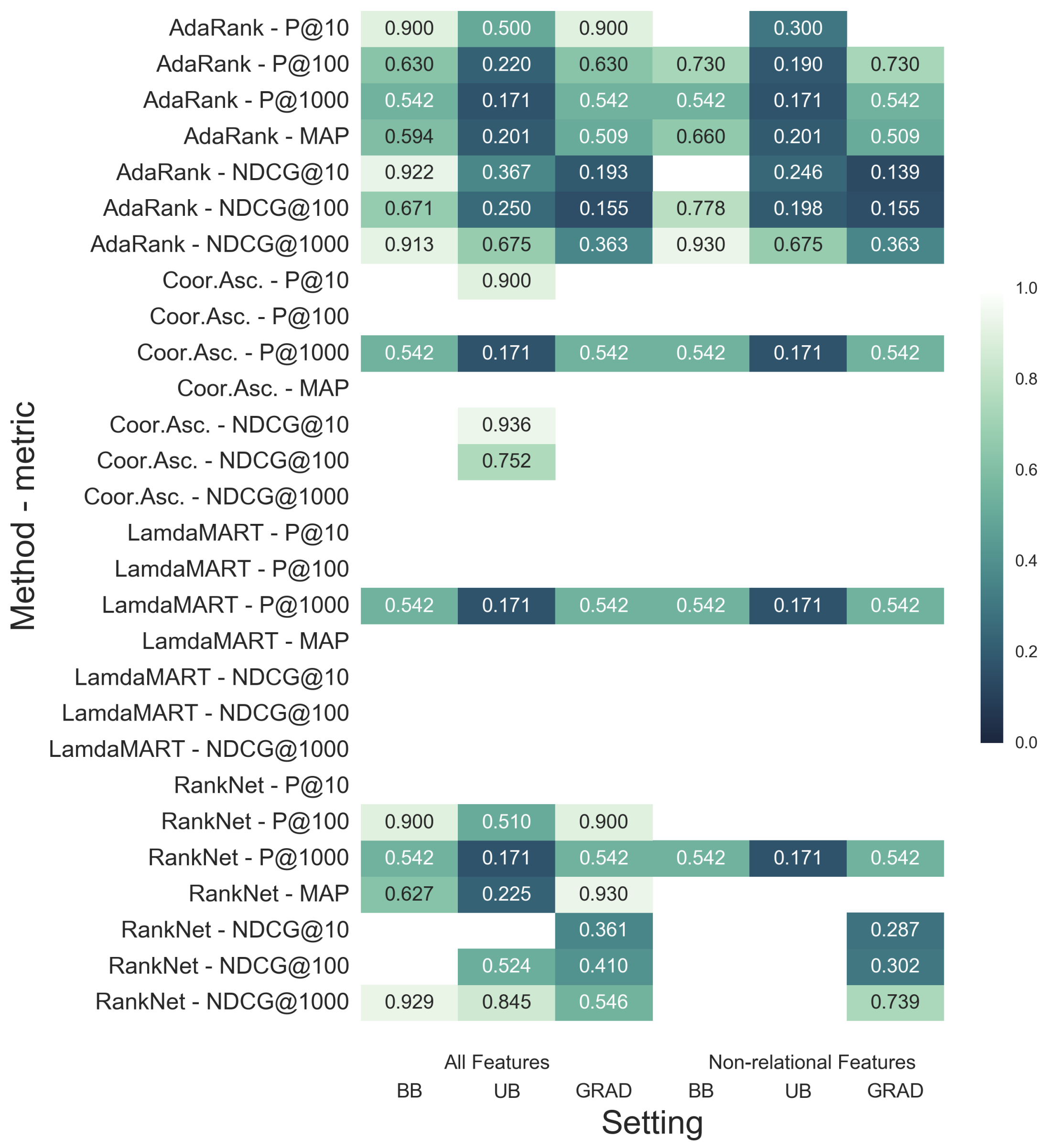


FIGURE 4.15: LTLR performance on subset D4 (GitHub network)

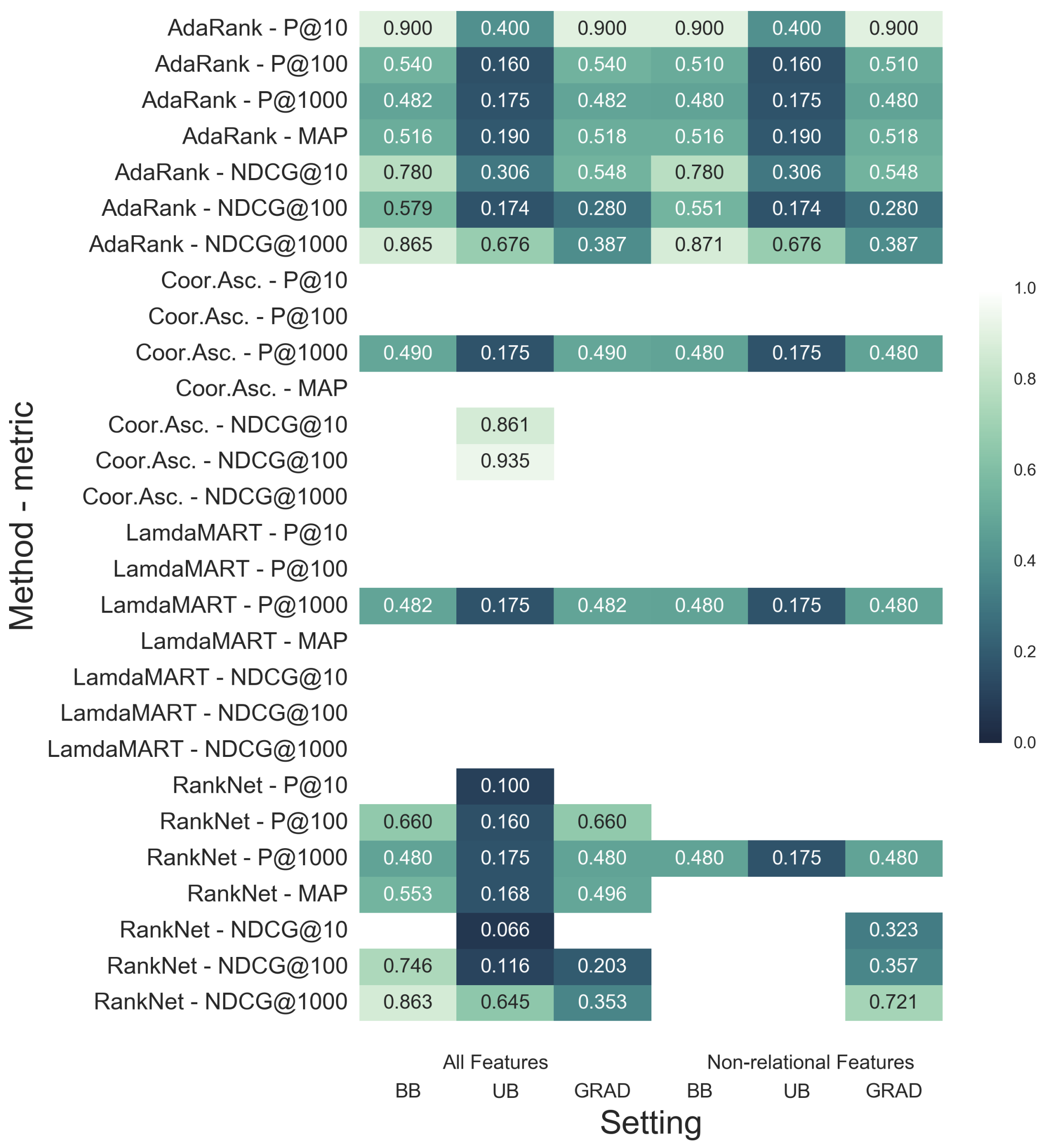
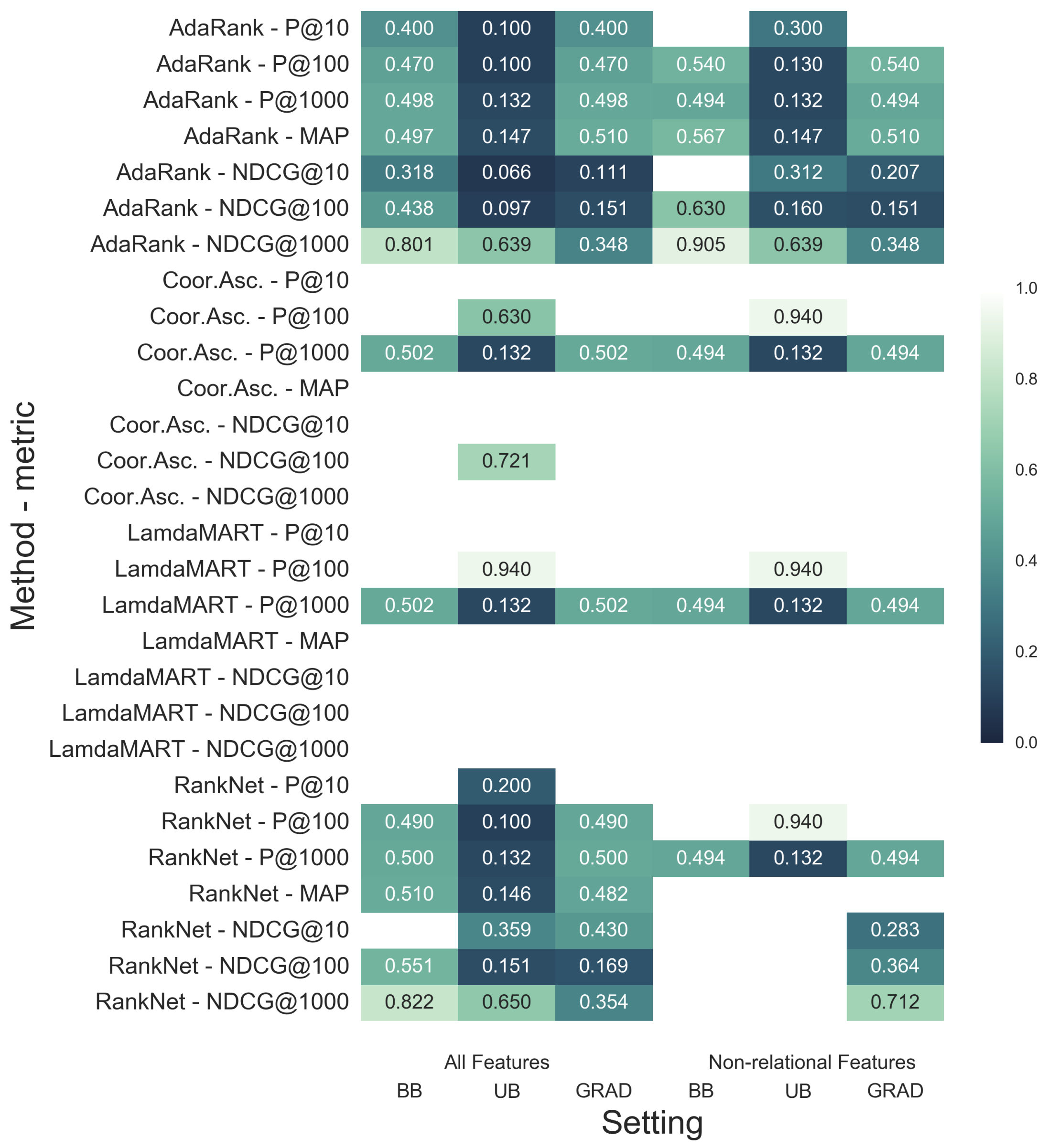


FIGURE 4.16: LTLR performance on subset D5 (StackOverflow network)



Method - metric	All Features BB	All Features UB	All Features GRAD	Non-relational Features BB	Non-relational Features UB	Non-relational Features GRAD
AdaRank - P@10	0.400	0.100	0.400		0.300	
AdaRank - P@100	0.470	0.100	0.470	0.540	0.130	0.540
AdaRank - P@1000	0.498	0.132	0.498	0.494	0.132	0.494
AdaRank - MAP	0.497	0.147	0.510	0.567	0.147	0.510
AdaRank - NDCG@10	0.318	0.066	0.111		0.312	0.207
AdaRank - NDCG@100	0.438	0.097	0.151	0.630	0.160	0.151
AdaRank - NDCG@1000	0.801	0.639	0.348	0.905	0.639	0.348
Coor.Asc. - P@10						
Coor.Asc. - P@100		0.630			0.940	
Coor.Asc. - P@1000	0.502	0.132	0.502	0.494	0.132	0.494
Coor.Asc. - MAP						
Coor.Asc. - NDCG@10						
Coor.Asc. - NDCG@100		0.721				
Coor.Asc. - NDCG@1000						
LamdaMART - P@10						
LamdaMART - P@100		0.940			0.940	
LamdaMART - P@1000	0.502	0.132	0.502	0.494	0.132	0.494
LamdaMART - MAP						
LamdaMART - NDCG@10						
LamdaMART - NDCG@100						
LamdaMART - NDCG@1000						
RankNet - P@10		0.200				
RankNet - P@100	0.490	0.100	0.490		0.940	
RankNet - P@1000	0.500	0.132	0.500	0.494	0.132	0.494
RankNet - MAP	0.510	0.146	0.482			
RankNet - NDCG@10		0.359	0.430			0.283
RankNet - NDCG@100	0.551	0.151	0.169			0.364
RankNet - NDCG@1000	0.822	0.650	0.354			0.712

FIGURE 4.17: LTLR performance on subset D6 (Twitter networks)

being significant and disadvantageous for RankNet for all the settings. Similar, but extended to AdaRank as well, is the situation corresponding to Fig. 4.18-d) relating to Twitter data. Overall, in the majority of the cases, the BB setting seems to be the least affected by the extension of the feature set, while most and higher variations are observed for UB and Grad. Note however that the above outcomes should be taken with a grain of salt, since the two weakest LTR methods among those selected particularly suffer from a limited amount of training data, which affects their inability to generalize from the available examples.

Evaluation on distinct networks

Figures 4.21-4.20 report on the performance of LTLR methods on test data, after training on network data belonging to different platforms/contexts. One general remark that stands out is the ability of LambdaMART and Coordinate Ascent to effectively learn a ranking function from a specific network, even from relatively few instances, and use it to successfully rank lurkers that belong to another network. As expected, the other two methods may have mid-to-low performance especially when there is unbalancing towards the size of test data (i.e., Fig. 4.21 and Fig. 4.20), in which case the use of all features could be beneficial. More interestingly, in the opposite case (i.e., when Flickr network data is used as training), AdaRank and RankNet yield as optimal performance as LambdaMART and Coordinate Ascent, with the exception of Grad setting and $nDCG$ criterion (which might be explained since the grain of relevance labels corresponding to the training instances could be coarse in some cases); note also that the performance of each method does not vary depending on whether all features or only non-relational-features are used.

4.4.4 Discussion

Results have shown that LTLR can be effective in learning to rank lurkers. More precisely, some LTLR methods are able to predict the ranking of lurkers once trained according to the (graded or not) relevance labels originally provided by LurkerRank. While best-performing LambdaMART and Coordinate Ascent can optimally predict the correct rank on test data even with few media-based, activity-rate or platform-specific features, a general trend is that a combination between relational and additional features should be used to improve accuracy and ranking performance, especially for worst-performing methods like AdaRank and RankNet.

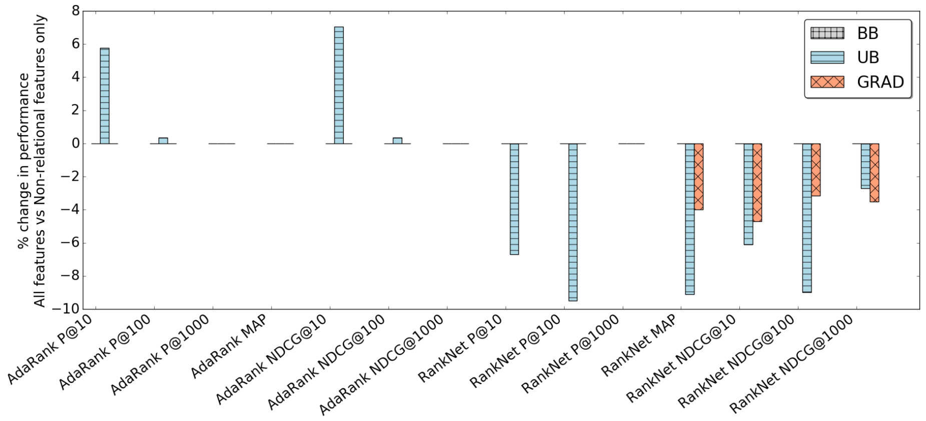
Our preliminary evaluation on the feature informativeness (cf. Section 4.4.2) indicated that relational features are the most important to estimate the prediction capability of the LTLR framework: while this is not actually surprising, since the training annotations in our network data were originally defined upon topological information, it is partially contradicted by our comprehensive evaluation of the performance of the learning-to-rank methods used in our framework. In fact, the most effective methods can equally perform by using non-relational features only or all features, whereas less effective LTR methods might take advantage from the use of all features compared to either relational or non-relational features only.

As a final remark, it should be noted that, though LambdaMART offers the best overall performance in our LTLR framework, it also turns out to need the highest memory requirements as well as to be the slowest method along with RankNet, being several order of magnitude slower than AdaRank and Coordinate Ascent. While the former, despite its efficiency, has not shown to generate a robust ranking model in most cases, Coordinate Ascent offers the best trade-off between accuracy and execution time.

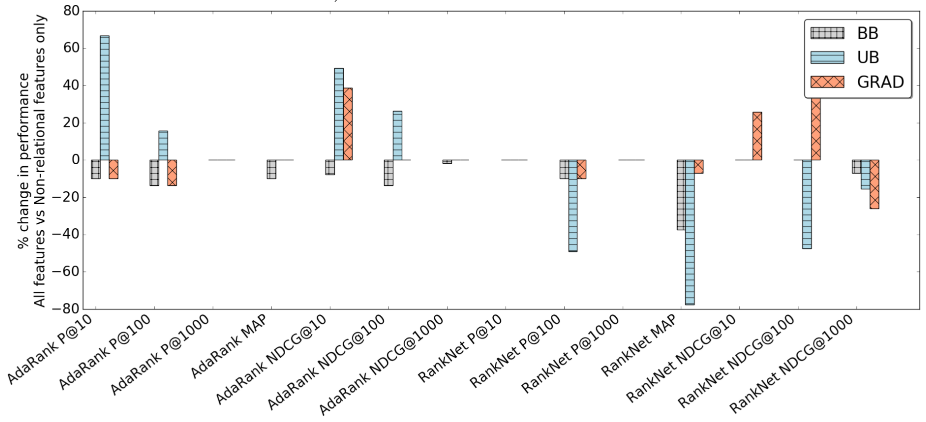
4.4.5 Conclusion

In this work, we advanced research on the topic of lurking behavior analysis, by pursuing the goal of developing a robust, multi-platform, supervised lurker-ranking model upon the unsupervised, eigenvector-centrality-based models existing in the literature. To this end, we proposed a learning-to-rank framework, named LTLR, whose key aspect is the capability of exploiting different types of information available from multiple OSNs in order to build training data tuples over different subspaces of features. These features include media-based, activity-rate and platform specific characteristics, besides topological information underlying lurking principles in OSNs. The proposed framework is versatile w.r.t. the learning-to-rank method. Experiments conducted on 23 network datasets and involving four state-of-the-art learning-to-rank methods have shown the meaningfulness of LTLR, which often corresponded to highly effective ranking.

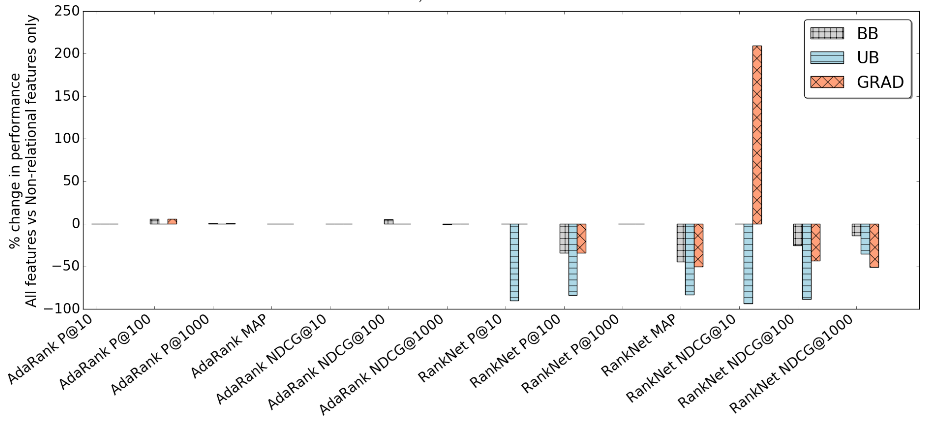
It should be noted that our proposed framework is also versatile w.r.t. the methodology for labeling the training data: while LurkerRank methods were used in this work to compensate for the lack of human-generated relevance labels, we nonetheless advocate for the use of ground-truth scores to assign with observed lurking behaviors. Moreover, it would be interesting to enhance LTLR to learn from cross-platform features, in order to improve our understanding of behavioral profiles of users who may act as lurkers in one network and, conversely, as contributors in one or more other networks [190].



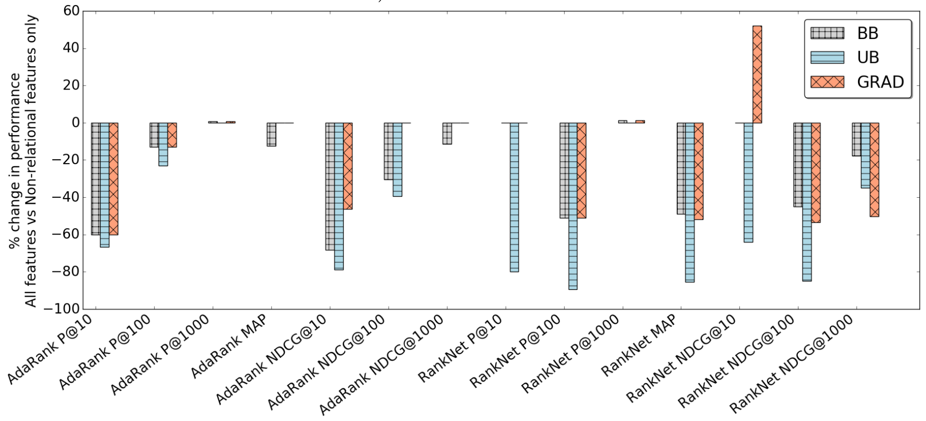
a) Flickr and FriendFeed



b) GitHub



c) StackOverflow



d) Twitter

FIGURE 4.18: AdaRank and RankNet performance increase due to the use of all features compared to the use of non-relational features only

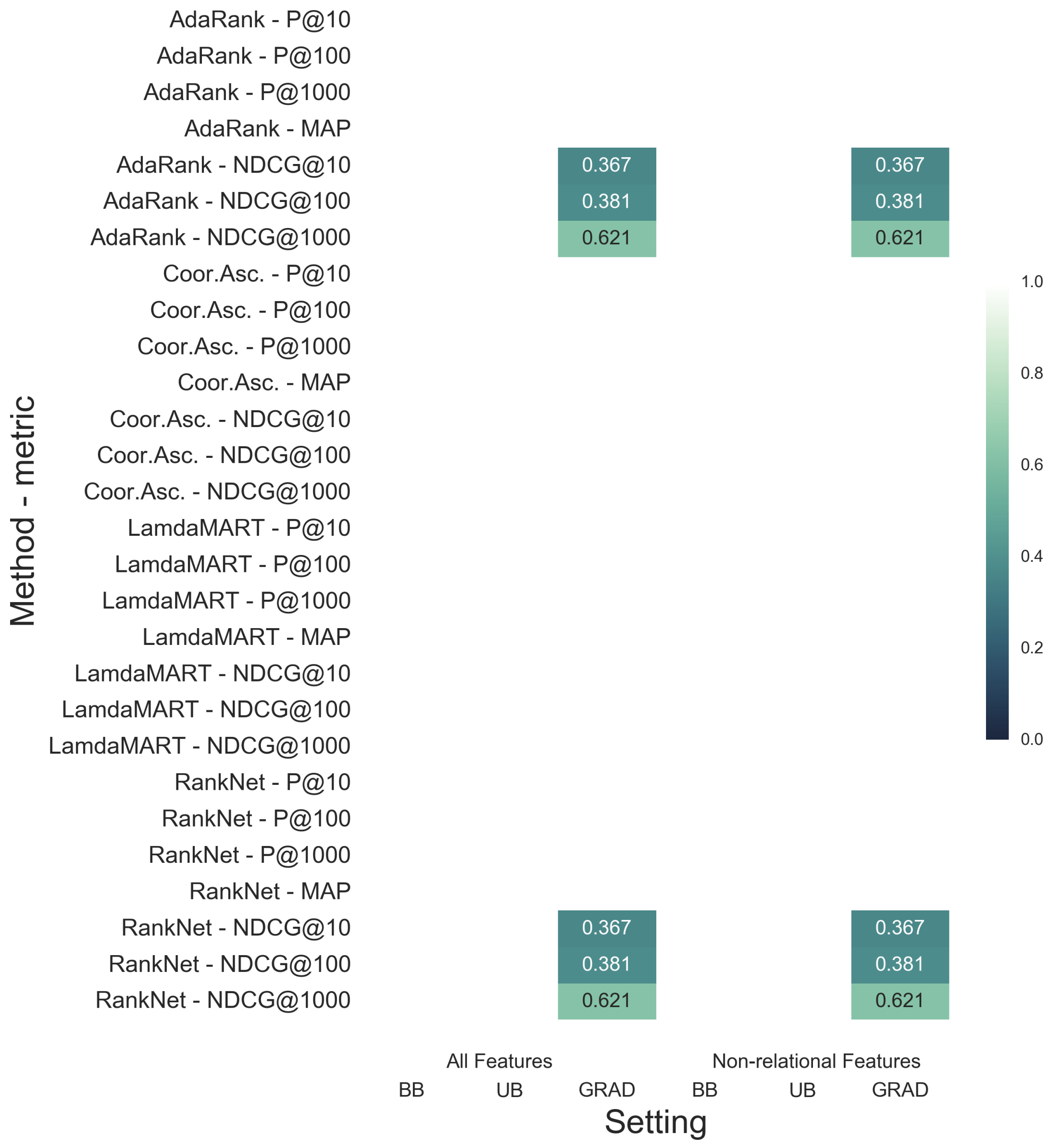


FIGURE 4.19: LTLR performance on subset D2, with training on Flickr and testing on Instagram

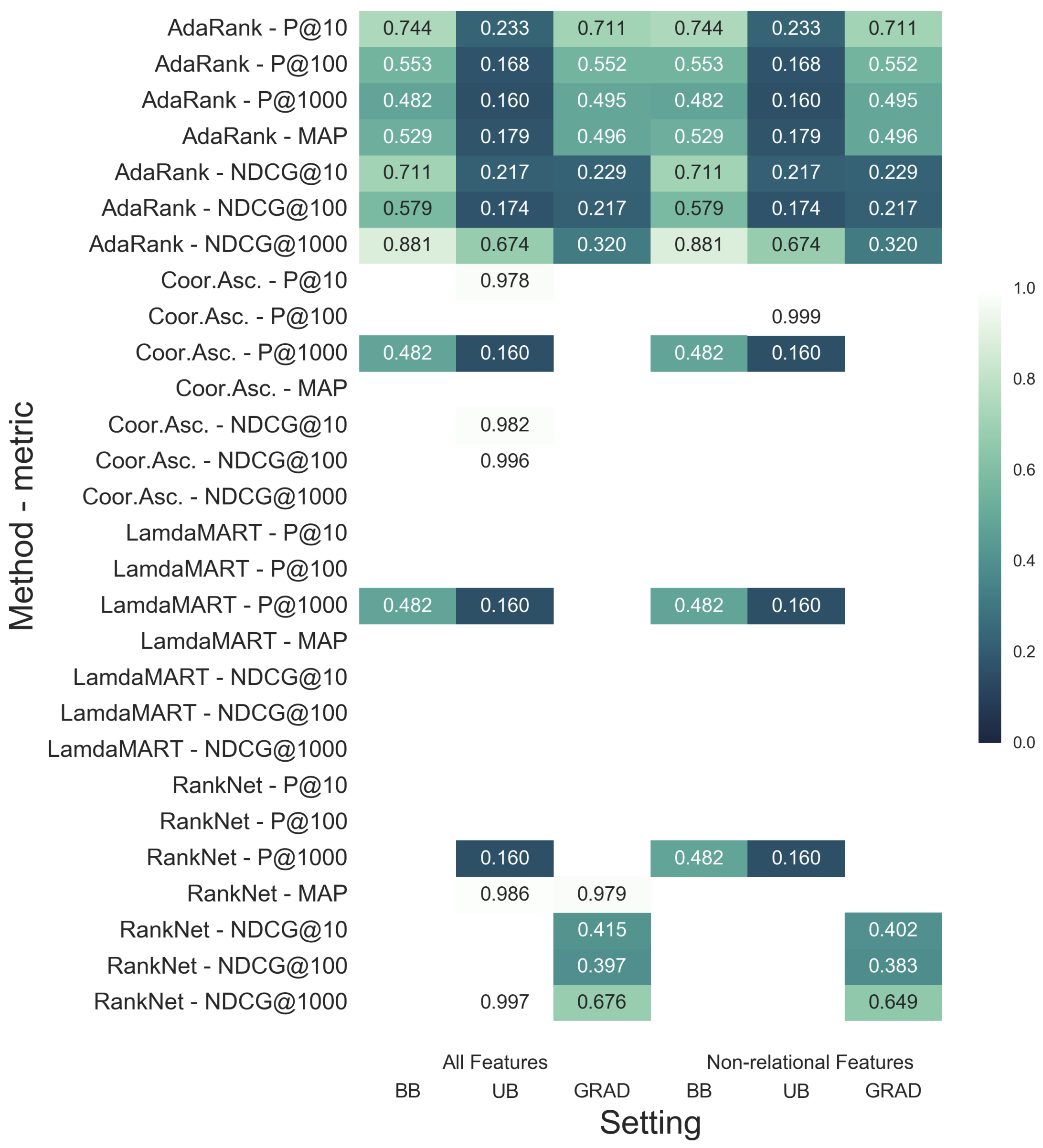
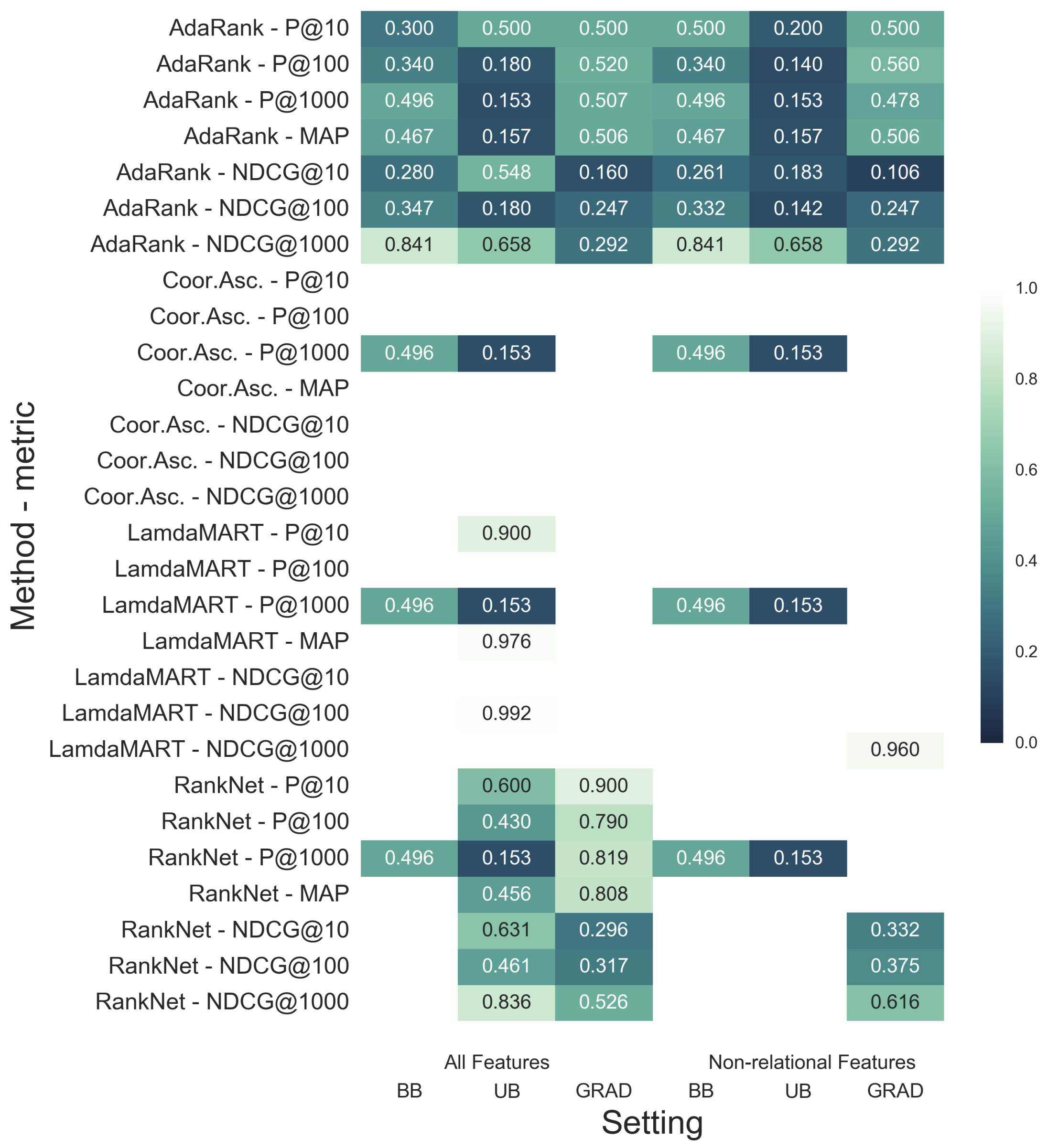


FIGURE 4.20: LTLR performance on subset D2, with training on Instagram and testing on Flickr



Method - metric	All Features BB	All Features UB	All Features GRAD	Non-relational Features BB	Non-relational Features UB	Non-relational Features GRAD
AdaRank - P@10	0.300	0.500	0.500	0.500	0.200	0.500
AdaRank - P@100	0.340	0.180	0.520	0.340	0.140	0.560
AdaRank - P@1000	0.496	0.153	0.507	0.496	0.153	0.478
AdaRank - MAP	0.467	0.157	0.506	0.467	0.157	0.506
AdaRank - NDCG@10	0.280	0.548	0.160	0.261	0.183	0.106
AdaRank - NDCG@100	0.347	0.180	0.247	0.332	0.142	0.247
AdaRank - NDCG@1000	0.841	0.658	0.292	0.841	0.658	0.292
Coor.Asc. - P@10						
Coor.Asc. - P@100						
Coor.Asc. - P@1000	0.496	0.153		0.496	0.153	
Coor.Asc. - MAP						
Coor.Asc. - NDCG@10						
Coor.Asc. - NDCG@100						
Coor.Asc. - NDCG@1000						
LamdaMART - P@10		0.900				
LamdaMART - P@100						
LamdaMART - P@1000	0.496	0.153		0.496	0.153	
LamdaMART - MAP		0.976				
LamdaMART - NDCG@10						
LamdaMART - NDCG@100		0.992				
LamdaMART - NDCG@1000						0.960
RankNet - P@10		0.600	0.900			
RankNet - P@100		0.430	0.790			
RankNet - P@1000	0.496	0.153	0.819	0.496	0.153	
RankNet - MAP		0.456	0.808			
RankNet - NDCG@10		0.631	0.296			0.332
RankNet - NDCG@100		0.461	0.317			0.375
RankNet - NDCG@1000		0.836	0.526			0.616

FIGURE 4.21: LTLR performance on subset D3, with training on FriendFeed and testing on Flickr

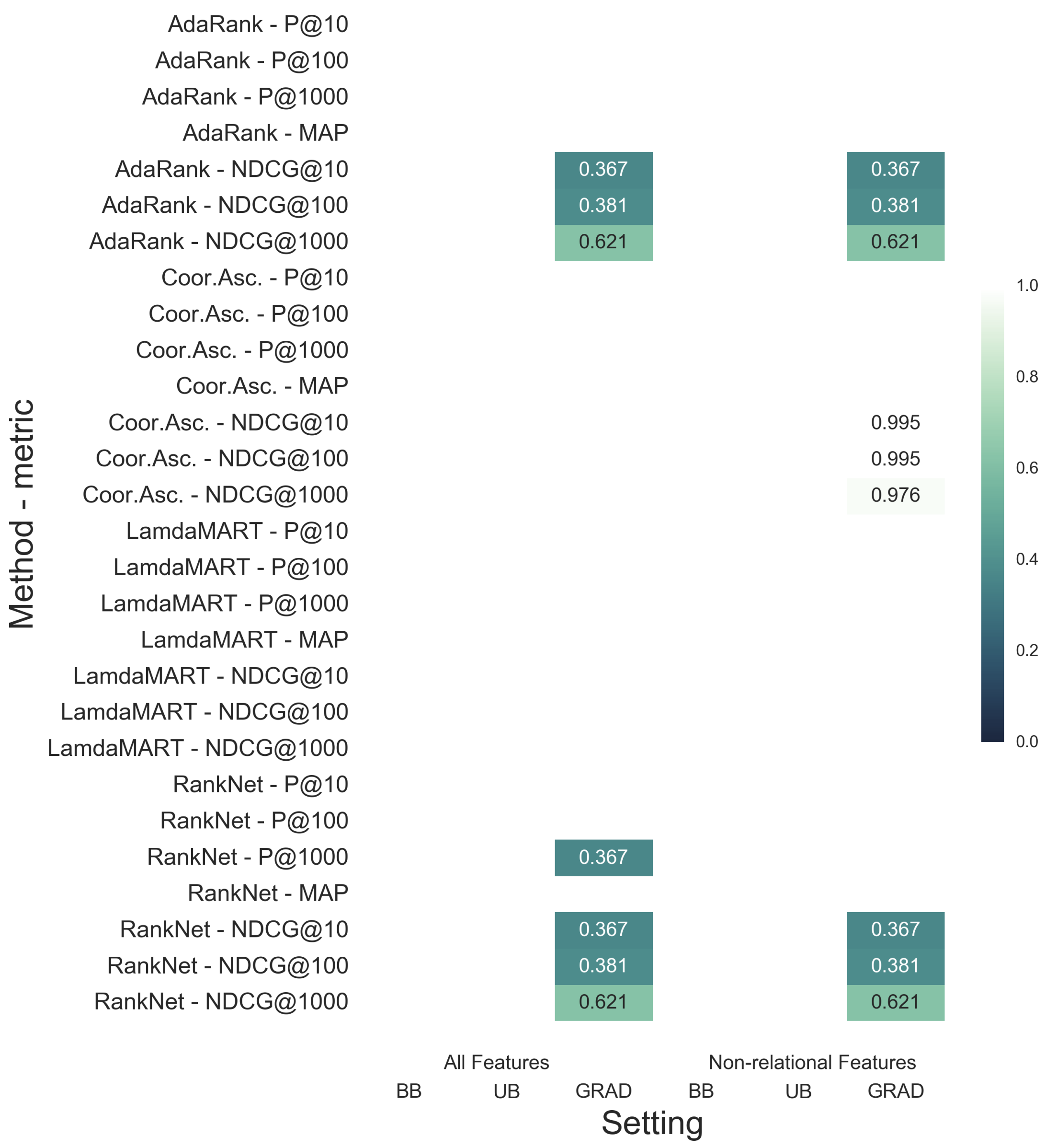


FIGURE 4.22: LTLR performance on subset D3, with training on Flickr and testing on FriendFeed

Chapter 5

Enhancing the modeling of complex social networks via embedding

5.1 Introduction

Graph embedding techniques aim to transform a graph in a low-dimensional representation, enabling the use of a rich set of tools mostly based on state-of-the-art machine-learning methods [192]. A clear advantage in obtaining a low-dimensional representation is to reduce the memory-footprint requirements while retaining relevant information for the task at hand [193]. Example tasks include node classification, node clustering, node recommendation, retrieval and ranking, link (i.e., edge) prediction, and graph classification [194]. Moreover, since the embedding corresponds to a vectorial representation, vector operations on the learned model might in principle be computationally more convenient than graph operations.

The general goal of a graph embedding process is to learn a function f that maps one or many features of the network (i.e., nodes, edges, or the whole graph) to a new d-dimensional space $\mathbb{R}^d$. For instance, in the case of node embedding for a multilayer network, the function to learn can be of the form $f : \mathcal{A} \rightarrow \mathbb{R}^d$ or $f : V \rightarrow \mathbb{R}^d$ where $\mathcal{A}$ and $\mathcal{V}$ denote respectively the set of actors and the set of nodes. In effect, graph embedding approaches can be classified based on the constituent(s) of a graph they are designed for. Several works have been recently developed for nodes [195], [196], edges [197], subgraphs (e.g., communities [198], [199]), and whole-graph embedding[200]–[202].

Matrix factorization based algorithms [203]–[205] are the pioneering methods in graph embedding, since they apply a decomposition technique to a matrix representation of an input graph. There are mainly two types of matrix-factorization-based graph embedding: factorization of graph Laplacian eigenmaps, and direct factorization of the node proximity matrix [192]. One of the earliest methods designed for multilayer networks, specifically for community detection purposes, is Principled Modularity Maximization (PMM) [206]. PMM infers a latent community structure for the nodes in a multilayer network by performing a two-step methodology. Firstly, PMM extracts low dimensional vectors from each layer through modularity maximization and aggregates the extracted information through cross-dimension integration. Finally, a simple k-means is carried out on the learned representation to find out the communities of the network.

From a different perspective, in the LINE method [207], two functions are defined for both first- and second-order proximities, where first-order proximity refers to edge weights and second-order proximity refers to neighborhood similarity. LINE defines two joint probability distributions for each pair of nodes and minimizes the Kullback–Leibler divergence of these two distributions.

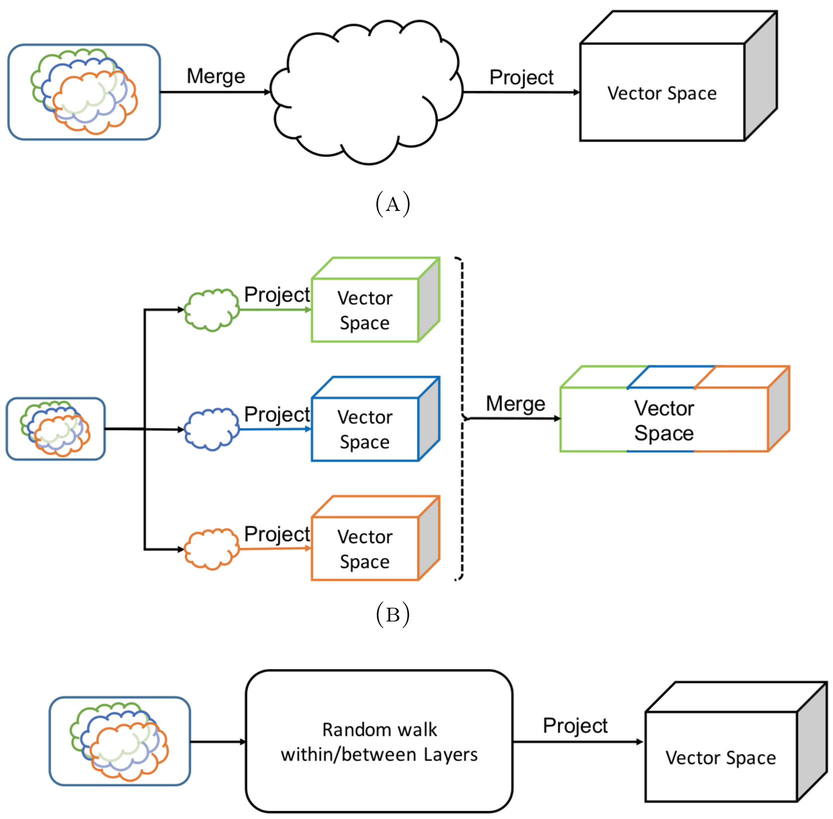
More recently, there has been momentum for the development of *deep-learning-oriented* techniques. DeepWalk [195] and Node2Vec [196] are two exemplary random-walk-based methods. According to the *Skip-gram* [208] model, these methods treat nodes as words and paths as sentences, then apply deep learning to the sampled random-walk paths. As the skip-gram model aims to maximize the co-occurrence probability among the words that appear within a window in the same sentence, the resulting graph embedding by DeepWalk and Node2Vec preserves first- and second-order proximity of nodes.

Deep-learning-oriented methods include the use of autoencoders and their variants (i.e., denoising, variational, etc.), which aim to maximize the reconstruction accuracy of the input graph, by applying a *decoder* block to the latent representation learned by an *encoder* [209]–[211]. Another deep-learning common approach is to directly apply a convolutional neural network (CNN) to Euclidean data generated from a graph [212], or to adapt CNN to graphs [213], [214]. In addition to this group of methods, GraphSAGE [215] aims to learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. GraphSAGE, which can be seen as an extension of the GCN [213] framework to the inductive setting, can deal with evolving graphs and can easily generate embedding vectors for previously unseen nodes.

5.1.1 Graph embedding for multilayer networks

At the time of writing of this thesis, there is a relatively small corpus of deep-learning-oriented methods of graph embedding specifically conceived for multilayer networks, namely: PMNE [216], MNE [217], and MELL [218].

The first two methods adapt and extend the Word2Vec [219], [220] model to multilayer and multiplex networks, respectively. The input sentences (i.e., paths) are generated by a second-order random walk process, which is constrained to explore one layer at the time, with the exception of PMNE Layer co-analysis method in which the random walker gains the ability to jump from a layer to another. As depicted in Fig.5.1, PMNE includes three different approaches: two naive baselines, and one natively multilayer. In the Network Aggregation approach, the multilayer network is merged into a single weighted network (where multiple edges between two nodes are not allowed) and the embeddings for actors are computed on the aggregated network, i.e., $f : \mathcal{A} \rightarrow \mathbb{R}^d$. In the Results Aggregation approach, the embeddings are computed separately on each layer and then successively concatenated together, i.e., $f_l : V \rightarrow \mathbb{R}^{d_l}$ and $f = f_1||f_2||\cdots||f_{|L|}$ with $f : V \rightarrow \mathbb{R}^{d'|L|}$. Unlike the previous two approaches, in the Layer Co-analysis approach the second-order random walker



(C)

FIGURE 5.1: Architecture of PMNE: (a) Network Aggregation, (b) Results Aggregation, (c) Layer Co-analysis [216].

acquires the ability to jump across layers, allowing the generation of walks that are not limited to a single layer of the network, i.e., $f : V \to R^d$. In all of the three approaches, the Node2Vec model is chosen to be applied to the generated paths.

MNE generates a layer-wise embedding $\mathbf{v}_n^i$, for every node n and layer i, which consists of an embedding $\mathbf{b}_n$ shared across all the layers and that describes the node n globally, and a local (i.e., layer-wise) embedding $\mathbf{u}_n^i$:

$$\mathbf{v}_n^i = \mathbf{b}_n + w^i \cdot \mathbf{X}^{i^T} \mathbf{u}_n^i.$$

In the above equation, w^i denotes the importance of layer i, and $\mathbf{X}^{i^T}$ is a transformation matrix that aligns the global and local embedding vectors.

MELL is based on a regression framework, and unlike the previously described methods, it also takes into account the directionality of the edges (i.e., $(u, v) \neq (v, u)$) by using two vectors for each node $\mathbf{v}_H$ and $\mathbf{v}_T$. Node embeddings belonging to the same node are enforced to be close to each other through a regularization term. In addition to node embedding, MELL also learns a set of *layer vectors* representing layers' connectivity, in order to differentiate edge probabilities in each layer. The probability between two nodes v_i^l and v_j^l, both belonging to the layer l, is equal to:

$$p(\mathbf{v}_i^l, \mathbf{v}_j^l) = \frac{1}{1 + \exp\left(-(\mathbf{v}_i^l + \mathbf{r}_l)^T \cdot \mathbf{v}_j^l\right)}$$

where $\mathbf{v}_i^l$ (resp. $\mathbf{v}_j^l$) denotes the embedding vector for v_i^l (resp. v_j^l), and $\mathbf{r}_l$ is the vector embedding for the l-th layer.

Both MELL and MNE are well-suited to link prediction and node classification, where high order proximity information, extracted through random walk, reveal to be particularly expressive. MELL should be preferred when the information carried by edge directionality is valuable for the task at hand. Conversely, MNE should be preferred when having both *global* and *local* embedding for each node is important.

5.2 From Word2Vec to Node2Vec and beyond

The Word2Vec model [221], [222] and its extensions have recently raised a great deal of attention in the machine learning community. In [221], [222] Mikolov et al. presented two complementary models: i) Continuous Bag of Words (CBOW) with the goal of predicting a word given a context, and ii) the Skip-gram model that conversely tries to predict the context given an input word. Here we are going to describe the Skip-gram model only, both for sake of brevity and also because it is the base model used by different graph embedding methods.

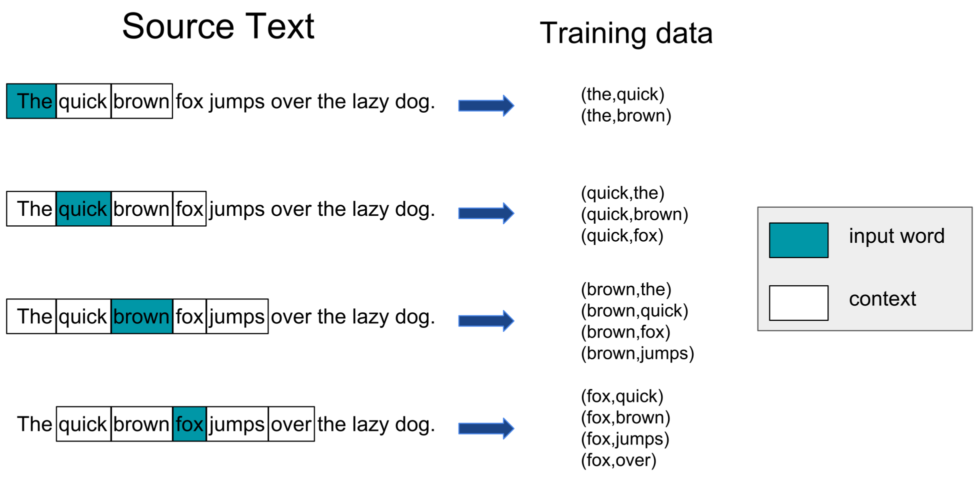


FIGURE 5.2: Example application of Skip-gram model for the generation of training-data for a text corpus.

As already seen in other learning-based models, Word2Vec trains the weights of its shallow neural network for a prediction task. In other words, while the final goal is learning a new representation of a given corpus of text, and at the same time retaining most of the information, the fake task that is being carried out consists of predicting the context, within a predefined window, given an input word w.

The cost function being maximized by Word2Vec is the following:

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta) \tag{5.1}$$

where D is the set of all word-context pairs extracted from the input text corpus. The conditional probability $p(c|w;\theta)$ is defined as:

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} v_{c'} \cdot v_w} \tag{5.2}$$

where v_c and $v_w \in R^d$ are vector representations of the context c and the word w respectively, and C represents the set of all available contexts.

Starting from a text corpus given in input, the model generates a vocabulary containing all the words encountered into the training data. During the generation of training data (Fig. 5.2), a window, in this case of size two, is used to slide over the sentence generating the word-context pairs. Each of these words is represented through one-hot encoding by a vector; this means that supposing a vocabulary of size n, each one-hot vector will have a size equal to n, with $n-1$ components set to zero and one component equal to one, as shown in Fig. 5.3. The adoption of one-hot encoding has an impact on the size of all the one-hot vectors, that will be equal to the number of words contained into the vocabulary. On the other hand, using this type of encoding allows, through multiplying the input word vector v_w by the hidden-layer weight matrix, to select a single row of the hidden-layer weight matrix that represents the vectorial representation of the word w in the new representational space. More specifically, the hidden-layer weight matrix is used as a simple look-up table.

The shallow neural network is composed of one hidden layer and one output layer. The number of neurons that compose the hidden layer represents the number of features f that characterize each word. Therefore, the hidden layer is represented by a matrix with n rows and f columns. Where f represents a hyper-parameter of the model and should be tuned in order to achieve the best trade-off in terms of performance and efficiency.

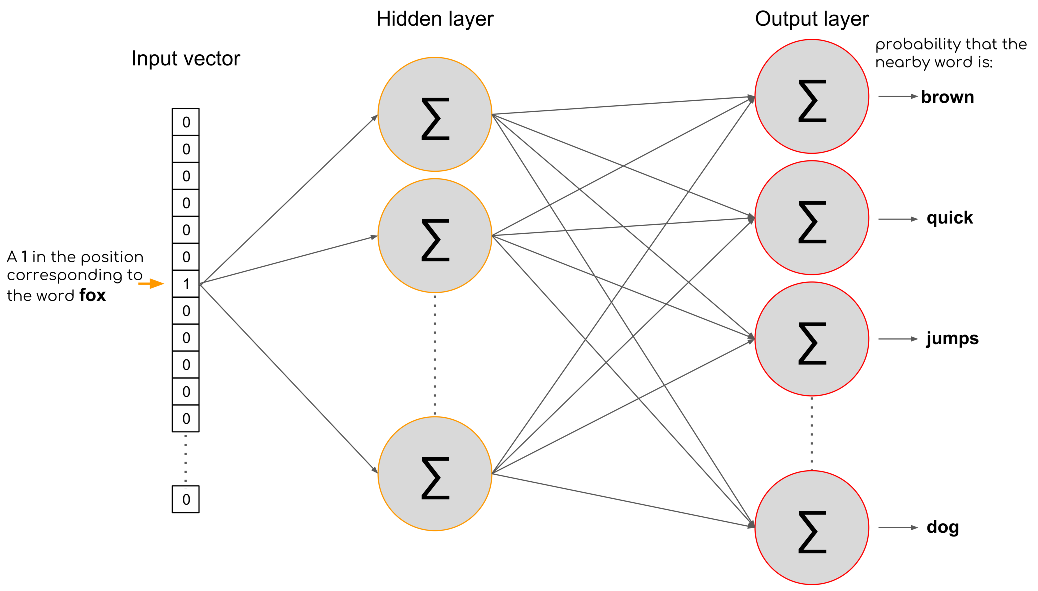

Hidden layer
Output layer
Input vector
probability that the nearby word is:
brown
quick
jumps
dog
A 1 in the position corresponding to the word **fox**
0
0
0
0
0
0
1
0
0
0
0
0
0
Σ

FIGURE 5.3: Example of Word2Vec neural network architecture.

The output vector consists of n components, but differently from the input, does not use one-hot encoding: more than one component can be greater than zero, and their value $\in [0, 1]$ represents the probability that the word corresponding to that position will be part of the context of the input word within a predefined window. The output layer, thanks to the use of softmax function, achieves two desirable characteristics of the output of the neural network: all the values are in the range $[0, 1]$, and all the values sum up to one.

It is important to notice that the model does not learn a different set of weights whether the words belonging to the context are usually found before or after the word w.

In Fig. 5.4 we show, through an example, how Word2Vec computes the probability that the word *dog* will be selected randomly to be nearby the word *fox*. As reported before, the one-hot encoding vector of the word *fox* selects the word vector v_{fox}, which is then multiplied by the context vector $v_{context}$, in this case of the word *dog* (i.e., v_{dog}), and finally, the softmax function is in charge of normalizing the output of the model. It is important to remember that the weights learned are going to reflect the frequency of occurrence for each pair of *words-context* present in the training data.

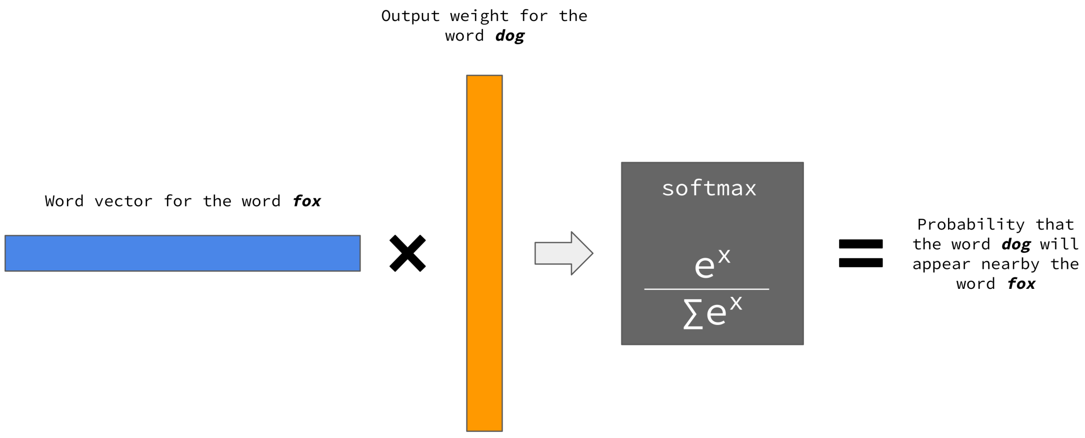

Output weight for the word **dog**
Word vector for the word **fox**
softmax
$\frac{e^x}{\sum e^x}$
Probability that the word **dog** will appear nearby the word **fox**

FIGURE 5.4: Example of Word2Vec computing the probability that the word dog will appear nearby the word fox.

Depending on the number of features f and the number of unique words n contained in the input data, the number of weights that compose the shallow neural network can be relatively high. For instance, if we use $f = 100$ and a corpus with $10,000$ unique words, the number of weights will be equal to two million, i.e., one million for the hidden layer and another million for the output layer.

In order to improve efficiency, scalability and the performance of the Word2Vec model, Mikolov et al. introduce three main modifications: i) to treat common word pairs or phrases as single "words" (i.e., *New_ York* instead of *New* and *York*), ii) to sub-sample frequent words in order to decrease the number of training examples, iii) "negative sampling" is applied mainly in order to limit the number of weights that are being updated while processing new training data.

Treat common word pairs as single words reflect the differences in terms of meaning between the two words the compose the pair, if taken separately, and the pair itself. For example, the word pair "Boston Globe" (a newspaper) has a different meaning than the individual words "Boston" and "Globe". This approach should not be limited to bi-grams, and can be extended to n-grams (e.g., *New_ York_ Times* instead of *New York Times*).

As shown in Fig. 5.2, very common words such as the stop-word *the* in the example, will be present in several training instances without adding meaning or useful information to the word pair. Also, given the frequency of words such as *the* in a generic corpus, we will have a higher number of training instances containing the word *the* than needed. In [221], to address this problem Word2Vec implements a "sub-sampling" scheme, where for each word encountered in the training text, there is a chance that the word will be effectively deleted from the text, and the probability of cutting the word is related to its frequency. The probability that defines whether Word2Vec will keep the word w_i is:

$$P_{keep}(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1\right) \cdot \frac{0.001}{z(w_i)} \tag{5.3}$$

where $z(w_i)$ represents the ratio between the number of occurrences of the word w_i over the total number of words.

Training a neural network generally means to take several training examples, and for each one of them, slightly adjust all the weights using stochastic gradient descent (or other optimization techniques), in order to improve the accuracy of the model by learning from the processed data. As discussed above, the size of the vocabulary, and consequentially the number of weights, could range from one to several million, all of which would need to be updated while processing every training sample. Mikolov et al. overcome this scalability issue by updating only a small portion of the weights for each training example. For instance, when training the neural network on the word pair *fox-quick* we aim to modify the weights of the network in order to have in output a one in correspondence of the word *quick* when the input word is *fox*, and zero for all the other words.

Based on Noise Contrastive Estimation [223], [224], negative sampling aims to update a small portion of the weights of the neural network. In addition to the "positive" weight, we randomly select just a small number k of "negative" words for which the weights are updated. With the term *negative* is intended a word for which the desired output is zero with the current input word. The selection of the negative samples relies on a unigram distribution, and the probability for selecting a word as a negative sample is related to its frequency. In particular, more frequent words are more likely to be selected as negative samples, as described by the following equation:

$$P_{negative}(w_i) = \frac{freq(w_i)^{3/4}}{\sum_{j=0}^{n}(freq(w_i)^{3/4})} \tag{5.4}$$

where $freq(w_i)$ indicates the frequency of the word w_i.

Probably one of the most significant properties of the vectors learned by Word2Vec is that the vector representing a specific word carries also its semantic meaning, and more surprisingly the representation tends to obey to the law of analogy, as shown by the well-known example "Woman is to queen as man is to king", corresponding to:

$$v_{queen} - v_{woman} + v_{man} \approx v_{king} \tag{5.5}$$

where v_{queen}, v_{woman}, v_{man} ,and v_{king} are the vector representation of the words *queen*, *woman*, *man*, and *king* respectively.

5.2.1 Node2Vec

The first work in which the Skip-gram model has been applied to a graph is by Perozzi et al. in [195], followed by Grover and Leskovec with Node2Vec in [196]. Since Node2Vec can be considered a generalization of the DeepWalk algorithm, therefore we will concentrate our focus on Node2Vec only.

The main challenge in applying Word2Vec to a graph is represented by the transformation of a graph in a corpus. In particular, the structure of a group of nodes does not resemble the one of a sentence, a paragraph, or a document: while, in a generic corpus, each word is linked to t previous (resp. consecutive) words, in the case of a graph, the structure is usually more complex, allowing connections potentially between every two nodes.

Node2Vec adds to the Skip-gram model a pre-processing step in charge of generating a set of sentences from a graph by applying a specific sampling strategy. More in detail, Grover and Leskovec resorted to a second-order random walk with two parameters p and q, which guide the walk through the graph. Depending on the values assigned to p and q, the exploration can be carried out in a breadth-first sampling or depth-first sampling fashion.

The *return parameter* p controls the likelihood of immediately revisiting a node in the walk. Setting it to a high value ($> max(q, 1)$) ensures that we are less likely to sample an already-visited node in the following two steps (unless the next node in the walk had no other neighbor). This strategy encourages moderate exploration and avoids 2-hop redundancy in sampling. On the other hand, if p is low ($< min(q, 1)$), it would lead the walk to backtrack a step (Figure 5.5) and this would keep the walk "local" close to the starting node u [196].

The In-out parameter q allows the search to differentiate between "inward" and "outward" nodes. Going back to Figure 5.5, if $q > 1$, the random walk is biased towards nodes close to node t. Such walks obtain a local view of the underlying graph with respect to the start node in the walk and approximate Breadth-first Sampling behavior in the sense that our samples comprise of nodes within a small locality. In contrast, if $q < 1$, the walk is more inclined to visit nodes which are further away from the node t. Such behavior is reflective of Depth-first Sampling which encourages outward exploration [196].

For every node u in the graph, n walks, of fixed length l are being simulated. With the starting node $c_0 = u$, the nodes c_i are sampled by the following distribution:

$$P(c_i = x | C_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} \ if(v,x) \in E \\ 0 \ otherwise \end{cases} \tag{5.6}$$

where π_{vx} represents the unnormalized transition probability between nodes v and x, and Z is the normalizing constant. The unnormalized transition probability is set to $\pi_{vx} = \alpha_{p,q}(t,x) \cdot w_{vx}$, where:

$$\alpha_{p,q}(t,x) = \begin{cases} \frac{1}{p} if d_{tx} = 0 \\ 1 if d_{tx} = 1 \\ \frac{1}{q} if d_{tx} = 2 \end{cases} \tag{5.7}$$

and w_{vx} represents the weight of the edge that links x and v, d_{tx} denotes the shortest path distance between nodes t and x.

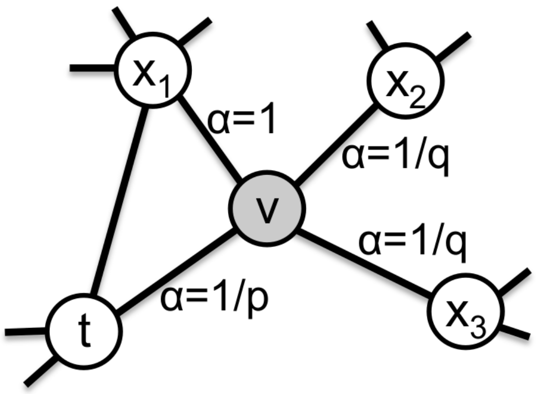


FIGURE 5.5: Node2Vec sampling strategy (from [196])

The use of random walks, to sample neighbors from a graph, brings efficiency advantages both in terms of space and time. The complexity to store the immediate neighbors of every node is $O(|E|)$, whereas for a second-order random walks, which stores the interconnections between the neighbors of every node, the space complexity is $O(a^2|V|)$, where a represents the average degree of the graph. Concerning time complexity, the main advantage of random-walks-based methods is the ability to reuse samples across different starting nodes.

5.2.2 Node2Vec for multilayer networks

With the goal of conceiving a method able to deal with multilayer networks, we started by devising four variations of Node2Vec, that can be organized into two groups, i) method based on a multilayer random walk, and ii) methods based on local (i.e., single-layer) random walk.

Neighborhood-similarity-based methods. In the first group, with the names Sim-ngh (i.e., similar neighborhood) and Dissim-ngh (i.e., dissimilar neighborhood) we identify two Node2Vec implementations inspired by the community detection method dubbed LART (Locally Adaptive Random Transitions) [225]. The similarity between these two methods and the community detection one is limited to how the inter-layers weights are computed. These weights are needed because, differently from Node2Vec algorithm, the second-order random walker present in Sim-ngh (resp. Dissim-ngh) has gained the ability to jump across layers, and in this case, the probability of changing layer is defined to be proportional to the similarity (resp. dissimilarity) of the neighborhood of the current node in the two layers under consideration (i.e. the starting layer and the supposed destination layer). This probability is associated to the inter-layer weight that reflects the similarity (resp. dissimilarity) in local topology between v_i^k and v_i^l and it is defined as the number of edges that the two nodes have in common between the two layers:

$$w_{i;kl} = |N_{i,k} \cap N_{i,l}|$$

In order to express *dissimilarity* among the numerous tested inversion functions we selected:

$$w_{i;kl}^{inverted} = \frac{1}{1 + |N_{i,k} \cap N_{i,l}|}$$

Furthermore, it is important to notice that the weights are in the range $\in [0, 1]$ due to a normalization step.

Context-mixing-based methods. To introduce the other two methods, dubbed Lcm and Ncm, we need to briefly restate the basics of Word2Vec framework, on which Node2Vec and all these variations are based. In Word2Vec the conditional probabilities at the core of the optimization problem are:

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}} \tag{5.8}$$

where v_c and $v_w \in R^d$ are vector representations for the context c and the word w, and C is the set of all available contexts.

We define Lcm (Layer-based context-mixing) and Ncm (Neighborhood-based context-mixing) upon the notion of context mixing, i.e., integrating node-context with information from other layers, depending on a layer-wise (resp. neighborhood-wise) similarity measure. More in detail, Lcm mixes the node's context with information coming from other layers, proportionally to layer similarity. Conversely, by evaluating neighborhood similarity Ncm decides to mix the context by a finer grain w.t.r. to Lcm.

In summary, we propose four variations of Node2Vec, based on the notion of neighborhood similarity (resp. context-mixing), namely Sim-ngh and Dissim-ngh (resp. Lcm and Ncm).

5.3 Evaluation

In order to assess the effectiveness of the proposed methods, we compared their performance with that of two graph embedding methods available designed to deal with multilayer networks, namely Effective Embedding Method for Multiplex Networks (MELL) [218] and Principled Multilayer Network Embedding (PMNE) [216], presented in Sec. 5.1.1.

Our evaluation consists of a twofold comparison that concerns the ability of each method to predict the missing edges from every dataset in Table 5.2 and the alleged quality of the embeddings in terms of information retained.

Concerning the link prediction task, for each dataset we split the set of existing edges into *train* and *testing* sets: the former is used to train each model and the latter to test its link prediction performance. Moreover, all the edges belonging to the test set involve nodes that already exist on the specific layer. We tested all the methods on four train-test split percentages, 60%-40%, 70%-30%, 80%-20%, and 90%-10%.

To assess the presumed existence of an edge, from the point of view of the newly learned representation, we compute the distance for each node pairs in the test edge set, ordered the list of distance values in ascending order (i.e., lower values at the top), and selected the top-10% as predicted edges.

For the second task of the evaluation process, we resort to the CS-Aarhus [226] dataset. The reason behind the use of this dataset relies upon the existence of ground-truth values for all the actors present in the network; ground-truth values represent the membership to a specific research group or department.

We learned a new vectorial representation with all the methods discussed above on increasing training rate in the classification testbed and on all edge available i.e., with a training rate equal to 100% in the clustering testbed. Starting from the newly learned representations, we tested several classification and clustering algorithms and compared their results with the available ground-truth.

5.3.1 Framework setting

To perform a fair comparison, given the high number of parameters that characterize Node2Vec and thus its variants, we tuned each method's parameter in order to obtain the best performance.

For both competing methods and Node2Vec variants, we assumed the embedding dimension to be constant and equal to 128, whereas in the case of MELL, since the embedding vector consists of three components, a *head* and a *tail* vector, and a *layer* vector; considering the size of the layer vector to be negligible, we set the size of both head and tail vectors to be $d/2$ (i.e., 64). Concerning the rest of the parameters of the MELL method, we optimized the number of negative samples k, the regularization coefficient of embedding vector λ, the regularization coefficient of the variance β, and the regularization coefficient for the layer vector γ. For the rest of the methods (i.e., PMNE and the Node2Vec variants) we tuned the following parameters: the length of each walk l, the number of walks per node n, the size of the context window cw, and the parameters that control the exploration of the multilayer graph (i.e., p, q, and r). Please note that the parameter r in PMNE and Node2Vec variants represents the probability to change layer. The best set of parameters for each pair of dataset-method is reported in Table 5.1.

Dataset	Method	Set of best parameters
CS-Aarhus	MELL	$k = 1$, $\lambda = 10$, $\beta = 1$, $\gamma = 4$
Pierre Auger Collaboration		$k = 8$, $\lambda = 10$, $\beta = 1$, $\gamma = 4$
C.Elegans Connectome		$k = 4$, $\lambda = 10$, $\beta = 100$, $\gamma = 1$
CS-Aarhus	PMNE	$l = 100$, $n = 20$, $cw = 14$, $p = 0.15$, $q = 0.15$, $r = 0.15$
Pierre Auger Collaboration		$l = 50$, $n = 16$, $cw = 14$, $p = 0.15$, $q = 0.95$, $r = 0.6$
C.Elegans Connectome		$l = 50$, $n = 18$, $cw = 10$, $p = 0.95$, $q = 0.15$, $r = 0.95$
CS-Aarhus	Sim-ngh	$l = 100$, $n = 20$, $cw = 14$, $p = 0.25$, $q = 0.25$, $r = 0.15$
Pierre Auger Collaboration		$l = 50$, $n = 16$, $cw = 14$, $p = 0.15$, $q = 0.95$, $r = 0.8$
C.Elegans Connectome		$l = 50$, $n = 18$, $cw = 10$, $p = 0.95$, $q = 0.15$, $r = 0.95$
CS-Aarhus	Dissim-ngh	$l = 100$, $n = 20$, $cw = 14$, $p = 0.15$, $q = 0.25$, $r = 0.15$
Pierre Auger Collaboration		$l = 50$, $n = 16$, $cw = 10$, $p = 0.15$, $q = 0.95$, $r = 0.6$
C.Elegans Connectome		$l = 50$, $n = 16$, $cw = 10$, $p = 0.6$, $q = 0.15$, $r = 0.95$
CS-Aarhus	Node2Vec	$l = 100$, $n = 20$, $cw = 14$, $p = 0.8$, $q = 0.15$
Pierre Auger Collaboration		$l = 100$, $n = 16$, $cw = 12$, $p = 0.15$, $q = 0.95$
C.Elegans Connectome		$l = 100$, $n = 16$, $cw = 10$, $p = 0.8$, $q = 0.15$
CS-Aarhus	Lcm	$l = 100$, $n = 20$, $cw = 14$, $p = 0.6$, $q = 0.15$, $r = 0.15$
Pierre Auger Collaboration		$l = 50$, $n = 16$, $cw = 12$, $p = 0.15$, $q = 0.95$, $r = 0.8$
C.Elegans Connectome		$l = 50$, $n = 16$, $cw = 10$, $p = 0.8$, $q = 0.15$, $r = 0.95$
CS-Aarhus	Ncm	$l = 100$, $n = 20$, $cw = 14$, $p = 0.95$, $q = 0.25$, $r = 0.15$
Pierre Auger Collaboration		$l = 50$, $n = 16$, $cw = 12$, $p = 0.15$, $q = 0.95$, $r = 0.8$
C.Elegans Connectome		$l = 50$, $n = 16$, $cw = 10$, $p = 0.8$, $q = 0.15$, $r = 0.95$

TABLE 5.1: Optimal parameters for MELL, PMNE , and Node2Vec variants.

5.3.2 Assessment criteria

We resort to well-known evaluation measures for both tasks. Given the class imbalance nature of the test, in the link prediction testbed, we compared performance results in terms of the area under the precision-recall curve (AUPR), summarized by the trapezoidal rule. Despite being well-known, for the sake of self-sufficiency, we briefly describe their definitions.

The Precision-Recall curve is defined as the plot of the precision versus the recall values, and the area under this curve well summarizes the performance of the classifier under evaluation. The comparison in the classification test relies on two common measures, macro- and micro-average of the F1-score, dubbed micro-F1 and micro-F1. The F1 score is defined as:

$$F1 = \frac{2 * (\text{precision} * \text{recall})}{(\text{precision} + \text{recall})} \tag{5.9}$$

Conversely, the assessment of the performance on the K-means algorithms relies on Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI) measures. NMI is a normalization of the Mutual Information (MI) score to scale the results between 0 and 1.

$$NMI(Y, C) = \frac{2 \cdot I(Y; C)}{[H(Y) + H(C)]} \tag{5.10}$$

AMI is an adjustment of the mutual information score to account for chance. It accounts for the fact that the mutual information is generally higher for two clusterings with a larger number of clusters, regardless of whether there is actually more information shared. For two clusterings U and V, the AMI is given as:

dataset	#nodes	#edges	#layer	directionality
CS-Aarhus [226]	61	620	5	undirected
Pierre Auger Collaboration [229]	514	7153	16	undirected
C.Elegans Connectome [227], [228]	279	5863	3	directed

TABLE 5.2: Main characteristics of evaluation datasets

$$AMI(U,V) = \frac{[MI(U,V) - E(MI(U,V))]}{[avg(H(U),H(V)) - E(MI(U,V))]} \tag{5.11}$$

where H represents the entropy associated with the partitioning U, and E is the expected mutual information.

5.3.3 Datasets

Table 5.2 reports the size of set $\mathcal{V}$, the number of edges in all layers, the number of layers, and the directionality of network (i.e., directed or undirected). We used three real-world multilayer networks of different type, namely *social*, *co-authorship*, and *neural*. The first dataset, dubbed AUCS [226], is a social network that includes both online and offline connections for a total of 5 layers. The 61 nodes in the multilayer network represent professors, postdoctoral researchers, Ph.D. students and administrative staff of a university department. The second dataset is a co-authorship network, where the layers represent the main topic/keyword of the publication. The network is composed by 16 layers, relating to topics ranging from *neutrinos* to *astrophysical-scenarios*. The third dataset, dubbed Caenorhabditis elegans connectome [227], [228] is a neural network where layers correspond to different synaptic junctions: electric, chemical monadic, and chemical polyadic.

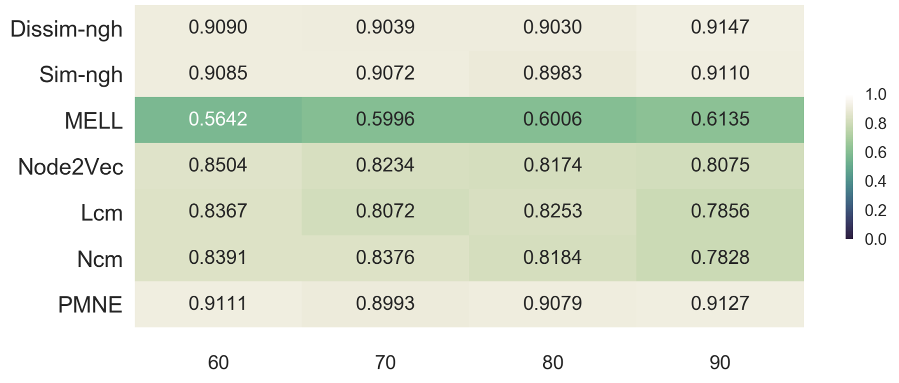
5.4 Experimental Results

In this section we show preliminary results of ongoing experiments concerning the extension of Node2Vec framework and the competing methods presented in Sect. 5.2.2. According to what previously stated in Sect. 5.3, we organize the presentation of the results into two parts: i) link prediction performance and ii) performance comparison of all the methods described in the previous sections, in terms of embedding quality, corresponding to information retained.

5.4.1 Link prediction

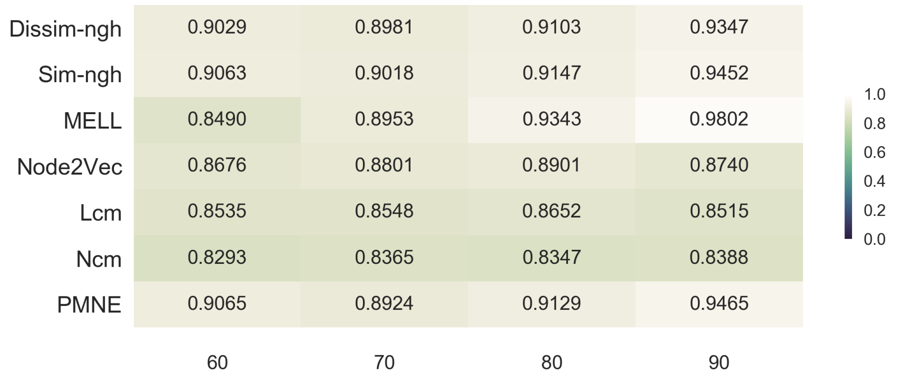
As previously described in Sect. 5.3, we compared the performance of Sim-ngh, Dissim-ngh, Lcm, and Ncm to the competing methods Node2Vec, PMNE, and MELL. We used four training rates, namely 60%, 70%, 80%, and 90%, and incremented the number of epochs from 1 to 250. To improve readability AUPR scores for varying training-rate and different methods, are reported in heatmap format.

One general remark is that MELL achieves the best overall performance at the cost of efficiency, i.e., in order to achieve best results MELL needs a number



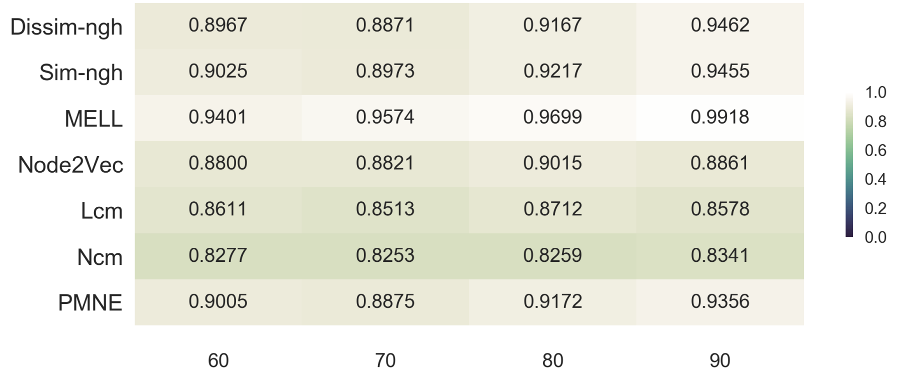
	60	70	80	90
Dissim-ngh	0.9090	0.9039	0.9030	0.9147
Sim-ngh	0.9085	0.9072	0.8983	0.9110
MELL	0.5642	0.5996	0.6006	0.6135
Node2Vec	0.8504	0.8234	0.8174	0.8075
Lcm	0.8367	0.8072	0.8253	0.7856
Ncm	0.8391	0.8376	0.8184	0.7828
PMNE	0.9111	0.8993	0.9079	0.9127

(a) 1 epoch



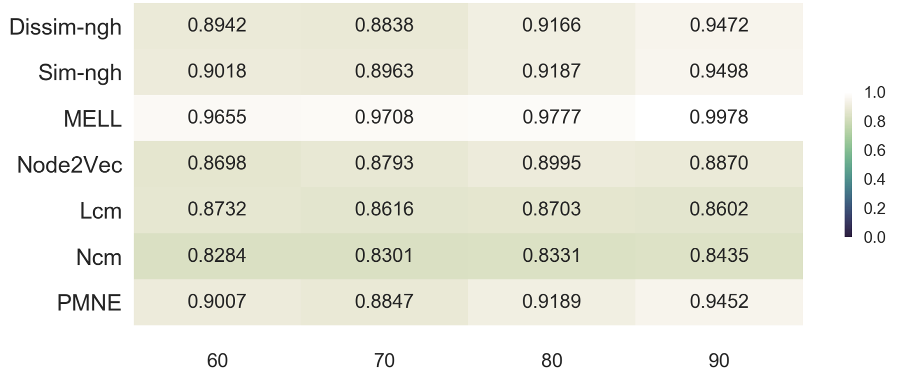
	60	70	80	90
Dissim-ngh	0.9029	0.8981	0.9103	0.9347
Sim-ngh	0.9063	0.9018	0.9147	0.9452
MELL	0.8490	0.8953	0.9343	0.9802
Node2Vec	0.8676	0.8801	0.8901	0.8740
Lcm	0.8535	0.8548	0.8652	0.8515
Ncm	0.8293	0.8365	0.8347	0.8388
PMNE	0.9065	0.8924	0.9129	0.9465

(b) 10 epochs



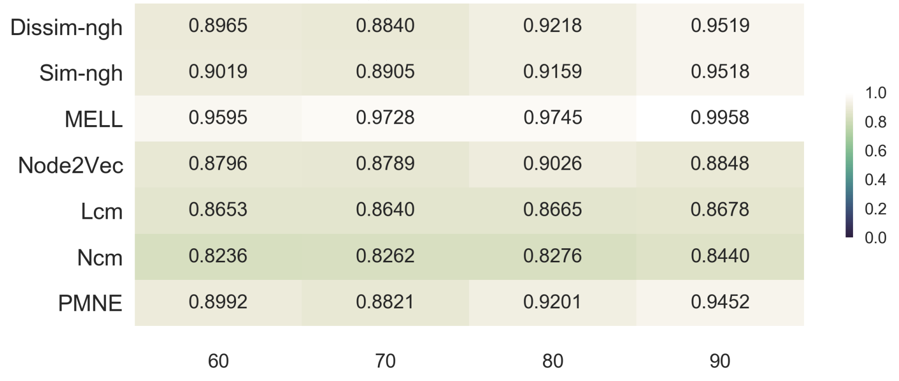
	60	70	80	90
Dissim-ngh	0.8967	0.8871	0.9167	0.9462
Sim-ngh	0.9025	0.8973	0.9217	0.9455
MELL	0.9401	0.9574	0.9699	0.9918
Node2Vec	0.8800	0.8821	0.9015	0.8861
Lcm	0.8611	0.8513	0.8712	0.8578
Ncm	0.8277	0.8253	0.8259	0.8341
PMNE	0.9005	0.8875	0.9172	0.9356

(c) 25 epochs



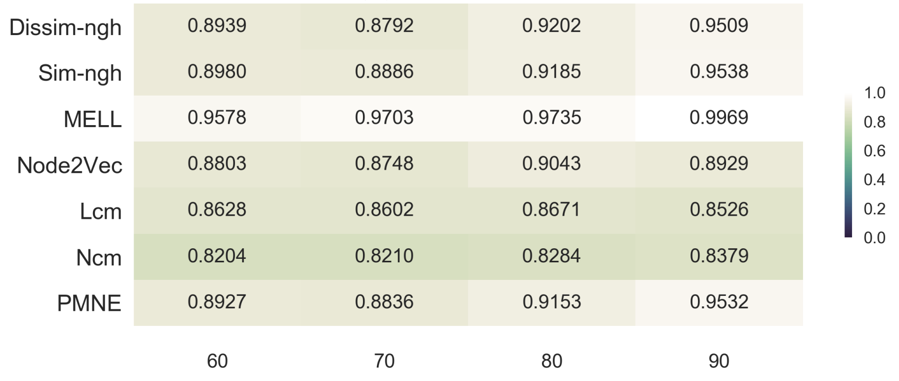
	60	70	80	90
Dissim-ngh	0.8942	0.8838	0.9166	0.9472
Sim-ngh	0.9018	0.8963	0.9187	0.9498
MELL	0.9655	0.9708	0.9777	0.9978
Node2Vec	0.8698	0.8793	0.8995	0.8870
Lcm	0.8732	0.8616	0.8703	0.8602
Ncm	0.8284	0.8301	0.8331	0.8435
PMNE	0.9007	0.8847	0.9189	0.9452

(d) 50 epochs



	60	70	80	90
Dissim-ngh	0.8965	0.8840	0.9218	0.9519
Sim-ngh	0.9019	0.8905	0.9159	0.9518
MELL	0.9595	0.9728	0.9745	0.9958
Node2Vec	0.8796	0.8789	0.9026	0.8848
Lcm	0.8653	0.8640	0.8665	0.8678
Ncm	0.8236	0.8262	0.8276	0.8440
PMNE	0.8992	0.8821	0.9201	0.9452

(e) 100 epochs



	60	70	80	90
Dissim-ngh	0.8939	0.8792	0.9202	0.9509
Sim-ngh	0.8980	0.8886	0.9185	0.9538
MELL	0.9578	0.9703	0.9735	0.9969
Node2Vec	0.8803	0.8748	0.9043	0.8929
Lcm	0.8628	0.8602	0.8671	0.8526
Ncm	0.8204	0.8210	0.8284	0.8379
PMNE	0.8927	0.8836	0.9153	0.9532

(f) 250 epochs

FIGURE 5.6: Performance results on CS-Aarhus data with an increasing number of epochs. *(Best viewed in color version, available in electronic format)*

	60	70	80	90
Dissim-ngh	0.8270	0.8401	0.8631	0.8672
Sim-ngh	0.8285	0.8394	0.8632	0.8774
MELL	0.5081	0.5225	0.4886	0.5212
Node2Vec	0.7926	0.7656	0.7951	0.7732
Lcm	0.7624	0.7434	0.7618	0.7569
Ncm	0.7748	0.7406	0.7685	0.7601
PMNE	0.8311	0.8371	0.8638	0.8723

(a) 1 epoch

	60	70	80	90
Dissim-ngh	0.8594	0.8861	0.8912	0.9081
Sim-ngh	0.8621	0.8808	0.8919	0.9148
MELL	0.6200	0.5949	0.5576	0.5427
Node2Vec	0.8503	0.8561	0.8635	0.8885
Lcm	0.8685	0.8730	0.8814	0.8830
Ncm	0.8664	0.8786	0.8770	0.8878
PMNE	0.8607	0.8856	0.8880	0.9073

(b) 10 epochs

	60	70	80	90
Dissim-ngh	0.8522	0.8676	0.8783	0.9004
Sim-ngh	0.8468	0.8703	0.8776	0.9072
MELL	0.6576	0.6956	0.5742	0.5555
Node2Vec	0.8552	0.8584	0.8736	0.9002
Lcm	0.8650	0.8678	0.8816	0.8745
Ncm	0.8632	0.8672	0.8767	0.8794
PMNE	0.8475	0.8685	0.8801	0.8986

(c) 25 epochs

	60	70	80	90
Dissim-ngh	0.8338	0.8576	0.8767	0.8930
Sim-ngh	0.8370	0.8569	0.8696	0.8972
MELL	0.7589	0.8050	0.6338	0.6735
Node2Vec	0.8358	0.8469	0.8661	0.8954
Lcm	0.8542	0.8582	0.8773	0.8637
Ncm	0.8539	0.8576	0.8761	0.8641
PMNE	0.8379	0.8584	0.8747	0.8957

(d) 50 epochs

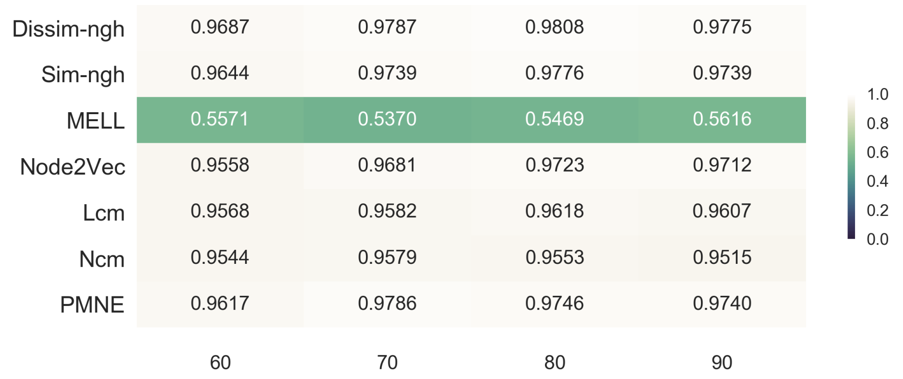
	60	70	80	90
Dissim-ngh	0.8281	0.8518	0.8659	0.8915
Sim-ngh	0.8318	0.8527	0.8659	0.8947
MELL	0.9058	0.9240	0.6870	0.7421
Node2Vec	0.8177	0.8342	0.8584	0.8997
Lcm	0.8423	0.8511	0.8722	0.8635
Ncm	0.8460	0.8525	0.8732	0.8698
PMNE	0.8330	0.8558	0.8708	0.8975

(e) 100 epochs

	60	70	80	90
Dissim-ngh	0.8218	0.8496	0.8597	0.8891
Sim-ngh	0.8247	0.8493	0.8612	0.8893
MELL	0.9429	0.9553	0.8496	0.9121
Node2Vec	0.8054	0.8302	0.8546	0.8882
Lcm	0.8421	0.8455	0.8682	0.8694
Ncm	0.8402	0.8457	0.8646	0.8686
PMNE	0.8241	0.8482	0.8617	0.8927

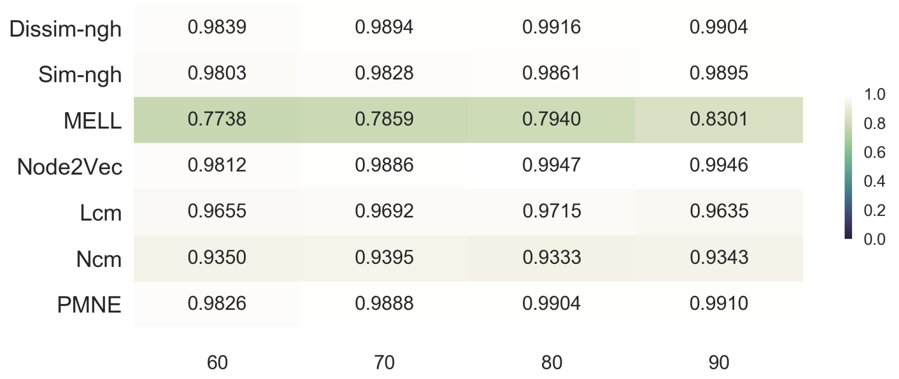
(f) 250 epochs

FIGURE 5.7: Performance results on C.Elegans Connectome data with an increasing number of epochs (i.e., 1, 10, 25, 50, 100, and 250).



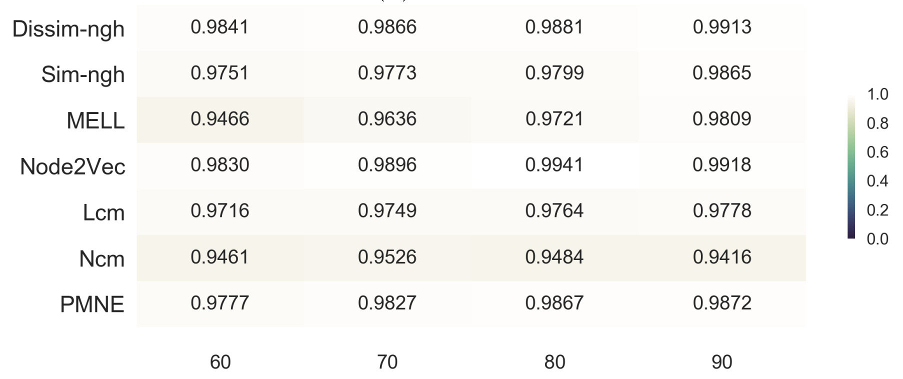
	60	70	80	90
Dissim-ngh	0.9687	0.9787	0.9808	0.9775
Sim-ngh	0.9644	0.9739	0.9776	0.9739
MELL	0.5571	0.5370	0.5469	0.5616
Node2Vec	0.9558	0.9681	0.9723	0.9712
Lcm	0.9568	0.9582	0.9618	0.9607
Ncm	0.9544	0.9579	0.9553	0.9515
PMNE	0.9617	0.9786	0.9746	0.9740

(a) 1 epoch



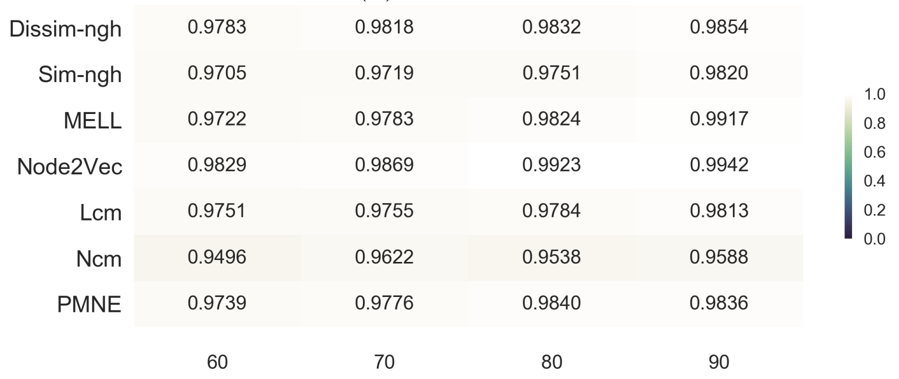
	60	70	80	90
Dissim-ngh	0.9839	0.9894	0.9916	0.9904
Sim-ngh	0.9803	0.9828	0.9861	0.9895
MELL	0.7738	0.7859	0.7940	0.8301
Node2Vec	0.9812	0.9886	0.9947	0.9946
Lcm	0.9655	0.9692	0.9715	0.9635
Ncm	0.9350	0.9395	0.9333	0.9343
PMNE	0.9826	0.9888	0.9904	0.9910

(b) 10 epochs



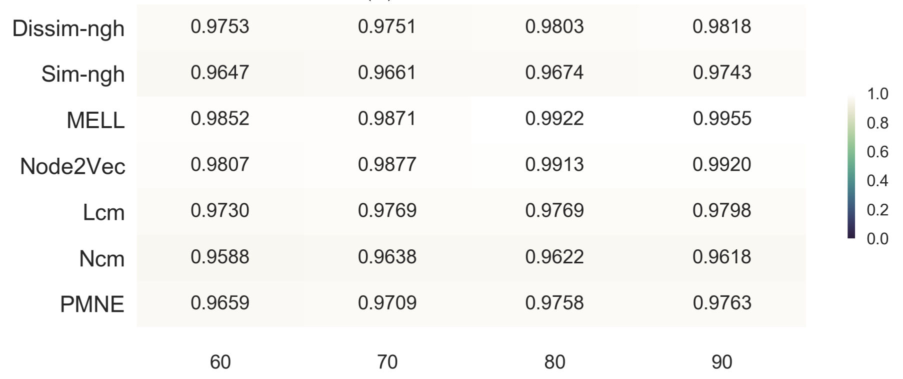
	60	70	80	90
Dissim-ngh	0.9841	0.9866	0.9881	0.9913
Sim-ngh	0.9751	0.9773	0.9799	0.9865
MELL	0.9466	0.9636	0.9721	0.9809
Node2Vec	0.9830	0.9896	0.9941	0.9918
Lcm	0.9716	0.9749	0.9764	0.9778
Ncm	0.9461	0.9526	0.9484	0.9416
PMNE	0.9777	0.9827	0.9867	0.9872

(c) 25 epochs



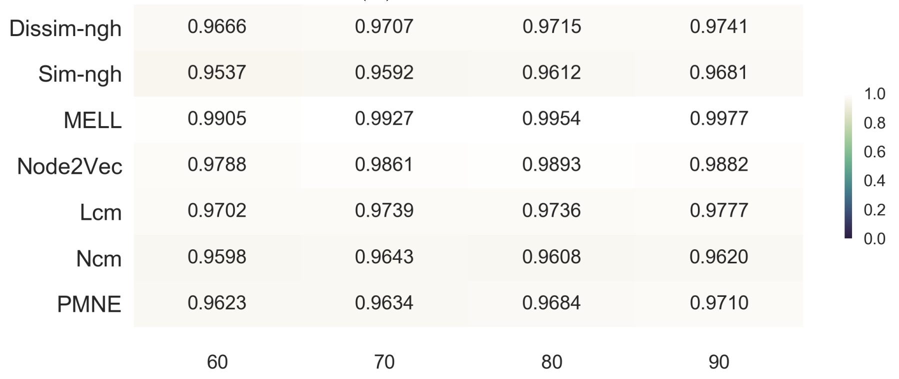
	60	70	80	90
Dissim-ngh	0.9783	0.9818	0.9832	0.9854
Sim-ngh	0.9705	0.9719	0.9751	0.9820
MELL	0.9722	0.9783	0.9824	0.9917
Node2Vec	0.9829	0.9869	0.9923	0.9942
Lcm	0.9751	0.9755	0.9784	0.9813
Ncm	0.9496	0.9622	0.9538	0.9588
PMNE	0.9739	0.9776	0.9840	0.9836

(d) 50 epochs



	60	70	80	90
Dissim-ngh	0.9753	0.9751	0.9803	0.9818
Sim-ngh	0.9647	0.9661	0.9674	0.9743
MELL	0.9852	0.9871	0.9922	0.9955
Node2Vec	0.9807	0.9877	0.9913	0.9920
Lcm	0.9730	0.9769	0.9769	0.9798
Ncm	0.9588	0.9638	0.9622	0.9618
PMNE	0.9659	0.9709	0.9758	0.9763

(e) 100 epochs



	60	70	80	90
Dissim-ngh	0.9666	0.9707	0.9715	0.9741
Sim-ngh	0.9537	0.9592	0.9612	0.9681
MELL	0.9905	0.9927	0.9954	0.9977
Node2Vec	0.9788	0.9861	0.9893	0.9882
Lcm	0.9702	0.9739	0.9736	0.9777
Ncm	0.9598	0.9643	0.9608	0.9620
PMNE	0.9623	0.9634	0.9684	0.9710

(f) 250 epochs

FIGURE 5.8: Performance results on Pierre Auger Collaboration data with an increasing number of epochs (i.e., 1, 10, 25, 50, 100, and 250).

of epochs equal to or higher than 100, and in general the higher the better. Conversely, Node2Vec-based methods generally achieve the best performance with just 1 epoch, and increasing the number of iterations rarely leads to an increase in performance.

In all three datasets, CS-Aarhus, C.Elegans Connectome, and Pierre Auger Collaboration network, we see two recurrent patterns. While the MELL algorithm increases its performance with a growing number of epochs, at the same time Node2Vec-based algorithms slightly decrease their performance with an increasing number of epochs and usually, the best result is achieved with 1, 10 or at most 25 epochs.

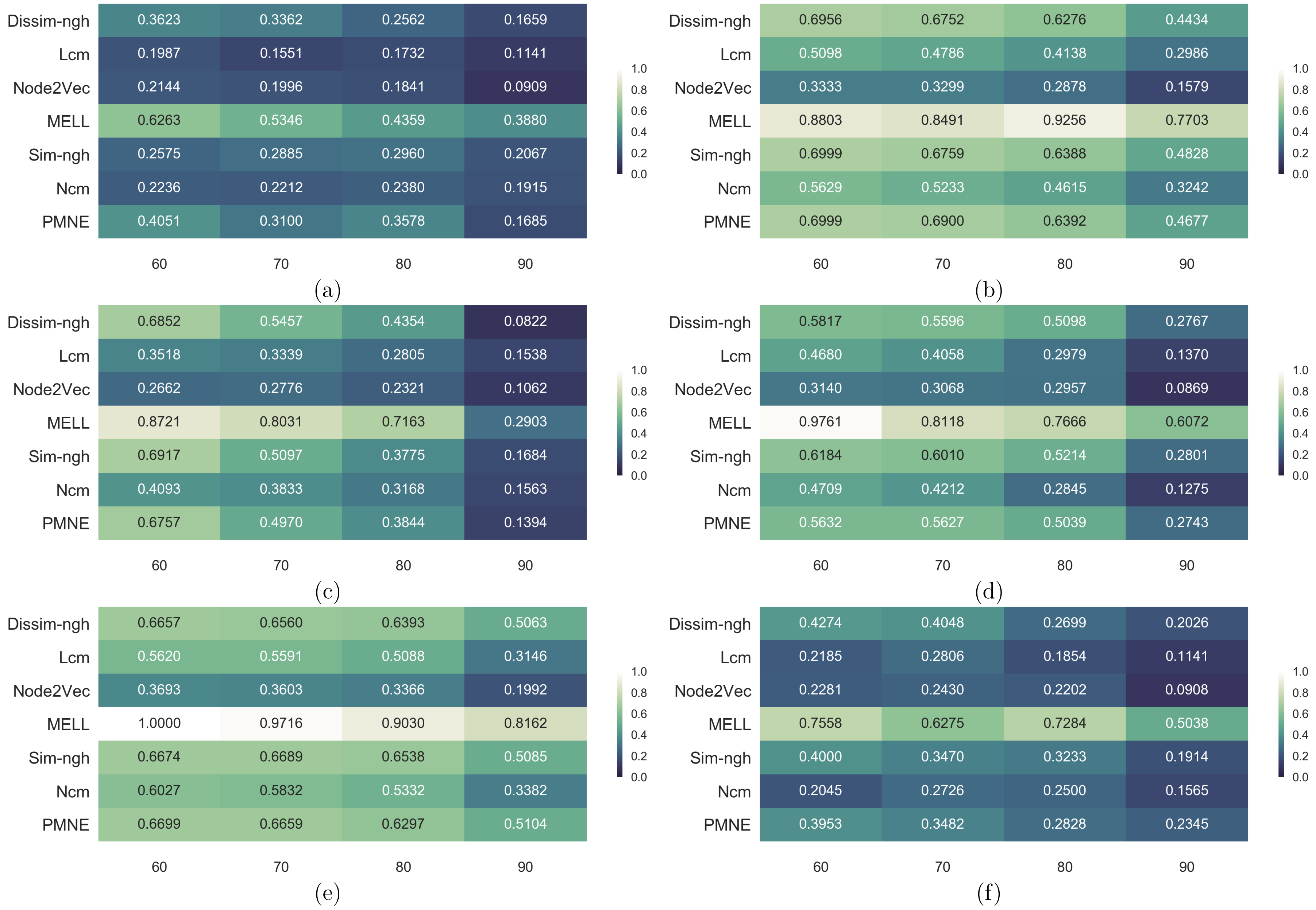
Figure 5.6 reports performance results on CS-Aarhus data. Among all methods, the best performance result with only 1 epoch is achieved by Dissim-ngh in correspondence of a 90% training rate with a 0.915 of AUPR, closely followed by PMNE and Sim-ngh. The other algorithms are separated by 0.1, with the exception of MELL, that sets the bar between 0.61 and 0.56. Among the Node2Vec-based methods, we see a distinctive pattern, where Sim-ngh and Dissim-ngh, albeit defined on opposite intuitions, have similar results and lead the rank, whereas Lcm and Ncm are relegated to the bottom of the rank. Notice that in this case, 25 epochs are enough for MELL algorithm to find a good arrangement of its embedding structure and become the best performing method.

The inherent complexity of the C.Elegans Connectome data requires a higher number of epochs needed for all the methods (see Figure 5.7). In particular, MELL needs 100 epochs to achieve the best result for 60% and 70% training rate, and 250 epochs in the case of 90% training rate. However, for the remaining case, i.e., a training rate equal to 80%, 250 iterations are not enough to exceed the other methods. Conversely, Sim-ngh and Dissim-ngh start, with 1 epoch, among the best with the PMNE algorithm, but are then reached by Lcm and Ncm at 10 iterations and then outperformed by Lcm and Ncm.

The Pierre Auger Collaboration data seems to be easy to learn and make prediction on by Node2Vec-based algorithms. With only 1 iteration they achieve near-optimal performance with a minimum AUPR value of 0.954 (Lcm method and 60% training rate) and a maximum value of 0.981 in correspondence of a training rate of 80% and the Dissim-ngh method (results shown in Figure 5.8). On the other hand, with the maximum number of iterations, all methods, including MELL, achieve near-optimal performance in all cases, ranging from around 0.96 to 0.998.

5.4.2 Classification and Clustering

In this subsection, we describe the performance results obtained in the classification and clustering task described in Section 5.3. In the former task, after learning a new representation, we tested six well-known classification algorithms comparing their prediction to the ground-truth values. In particular, we resort to macro-F1 and micro-F1 measures and to the following classification methods: *k-nearest neighbors* [230], *linear support vector classifier* [231], *decision*



(a)

	60	70	80	90
Dissim-ngh	0.3623	0.3362	0.2562	0.1659
Lcm	0.1987	0.1551	0.1732	0.1141
Node2Vec	0.2144	0.1996	0.1841	0.0909
MELL	0.6263	0.5346	0.4359	0.3880
Sim-ngh	0.2575	0.2885	0.2960	0.2067
Ncm	0.2236	0.2212	0.2380	0.1915
PMNE	0.4051	0.3100	0.3578	0.1685

(b)

	60	70	80	90
Dissim-ngh	0.6956	0.6752	0.6276	0.4434
Lcm	0.5098	0.4786	0.4138	0.2986
Node2Vec	0.3333	0.3299	0.2878	0.1579
MELL	0.8803	0.8491	0.9256	0.7703
Sim-ngh	0.6999	0.6759	0.6388	0.4828
Ncm	0.5629	0.5233	0.4615	0.3242
PMNE	0.6999	0.6900	0.6392	0.4677

(c)

	60	70	80	90
Dissim-ngh	0.6852	0.5457	0.4354	0.0822
Lcm	0.3518	0.3339	0.2805	0.1538
Node2Vec	0.2662	0.2776	0.2321	0.1062
MELL	0.8721	0.8031	0.7163	0.2903
Sim-ngh	0.6917	0.5097	0.3775	0.1684
Ncm	0.4093	0.3833	0.3168	0.1563
PMNE	0.6757	0.4970	0.3844	0.1394

(d)

	60	70	80	90
Dissim-ngh	0.5817	0.5596	0.5098	0.2767
Lcm	0.4680	0.4058	0.2979	0.1370
Node2Vec	0.3140	0.3068	0.2957	0.0869
MELL	0.9761	0.8118	0.7666	0.6072
Sim-ngh	0.6184	0.6010	0.5214	0.2801
Ncm	0.4709	0.4212	0.2845	0.1275
PMNE	0.5632	0.5627	0.5039	0.2743

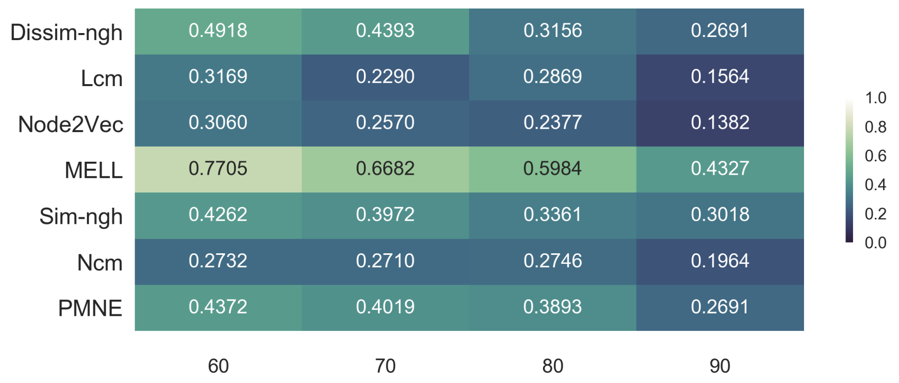
(e)

	60	70	80	90
Dissim-ngh	0.6657	0.6560	0.6393	0.5063
Lcm	0.5620	0.5591	0.5088	0.3146
Node2Vec	0.3693	0.3603	0.3366	0.1992
MELL	1.0000	0.9716	0.9030	0.8162
Sim-ngh	0.6674	0.6689	0.6538	0.5085
Ncm	0.6027	0.5832	0.5332	0.3382
PMNE	0.6699	0.6659	0.6297	0.5104

(f)

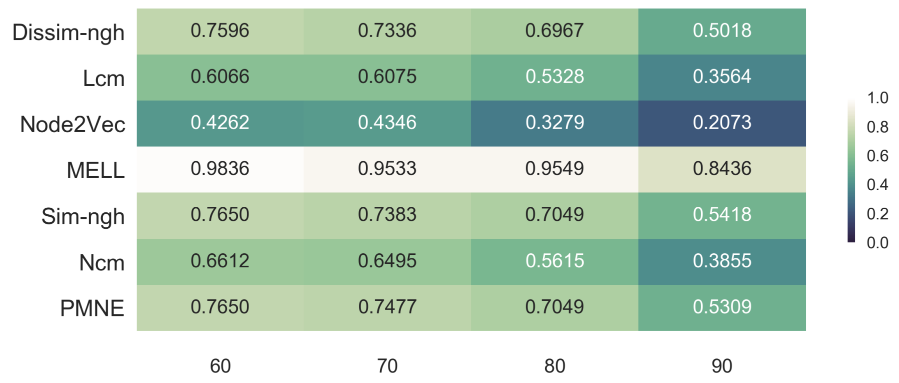
	60	70	80	90
Dissim-ngh	0.4274	0.4048	0.2699	0.2026
Lcm	0.2185	0.2806	0.1854	0.1141
Node2Vec	0.2281	0.2430	0.2202	0.0908
MELL	0.7558	0.6275	0.7284	0.5038
Sim-ngh	0.4000	0.3470	0.3233	0.1914
Ncm	0.2045	0.2726	0.2500	0.1565
PMNE	0.3953	0.3482	0.2828	0.2345

FIGURE 5.9: Classification performance results of the following classifiers with the macro-F1 measure: **a)** decision tree classifier, **b)** linear support vector classifier, **c)** gaussian naive Bayes classifier, **d)** k-nearest neighbors, **e)** multi-layer perceptron classifier, and **f)** random forest classifier.



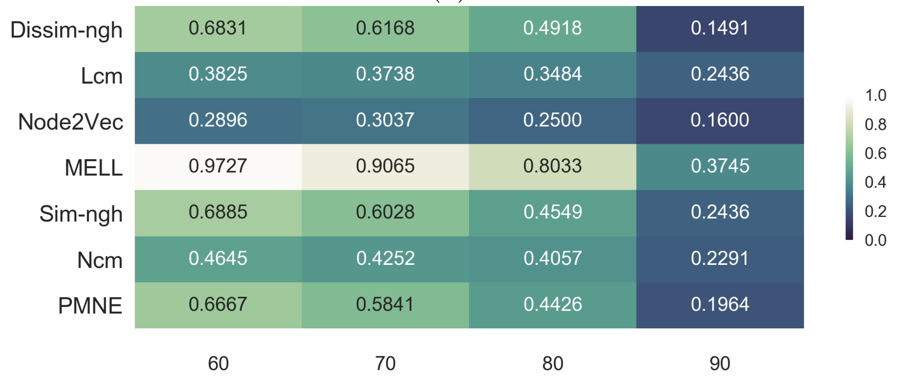
	60	70	80	90
Dissim-ngh	0.4918	0.4393	0.3156	0.2691
Lcm	0.3169	0.2290	0.2869	0.1564
Node2Vec	0.3060	0.2570	0.2377	0.1382
MELL	0.7705	0.6682	0.5984	0.4327
Sim-ngh	0.4262	0.3972	0.3361	0.3018
Ncm	0.2732	0.2710	0.2746	0.1964
PMNE	0.4372	0.4019	0.3893	0.2691

(a)



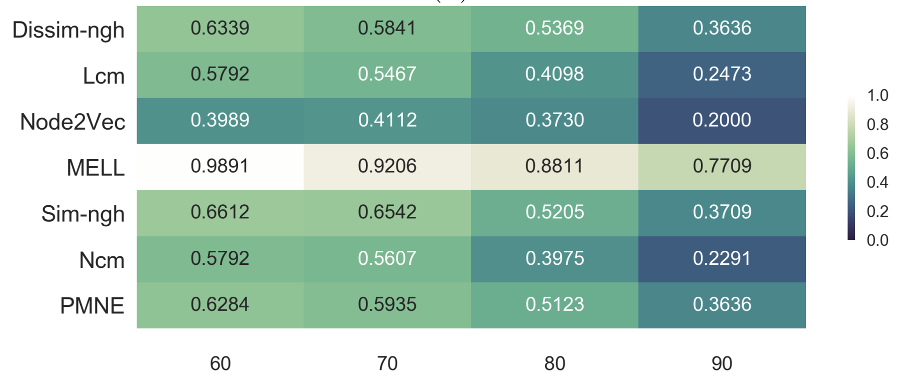
	60	70	80	90
Dissim-ngh	0.7596	0.7336	0.6967	0.5018
Lcm	0.6066	0.6075	0.5328	0.3564
Node2Vec	0.4262	0.4346	0.3279	0.2073
MELL	0.9836	0.9533	0.9549	0.8436
Sim-ngh	0.7650	0.7383	0.7049	0.5418
Ncm	0.6612	0.6495	0.5615	0.3855
PMNE	0.7650	0.7477	0.7049	0.5309

(b)



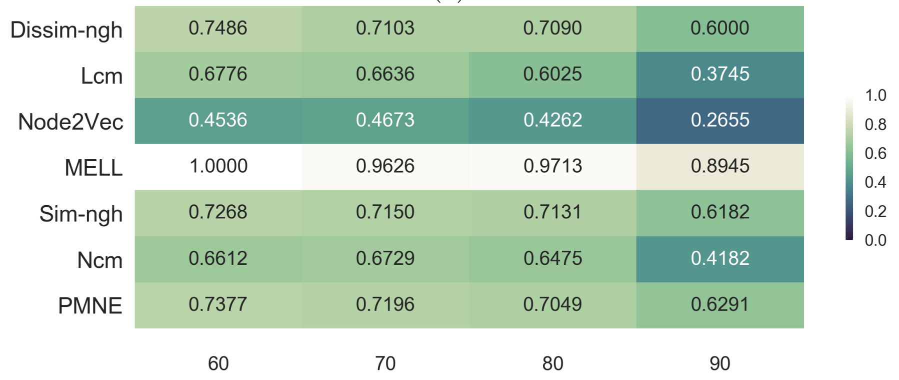
	60	70	80	90
Dissim-ngh	0.6831	0.6168	0.4918	0.1491
Lcm	0.3825	0.3738	0.3484	0.2436
Node2Vec	0.2896	0.3037	0.2500	0.1600
MELL	0.9727	0.9065	0.8033	0.3745
Sim-ngh	0.6885	0.6028	0.4549	0.2436
Ncm	0.4645	0.4252	0.4057	0.2291
PMNE	0.6667	0.5841	0.4426	0.1964

(c)



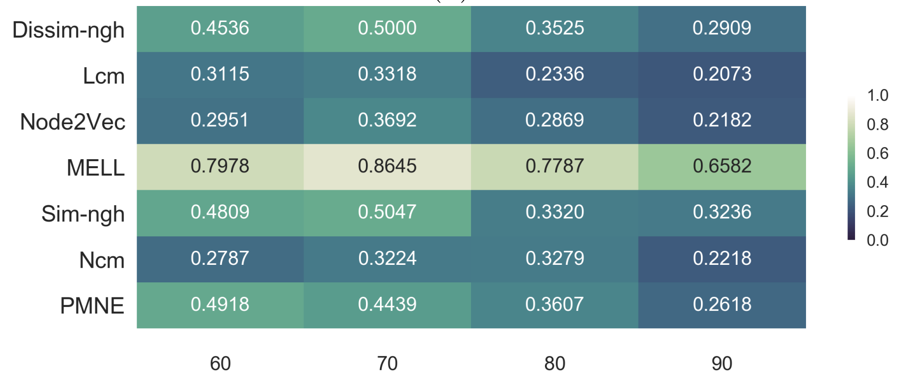
	60	70	80	90
Dissim-ngh	0.6339	0.5841	0.5369	0.3636
Lcm	0.5792	0.5467	0.4098	0.2473
Node2Vec	0.3989	0.4112	0.3730	0.2000
MELL	0.9891	0.9206	0.8811	0.7709
Sim-ngh	0.6612	0.6542	0.5205	0.3709
Ncm	0.5792	0.5607	0.3975	0.2291
PMNE	0.6284	0.5935	0.5123	0.3636

(d)



	60	70	80	90
Dissim-ngh	0.7486	0.7103	0.7090	0.6000
Lcm	0.6776	0.6636	0.6025	0.3745
Node2Vec	0.4536	0.4673	0.4262	0.2655
MELL	1.0000	0.9626	0.9713	0.8945
Sim-ngh	0.7268	0.7150	0.7131	0.6182
Ncm	0.6612	0.6729	0.6475	0.4182
PMNE	0.7377	0.7196	0.7049	0.6291

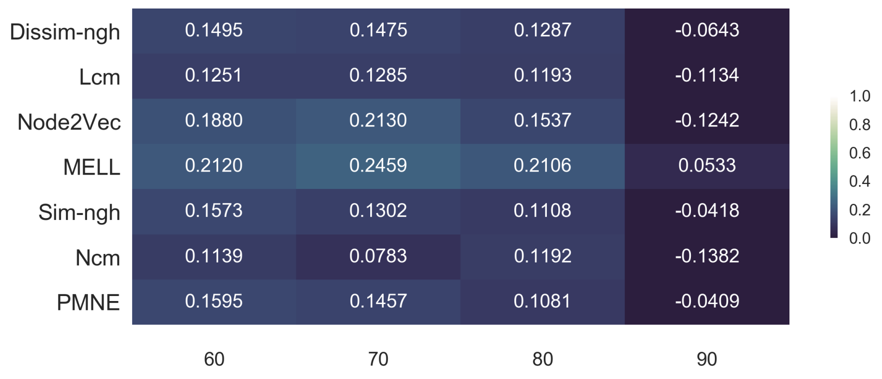
(e)



	60	70	80	90
Dissim-ngh	0.4536	0.5000	0.3525	0.2909
Lcm	0.3115	0.3318	0.2336	0.2073
Node2Vec	0.2951	0.3692	0.2869	0.2182
MELL	0.7978	0.8645	0.7787	0.6582
Sim-ngh	0.4809	0.5047	0.3320	0.3236
Ncm	0.2787	0.3224	0.3279	0.2218
PMNE	0.4918	0.4439	0.3607	0.2618

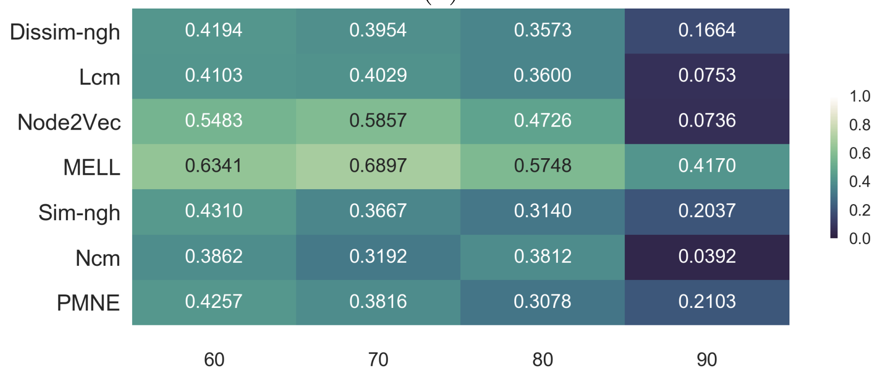
(f)

FIGURE 5.10: Classification performance results of the following classifiers with the micro-F1 measure: **a)** decision tree classifier, **b)** linear support vector classifier, **c)** gaussian naive Bayes classifier, **d)** k-nearest neighbors, **e)** multi-layer perceptron classifier, and **f)** random forest classifier.



	60	70	80	90
Dissim-ngh	0.1495	0.1475	0.1287	-0.0643
Lcm	0.1251	0.1285	0.1193	-0.1134
Node2Vec	0.1880	0.2130	0.1537	-0.1242
MELL	0.2120	0.2459	0.2106	0.0533
Sim-ngh	0.1573	0.1302	0.1108	-0.0418
Ncm	0.1139	0.0783	0.1192	-0.1382
PMNE	0.1595	0.1457	0.1081	-0.0409

(a)



	60	70	80	90
Dissim-ngh	0.4194	0.3954	0.3573	0.1664
Lcm	0.4103	0.4029	0.3600	0.0753
Node2Vec	0.5483	0.5857	0.4726	0.0736
MELL	0.6341	0.6897	0.5748	0.4170
Sim-ngh	0.4310	0.3667	0.3140	0.2037
Ncm	0.3862	0.3192	0.3812	0.0392
PMNE	0.4257	0.3816	0.3078	0.2103

(b)

FIGURE 5.11: K-means performance results on CS-Aarhus datasets with AMI **a)** and NMI **b)** evaluation metric.

tree classifier [232], *random forest classifier* [233], *multi-layer perceptron classifier* [234], [235], and *gaussian naive Bayes classifier* [236].

Analyzing the results in Figure 5.9 and Figure 5.10, and putting aside the comparison between the different classifiers, we concentrate our effort to estimate how easily and with which results, a generic classifier can correctly predict the class of membership with only the information contained into the learned representation at its disposable.

One general remark is that, for both evaluation metrics, all the methods based on random walks and the Word2Vec model are characterized by inferior performance than MELL algorithm. However, it is important to point out that in most cases all methods, including MELL, show worse performance with an increasing training rate, highlighting the limitations of this approach for complex data. Among Node2Vec-based algorithms, Sim-ngh, Dissim-ngh, and PMNE show consistently better performance than Node2Vec, Ncm, and Lcm.

5.5 Conclusion

In this chapter, we described some preliminary results in pursuing the development of a new network-embedding method for multilayer networks. To this end, we proposed several extensions of the Node2Vec algorithm. While performance results, in the link prediction task, are similar among the proposed methods and their competitors, the classification and clustering testbeds shed light about the inability of random-walk-based algorithms to extrapolate useful information about every single actor of the network to build a new representation able to predict the class of membership.

In most cases, better results are obtained by MELL algorithm, especially in the classification and clustering task. Albeit MELL is based on a regression framework, and is less efficient than Node2Vec and its variants (PMNE included), in terms of number of epochs needed to achieve the best performance, MELL algorithm resulted to be a better choice for both the link prediction task and especially the quality of its embeddings.

Chapter 6

Conclusions and future work

6.1 Conclusions

The research presented in this thesis focused on the development of methods in the context of complex network systems to investigate and characterize, in a quantitative manner, user behaviors in OSNs. One of the main contributions of this work is represented by the mlALCR ranking algorithm, which tackles the novel problem of identifying and characterizing opposite behaviors that users may alternately exhibit over the multiple layers of a complex network. In this respect, we proposed the first topology-driven ranking method for *alternate lurker-contributor behaviors* on a multilayer OSN, which is designed to identify users that behave as lurkers (resp. contributors) in one layer while conversely acting as contributors (resp. lurker) in one or many of the other layers. We empirically demonstrated, over four real-world multilayer networks, the significance and uniqueness of mlALCR solutions.

The second part of this research project has been devoted to leveraging machine-learning techniques to learn models from data in the context of behavioral analysis. In this context, we faced the tasks of identifying and ranking both lurking and automated (i.e., bot) behaviors in OSNs. Tackling the former problem, we developed a robust, multi-platform, supervised lurker-ranking model upon the unsupervised, eigenvector-centrality-based models existing in the literature. To this end, we proposed a learning-to-rank framework, named LTLR, whose key aspect is the capability of exploiting different types of information available from multiple OSNs in order to build training data tuples over different subspaces of features. Regarding the latter problem, we developed a supervised ranking model, that leverages different behavioral signals of bot activity. The proposed learning-to-rank framework, dubbed LTRSB, exploits the most relevant existing methods on bot detection for enhanced feature extraction, and state-of-the-art learning-to-rank methods for the optimization. We proved the meaningfulness and effectiveness of both framework on real-world data, according to different assessment criteria.

Finally, we described preliminary results in pursuing the development of a new network-embedding method for multilayer networks. While performance results, in the link prediction task, are similar for the proposed methods and their competitors, the classification and clustering testbeds shed light about the inability of random-walk-based algorithms to extrapolate enough useful information about every single actor of the network in order to build a new representation able to predict the class of membership.

6.2 Future work

Over the past decades, we have witnessed the introduction of new technologies, and while the ongoing digital transformation is improving our daily life, is also creating new unprecedented challenges. The development of new tools and platforms, their high diffusion and immediate accessibility, shape and transform the world we know.

With the information era and the consequentially over-production of information, it has become essential to properly select and assess the quality of the information we consume each day. The risk of consuming untrue information that aims to polarize our opinion is high and increasing. In this context, one main challenge concerns the development of tools that can protect the end user from the deluge of information, by helping her/him in the assessment of the truthfulness and the allegedly quality of contents. The challenge includes the conception of new business models that can promote the creation of high-quality content and prevent the diffusion of click-baits and fake pieces of information by leveraging new technologies. In this context, it is important also to evaluate the role of automated behavior as an enabling factor in the process of information dissemination.

In the last decade, thanks to the rapid diffusion of mobile technology, separating offline from the online experience has become more and more difficult. Such problems as bullying, from being limited in the context of school or work, recently it has expanded to digital space potentially losing its temporal and spatial limitation. In this context, a socially relevant challenge is to tackle the problem of cyber-bullying, with the goal of limiting or preventing stressful experiences, especially to young adults and teenagers. Moreover, leveraging the knowledge and the tools developed for the analysis of lurking behaviors could be interesting to analyze and understand bystanders and devise methods to increase their participation and thus preventing bullying acts.

In the past few years, the growing computational power of modern GPUs and the availability of large datasets have allowed the successful application of neural networks, characterized by a high number of layers, to Euclidean data. While deep learning models have proven to be an effective and powerful tool when dealing with data such as speech, images, or video, recently there has been a growing interest in trying to apply deep learning on non-Euclidean data. In this context, the development of emerging techniques attempting to generalize deep neural models to domains such as graphs and manifolds, with the goal of extending the application of deep learning models, represents a challenging line of research.

Bibliography

[1] I Pavlov, "Conditioned reflexes.", 1927.

[2] E. L. Thorndike, *Animal intelligence.* MacMillan, Princeton, NJ, 1898.

[3] E. Thorndike, *The fundamentals of learning new york: Teachers college, columbia university*, 1932.

[4] J. B. Watson, "Psychology as the behaviorist views it.", *Psychological review*, vol. 20, no. 2, p. 158, 1913.

[5] J. B. Watson and R. Rayner, "Conditioned emotional reactions.", *Journal of experimental psychology*, vol. 3, no. 1, p. 1, 1920.

[6] B. Skinner, "The behavior of organisms: An experimental analysis.", *New York*, 1938.

[7] B. F. Skinner, *About behaviorism.* Random House, New York, 1974.

[8] S. Wasserman and K. Faust, *Social Networks Analysis: Methods and Applications.* Cambridge University Press, 1994.

[9] G. Sabidussi, "The centrality index of a graph", *Psychometrika*, vol. 31, pp. 581–603, 1966.

[10] L. C. Freeman, "A set of measures of centrality based on betweenness", *Sociometry*, vol. 40, pp. 35–41, 1977.

[11] U. Brandes, "A faster algorithm for betweenness centrality", *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[12] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation", *Social Networks*, vol. 30, no. 2, pp. 136–145, 2008.

[13] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks", *Physical Review E*, vol. 69, 026113, 2004.

[14] K. Stephenson and M. Zelen, "Rethinking centrality: Methods and applications", *Social Networks*, vol. 11, pp. 1–37, 1989.

[15] L. C. Freeman, "Centrality in social networks conceptual clarification", *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.

[16] J. R. Seeley, "The net of reciprocal influence: a problem in treating sociometric data", *Canadian Journal of Psychology*, vol. 3, pp. 234–240, 1949.

[17] L. Katz, "A new status index derived from sociometric analysis", *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[18] P. Bonacich, "Factoring and weighing approaches to status scores and clique identification", *Journal of Mathematical Sociology*, vol. 2, pp. 113–120, 1972.

[19] P. Bonacich and P. Lloyd, "Eigenvector-like measures of centrality for asymmetric relations", *Social Networks*, vol. 23, pp. 191–201, 2001.

[20] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine", *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[21] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets.* Cambridge University Press, 2011.

[22] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra.* SIAM, 2000.

[23] A. N. Langville and C. D. Meyer, "Deeper inside PageRank", *Internet Mathematics*, vol. 1, no. 3, pp. 335–400, 2005.

[24] L. Pretto, "A theoretical analysis of PageRank", in *Proc. of the Int. Symposium on String Processing and Information Retrieval*, 2002, pp. 131–144.

[25] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment", in *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1998, pp. 668–677.

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

[27] R. Lempel and S. Moran, "The stochastic approach for link-structure analysis (SALSA) and the TKC effect", *Computer Networks*, vol. 33, no. 1-6, pp. 387–401, 2000.

[28] K. Bharat and M. R. Henzinger, "Improved algorithms for topic distillation in hyperlinked environments", in *Proc. of the ACM Conf. on Research and Development in Information Retrieval (SIGIR)*, 1998, pp. 104–111.

[29] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. M. Kleinberg, "Automatic resource compilation by analyzing hyperlink structure and associated text", in *Proc. of the ACM Conf. on World Wide Web (WWW)*, 1998, pp. 65–74.

[30] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas, "Finding authorities and hubs from link structures on the World Wide Web", in *Proc. of the ACM Conf. on World Wide Web (WWW)*, 2001, pp. 415–429.

[31] G. Jeh and J. Widom, "SimRank: a measure of structural-context similarity", in *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002, pp. 538–543.

[32] D. Fogaras and B. Racz, "Scaling link-based similarity search", in *Proc. ACM Conf. on World Wide Web (WWW)*, 2005, pp. 641–650.

[33] D. Lizorkin, P. Velikhov, M. N. Grinev, and D. Turdakov, “Accuracy estimate and optimization techniques for SimRank computation”, *Proc. of the VLDB Endowment (PVLDB)*, vol. 1, no. 1, pp. 422–433, 2008.

[34] I. Antonellis, H. Garcia-Molina, and C. Chang, “SimRank++: query rewriting through link analysis of the click graph”, *Proc. of the VLDB Endowment (PVLDB)*, vol. 1, no. 1, pp. 408–421, 2008.

[35] W. Yu, X. Lin, and W. Zhang, “Towards efficient SimRank computation on large networks”, in *Proc. Int. Conf. on Data Engineering (ICDE)*, 2013, pp. 601–612.

[36] L. Du, C. Li, H. Chen, L. Tan, and Y. Zhang, “Probabilistic SimRank computation over uncertain graphs”, *Information Sciences*, vol. 295, pp. 521–535, 2015.

[37] R. Weiss, B. Vélez, and M. A. Sheldon, “Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering”, in *Proc. ACM Conf. on Hypertext*, 1996, pp. 180–193.

[38] A. Broder, S. Glassman, M. Manasse, and G. Zweig, “Syntactic clustering of the web”, in *Proc. ACM Conf. on World Wide Web (WWW)*, 1997.

[39] L. Egghe and R. Rousseau, *Introduction to Informetrics*. Elsevier Science Publishers. Amsterdam, The Netherlands, 1990.

[40] R. R. Larson, “Bibliometrics of the world wide web: An exploratory analysis of the intellectual structure of cyberspace”, in *Proc. Annual Meeting American Society for Information Science*, vol. 33, 1996, pp. 71–78.

[41] J. Pitkow and P. Pirolli, “Life, death and lawfulness on the electronic frontier”, in *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, 1997.

[42] M. Perkowitz and O. Etzioni, “Adaptive web sites: an AI challenge”, in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1997.

[43] M. Perkowitz and O. Etzioni, “Towards adaptive web sites: Conceptual framework and case study”, *Computer Networks*, vol. 31, no. 11-16, pp. 1245–1258, 1999.

[44] J. Weng, E. P. Lim, J. Jiang, and Q. He, “TwitterRank: Finding Topic-Sensitive Influential Twitterers”, in *Proc. ACM Conf. on Web Search and Web Data Mining (WSDM)*, 2010, pp. 261–270.

[45] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet Allocation”, *Journal of Machine Learning Research*, vol. 3, no. 4–5, pp. 993–1022, 2003.

[46] Z. Gyöngyi, H. Garcia-Molina, and J. O. Pedersen, “Combating Web Spam with TrustRank”, in *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, 2004, pp. 576–587.

[47] Y. Sun and J. Han, *Mining Heterogeneous Information Networks: Principles and Methodologies*, ser. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, 2012.

[48] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "ObjectRank: Authority-Based Keyword Search in Databases", in *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, 2004, pp. 564–575.

[49] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma, "Object-level ranking: bringing order to Web objects", in *Proc. ACM Conf. on World Wide Web (WWW)*, 2005, pp. 567–574.

[50] S. D. Gollapalli, P. Mitra, and C. L. Giles, "Ranking authors in digital libraries", in *Proc. Int. Joint Conf. on Digital Libraries (JCDL)*, 2011, pp. 251–254.

[51] M. Zhang, S. Feng, J. Tang, B. A. Ojokoh, and G. Liu, "Co-Ranking Multiple Entities in a Heterogeneous Network: Integrating Temporal Factor and Users' Bookmarks", in *Proc. Int. Conf. on Asian Digital Libraries (ICADL)*, 2011, pp. 202–211.

[52] H. Deng, J. Han, M. R. Lyu, and I. King, "Modeling and exploiting heterogeneous bibliographic networks for expertise ranking", in *Proc. Int. Joint Conf. on Digital Libraries (JCDL)*, 2012, pp. 71–80.

[53] D. Liben-Nowell and J. M. Kleinberg, "The link-prediction problem for social networks", *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[54] D. A. Davis, R. Lichtenwalter, and N. V. Chawla, "Multi-relational link prediction in heterogeneous information networks", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2011, pp. 281–288.

[55] S. E. Helou, C. Salzmann, and D. Gillet, "The 3A Personalized, Contextual and Relation-based Recommender System", *J. UCS*, vol. 16, no. 16, pp. 2179–2195, 2010.

[56] S. Lee, S. Song, M. Kahng, D. Lee, and S. Lee, "Random walk based entity ranking on graph for multidimensional recommendation", in *Proc. ACM Conf. on Recommender Systems (RecSys)*, 2011, pp. 93–100.

[57] V. Vasuki, N. Natarajan, Z. Lu, B. Savas, and I. S. Dhillon, "Scalable Affiliation Recommendation using Auxiliary Networks", *ACM Trans. Intelligent Systems and Technology*, vol. 3, no. 1, p. 3, 2011.

[58] R. Interdonato and A. Tagarelli, "Multi-relational PageRank for Tree Structure Sense Ranking", *World Wide Web Journal*, 2014.

[59] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks", *Machine Learning*, vol. 81, no. 1, pp. 53–67, 2010.

[60] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks", *Procs. of the VLDB Endowment (PVLDB)*, vol. 4, no. 11, pp. 992–1003, 2011.

[61] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu, "RankClus: integrating clustering with ranking for heterogeneous information network analysis", in *Proc. Int. Conf. on Extending Database Technology (EDBT)*, 2009, pp. 565–576.

[62] Y. Sun, Y. Yu, and J. Han, "Ranking-based clustering of heterogeneous information networks with star network schema", in *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2009, pp. 797–806.

[63] W. Jiang, G. Wang, M. Z. A. Bhuiyan, and J. Wu, "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges", *ACM Comput. Surv.*, vol. 49, no. 1, 10:1–10:35, 2016.

[64] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks", *ACM Comput. Surv.*, vol. 45, no. 4, 47:1–47:33, 2013.

[65] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank", in *Proc. of Conf. on Very Large Data Bases (VLDB)*, 2004, pp. 576–587.

[66] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks", in *Proc. of Conf. on World Wide Web (WWW)*, 2003, pp. 640–651.

[67] R. Aringhieri, E. Damiani, S. D. C. Di Vimercati, S. Paraboschi, and P. Samarati, "Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems", *J. Am. Soc. Inf. Sci. Technol.*, vol. 57, no. 4, pp. 528–537, 2006.

[68] P. Massa and P. Avesani, "Trust metrics on controversial users: Balancing between tyranny of the majority", *Int. J. Semantic Web Inf. Syst.*, vol. 3, no. 1, pp. 39–64, 2007.

[69] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on epinions.com community", in *Proc. of National Conf. on Artificial Intelligence and Conf. on Innovative Applications of Artificial Intelligence*, 2005, pp. 121–126.

[70] P. Victor, C. Cornelis, M. D. Cock, and A. Teredesai, "A comparative analysis of trust-enhanced recommenders for controversial items", in *Proc. AAAI Conf. on Weblogs and Social Media (ICWSM)*, 2009.

[71] P. Zicari, R. Interdonato, D. Perna, A. Tagarelli, and S. Greco, "Controversy in trust networks", in *Proc. of Int. Conf. on Trust and Trustworthy Computing (TRUST)*, 2016, pp. 82–100.

[72] J. Golbeck, B. Parsia, and J. A. Hendler, "Trust networks on the semantic web", in *Workshop on Cooperative Information Agents (CIA)*, 2003, pp. 238–249.

[73] W. Jiang, G. Wang, and J. Wu, "Generating trusted graphs for trust evaluation in online social networks.", *Future Generation Comp. Syst.*, pp. 48–58, 2014.

[74] S. Nepal, W. Sherchan, and C. Paris, "Strust: A trust model for social networks", in *Proc. of IEEE Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2011, pp. 841–846.

[75] S. Adali, R. Escriva, M. K. Goldberg, M. Hayvanovych, M. Magdon-Ismail, B. K. Szymanski, W. A. Wallace, and G. T. Williams, "Measuring behavioral trust in social networks", in *Proc. of IEEE Conf. on Intelligence and Security Informatics (ISI)*, 2010, pp. 150–152.

[76] J. A. Golbeck, "Computing and applying trust in web-based social networks", PhD thesis, College Park, MD, USA, 2005.

[77] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web", Stanford, USA, Tech. Rep., 1999.

[78] S. Hamdi, A. Bouzeghoub, A. L. Gançarski, and S. B. Yahia, "Trust inference computation for online social networks.", in *Proc. of IEEE Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom) / IEEE Symposium on Parallel and Distributed Processing with Applications (ISPA) / IEEE Conf. on Ubiquitous Computing and Communications (IUCC)*, 2013, pp. 210–217.

[79] P. Victor, C. Cornelis, M. D. Cock, and A. Teredesai, "Trust- and distrust-based recommendations for controversial reviews", *IEEE Intelligent Systems*, vol. 26, no. 1, pp. 48–55, 2011.

[80] C. Haydar, A. Roussanaly, and A. Boyer, "Local trust versus global trust networks in subjective logic", in *Proc. IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 2013, pp. 29–36.

[81] M. Tang, Y. Xu, J. Liu, Z. Zheng, and X. F. Liu, "Combining global and local trust for service recommendation", in *Proc. of IEEE Conf. on Web Services (ICWS)*, 2014, pp. 305–312.

[82] M. E. Dickison, M. Magnani, and L. Rossi, *Multilayer social networks.* Cambridge University Press, 2016.

[83] S. Boccaletti, G. Bianconi, R. Criado, C. I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin, "The structure and dynamics of multilayer networks", *Physics Reports*, vol. 544, pp. 1–122, 2014.

[84] M. Kivela, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Mutilayer networks", *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.

[85] N. Sun, P. P.-L. Rau, and L. Ma, "Understanding lurkers in online communities: A literature review", *Computers in Human Behavior*, vol. 38, pp. 110–117, 2014.

[86] N. Edelmann, "Reviewing the definitions of "lurkers" and some implications for online research", *Cyberpsychology, Behavior, and Social Networking*, vol. 16, no. 9, pp. 645–649, 2013.

[87] J. Lave and E. Wenger, *Situated Learning: Legitimate Peripheral Participation.* Cambridge University Press, 1991.

[88] N. Kahnwald and T. Köhler, "Microlearning in virtual communities of practice? an explorative analysis of changing information behaviour", in *Proc. Microlearning Conf.*, 2006, pp. 157–172.

[89] A. Ardichvili, "Learning and knowledge sharing in virtual communities of practice: motivators, barriers, and enablers", *Advances in Developing Human Resources*, vol. 10, pp. 541–554, 2008.

[90] J. T. Child and S. C. Starcher, "Fuzzy Facebook privacy boundaries: Exploring mediated lurking, vague-booking, and Facebook privacy management", *Computers in Human Behavior*, vol. 54, pp. 483–490, 2016.

[91] B. Osatuyi, "Is lurking an anxiety-masking strategy on social media sites? the effects of lurking and computer anxiety on explaining information privacy concern on social media platforms", *Computers in Human Behavior*, vol. 49, pp. 324–332, 2015.

[92] J. Cranefield, P. Yoong, and S. L. Huff, "Beyond Lurking: The Invisible Follower-Feeder In An Online Community Ecosystem", in *Proc. Pacific Asia Conf. on Information Systems (PACIS)*, 2011, p. 50.

[93] "Boundary spanning", in *Encyclopedia of Social Network Analysis and Mining*, R. Alhajj and J. Rokne, Eds., 2014, p. 82.

[94] H. Tong, S. Papadimitriou, C. Faloutsos, P. S. Yu, and T. Eliassi-Rad, "Gateway finder in large graphs: Problem definitions and fast solutions", *Inf. Retr.*, vol. 15, no. 3-4, pp. 391–411, 2012.

[95] A. J. Morales, J. C Losada, and R. M. Benito, "Users structure and behavior on an online social network during a political protest", *Physica A*, vol. 391, no. 21, pp. 5244–5253, 2012.

[96] E. Zhong, W. Fan, and Q. Yang, "User Behavior Learning and Transfer in Composite Social Networks", *ACM Trans. Knowl. Discov. D.*, vol. 8, no. 1, art. 6, 2014.

[97] A. Tagarelli and R. Interdonato, "Lurking in social networks: Topology-based analysis and ranking methods", *Social Netw. Analys. Mining*, vol. 4, no. 230, p. 27, 2014.

[98] R. Interdonato, C. Pulice, and A. Tagarelli., "Community-based delurking in social networks", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2016.

[99] W. Chen, L. V. S. Lakshmanan, and C. Castillo, *Information and Influence Propagation in Social Networks*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.

[100] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Steering user behavior with badges", in *Proc. ACM Conf. on World Wide Web (WWW)*, 2013.

[101] Z. Pan, Y. Lu, and S. Gupta, "How heterogeneous community engage newcomers? The effect of community diversity on newcomers' perception of inclusion: An empirical study in social media service", *Computers in Human Behavior*, vol. 39, pp. 100–111, 2014.

[102] J. Imlawi and D. G. Gregg, "Engagement in online social networks: The impact of self-disclosure and humor", *Int. J. Hum. Comput. Interaction*, vol. 30, no. 2, pp. 106–125, 2014.

[103] R. Farzan and P. Brusilovsky, "Encouraging user participation in a course recommender system: An impact on user behavior", *Computers in Human Behavior*, vol. 27, no. 1, pp. 276–284, 2011.

[104] A. Halu, R. J. Mondragon, P. Panzarasa, and G. Bianconi, "Multiplex pagerank", *PLOS One*, vol. 8, no. 10, e78293, 2013.

[105] A. Sole, M. D. Domenico, S. Gomez, and A. Arenas, "Centrality rankings in multiplex networks", pp. 149–155, 2014.

[106] M. D. Domenico, A. Sole, S. Gomez, and A. Arenas, "Navigability of interconnected networks under random failures", *PNAS*, vol. 111, no. 23, pp. 8351–8356, 2014.

[107] J. Kim and J. Lee, "Community detection in multi-layer graphs: A survey", *SIGMOD Record*, vol. 44, no. 3, pp. 37–48, 2015.

[108] C. W. Loe and H. J. Jensen, "Comparison of communities detection algorithms for multiplex", *Physica A*, vol. 431, pp. 29–45, 2015.

[109] M. Fazeen, R. Dantu, and P. Guturu, "Identification of leaders, lurkers, associates and spammers in a social network: Context-dependent and context-independent approaches", *Social Netw. Analys. Mining*, vol. 1, no. 3, pp. 241–254, 2011.

[110] A. Tagarelli and R. Interdonato, ""Who's out there?": Identifying and Ranking Lurkers in Social Networks", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2013, pp. 215–222.

[111] R. Interdonato and A. Tagarelli, "Time-aware analysis and ranking of lurkers in social networks", *Social Netw. Analys. Mining*, vol. 5, no. 1, 46:1–46:23, 2015.

[112] R. Interdonato and A. Tagarelli, "To trust or not to trust lurkers?: Evaluation of lurking and trustworthiness in ranking problems", in *Proc. Int. School and Conf. on Network Science (NetSciX)*, 2016.

[113] P. Bródka, K. Skibicki, P. Kazienko, and K. Musial, "A degree centrality in multi-layered social network", in *Proc. Conf. on Computational Aspects of Social Networks (CASoN)*, 2011, pp. 237–242.

[114] S. Tavassoli and K. A. Zweig, "Most central or least central? how much modeling decisions influence a node's centrality ranking in multiplex networks", *CoRR*, vol. abs/1606.05468, 2016.

[115] T. Chakraborty and R. Narayanam, "Cross-layer betweenness centrality in multiplex networks with applications", in *Proc. Int. Conf. on Data Engineering (ICDE)*, 2016, pp. 397–408.

[116] F. Battiston, V. Nicosia, and V. Latora, "Efficient exploration of multiplex networks", *New Journal of Physics*, vol. 18, no. 4, p. 043 035, 2016.

[117] Z. Kuncheva and G. Montana, "Community detection in multiplex networks using locally adaptive random walks", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2015, pp. 1308–1315.

[118] R. Interdonato and A. Tagarelli, "Multi-relational PageRank for tree structure sense ranking", *World Wide Web*, vol. 18, no. 5, pp. 1301–1329, 2015.

[119] H. Abdi, "The Kendall Rank Correlation Coefficient", in *Encyclopedia of Measurement and Statistics*, 2007.

[120] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing Top k Lists", *SIAM Journal on Discrete Mathematics*, vol. 17, no. 1, pp. 134–160, 2003.

[121] G. Silvestri, J. Yang, A. Bozzon, and A. Tagarelli, "Linking accounts across social networks: The case of stackoverflow, github and twitter", in *Procs. Int. Workshop on Knowledge Discovery on the Web*, 2015, pp. 41–52.

[122] M. D. Domenico, A. Lima, P. Mougel, and M. Musolesi, "The anatomy of a scientific rumor", *Scientific Reports*, vol. 3, no. 2980, 2013.

[123] A. Mislove, H. S. Koppula, P. K. Gummadi, P. Druschel, and B. Bhattacharjee, "Growth of the Flickr Social Network", in *Proc. of the First Workshop on Online Social Networks (WOSN)*, 2008, pp. 25–30.

[124] M. Cha, A. Mislove, and K. P. Gummadi, "A Measurement-driven Analysis of Information Propagation in the Flickr Social Network", in *Proc. ACM Conf. on World Wide Web (WWW)*, 2009, pp. 721–730.

[125] M. E. J. Newman, "Assortative mixing in networks", *Phys. Rev. Lett.*, vol. 89, no. 208701, 2002.

[126] M. Piraveenan, M. Prokopenko, and A. Y. Zomaya, "Local assortativeness in scale-free networks", *EPL*, vol. 84, no. 28002, 2008.

[127] L. Bogolubsky, P. Dvurechensky, A. Gasnikov, G. Gusev, Y. Nesterov, A. M. Raigorodskii, A. Tikhonov, and M. Zhukovskii, "Learning Supervised PageRank with Gradient-Based and Gradient-Free Optimization Methods", in *Proc. Annual Conf. on Neural Information Processing Systems (NIPS)*, 2016, pp. 4907–4915.

[128] T. Liu, *Learning to Rank for Information Retrieval.* Springer, 2011.

[129] S. Chakrabarti, "Learning to Rank in Vector Spaces and Social Networks", *Internet Mathematics*, vol. 4, no. 1–3, pp. 267–298, 2007.

[130] R. Busa-Fekete, G. Szarvas, T. Elteto, and B. Kégl, "An apple-to-apple comparison of learning-to-rank algorithms in terms of normalized discounted cumulative gain", in *Proc. ECAI Work. on Preference Learning: Problems and Applications in AI*, vol. 242, 2012.

[131] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender, “Learning to rank using gradient descent”, in *Proc. Int. Conf. on Machine Learning (ICML)*, 2005, pp. 89–96.

[132] D. Metzler and W. B. Croft, “Linear feature-based models for information retrieval”, *Inf. Retr.*, vol. 10, no. 3, pp. 257–274, 2007.

[133] J. Xu and H. Li, “Adarank: A boosting algorithm for information retrieval”, in *Proc. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, 2007, pp. 391–398.

[134] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting”, in *Proc. European Conf. on Computational Learning Theory (EuroCOLT)*, 1995, pp. 23–37.

[135] Q. Wu, C. J. C. Burges, K. M. Svore, and J. Gao, “Adapting boosting for information retrieval measures”, *Inf. Retr.*, vol. 13, no. 3, pp. 254–270, 2010.

[136] C. J. C. Burges, R. Ragno, and Q. V. Le, “Learning to rank with nonsmooth cost functions”, in *Proc. Conf. on Neural Information Processing Systems (NIPS)*, 2006, pp. 193–200.

[137] J. H. Friedman, “Greedy function approximation: A gradient boosting machine”, *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

[138] C. J. Burges, “From ranknet to lambdarank to lambdamart: An overview”, Tech. Rep. MSR-TR-2010-82, 2010, p. 81.

[139] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi, “Measuring user influence in twitter: The million follower fallacy”, in *Proc. Int. Conf. on Weblogs and Social Media (ICWSM)*, 2010.

[140] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna, “Poultry markets: On the underground economy of twitter followers”, in *Proc. of ACM Workshop on Online Social Networks (WOSN)*, 2012, pp. 1–6.

[141] J. Ratkiewicz, M. Conover, M. R. Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer, “Truthy: Mapping the spread of astroturf in microblog streams”, in *Proc. ACM Conf. on World Wide Web (WWW)*, 2011, pp. 249–252.

[142] A. Bessi and E. Ferrara, “Social bots distort the 2016 U.S. presidential election online discussion”, *First Monday*, vol. 21, no. 11, 2016.

[143] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race”, in *Proc. ACM Conf. on World Wide Web (WWW)*, 2017, pp. 963–972.

[144] E. Ferrara, O. Varol, C. A. Davis, F. Menczer, and A. Flammini, “The rise of social bots”, *Commun. ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[145] K. Lee, B. D. Eoff, and J. Caverlee, “Seven months with the devils: A long-term study of content polluters on twitter”, in *Proc. Int. Conf. on Weblogs and Social Media*, 2011.

[146] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, “Design and analysis of a social botnet”, *Computer Networks*, vol. 57, no. 2, pp. 556–578, 2013.

[147] S. Hegelich and D. Janetzko, “Are social bots on twitter political actors? empirical evidence from a ukrainian social botnet”, in *Proc. Int. Conf. on Weblogs and Social Media (ICWSM)*, 2016, pp. 579–582.

[148] P. N. Howard and B. Kollanyi, “Bots, #strongerin, and #brexit: Computational propaganda during the UK-EU referendum”, *Social Science Research Network (SSRN) Electronic Journal*, 2016.

[149] M. Forelle, P. Howard, A. Monroy-Hernández, and S. Savage, “Political bots and the manipulation of public opinion in venezuela”, *Social Science Research Network (SSRN) Electronic Journal*, 2015.

[150] J. Ratkiewicz, M. Conover, M. R. Meiss, B. Gonçalves, A. Flammini, and F. Menczer, “Detecting and tracking political abuse in social media”, in *Proc. Int. Conf. on Weblogs and Social Media (ICWSM)*, 2011.

[151] P. T. Metaxas and E. Mustafaraj, “Social media and the elections”, *Science*, vol. 338, no. 6106, pp. 472–473, 2012.

[152] L. M. Aiello, M. Deplano, R. Schifanella, and G. Ruffo, “People are strange when you’re a stranger: Impact and influence of bots on social networks”, in *Proc. Int. Conf. on Weblogs and Social Media (ICWSM)*, 2012.

[153] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. J. Metzger, H. Zheng, and B. Y. Zhao, “Social turing tests: Crowdsourcing sybil detection”, in *Proc. of Symp. on Network and Distributed System Security, NDSS*, 2013.

[154] J. Zhang, D. Carpenter, and M. Ko, “Online astroturfing: A theoretical perspective”, in *Proc. Americas Conf. on Information Systems (AMCIS)*, 2013.

[155] F. B. Keller, D. Schoch, S. Stier, and J. Yang, “How to manipulate social media: Analyzing political astroturfing using ground truth data from south korea”, in *Proc. Int. Conf. on Weblogs and Social Media (ICWSM)*, 2017, pp. 564–567.

[156] C. Wagner, S. Mitter, C. Körner, and M. Strohmaier, “When social bots attack: Modeling susceptibility of users in online social networks”, in *Proc. ACM Conf. on World Wide Web (WWW)*, 2012, pp. 41–48.

[157] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, “Aiding the detection of fake accounts in large scale social online services”, in *Proc. of Symp. on Networked Systems Design and Implementation, NSDI*, 2012, pp. 197–210.

[158] Y. Xie, F. Yu, Q. Ke, M. Abadi, E. Gillum, K. Vitaldevaria, J. Walter, J. Huang, and Z. M. Mao, “Innocent by association: Early recognition of legitimate users”, in *Proc. of ACM Conf. on Computer and Communications Security, (CCS)*, 2012, pp. 353–364.

[159] A. Paradise, R. Puzis, and A. Shabtai, "Anti-reconnaissance tools: Detecting targeted socialbots", *IEEE Internet Computing*, vol. 18, no. 5, pp. 11–19, 2014.

[160] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: Stopping group attacks by spotting lockstep behavior in social networks", in *Proc. ACM Conf. on World Wide Web (WWW)*, 2013, pp. 119–130.

[161] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks", in *Proc. of 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 477–488.

[162] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection", in *Proc. of Symp. on Security USENIX*, 2013, pp. 241–256.

[163] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild", *TKDD*, vol. 8, no. 1, 2:1–2:29, 2014.

[164] N. Chavoshi, H. Hamooni, and A. Mueen, "Debot: Twitter bot detection via warped correlation", in *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, 2016, pp. 817–822.

[165] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of twitter accounts: Are you a human, bot, or cyborg?", *IEEE Trans. Dependable Sec. Comput.*, vol. 9, no. 6, pp. 811–824, 2012.

[166] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization", in *Proc. Int. Conf. on Weblogs and Social Media (ICWSM)*, 2017, pp. 280–289.

[167] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "Botornot: A system to evaluate social bots", in *Proc. ACM Conf. on World Wide Web (WWW)*, 2016, pp. 273–274.

[168] J. P. Dickerson, V. Kagan, and V. S. Subrahmanian, "Using sentiment to detect bots on twitter: Are humans more opinionated than bots?", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2014, pp. 620–627.

[169] A. J. Minnich, N. Chavoshi, D. Koutra, and A. Mueen, "Botwalk: Efficient adaptive exploration of twitter bot networks", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2017, pp. 467–474.

[170] S. Sedhai and A. Sun, "Hashtag recommendation for hyperlinked tweets", in *Proc. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, 2014, pp. 831–834.

[171] A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier, "Tweetcred: Real-time credibility assessment of content on twitter", pp. 228–243, 2014.

[172] M. Surdeanu, M. Ciaramita, and H. Zaragoza, “Learning to rank answers on large online QA collections”, in *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics*, vol. 8, 2008, pp. 719–727.

[173] H. Zamani, A. Shakery, and P. Moradi, “Regression and learning to rank aggregation for user engagement evaluation”, in *Proc. ACM Recommender Systems Challenge (RecSysChallenge)*, 2014.

[174] D. Perna and A. Tagarelli, “An evaluation of learning-to-rank methods for lurking behavior analysis”, in *Proc. Int. Conf. on User Modeling, Adaptation and Personalization, UMAP*, 2017, pp. 381–382.

[175] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. J. Keogh, “Experimental comparison of representation methods and distance measures for time series data”, *Data Min. Knowl. Discov.*, vol. 26, no. 2, pp. 275–309, 2013.

[176] F. Morstatter, L. Wu, T. H Nazer, K. M Carley, and H. Liu, “A new approach to bot detection: Striking the balance between precision and recall”, in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2016, pp. 533–540.

[177] Z. Gilani, E. Kochmar, and J. Crowcroft, “Classification of twitter accounts into automated agents and human users”, in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, ACM, 2017, pp. 489–496.

[178] M. A. Hall, “Correlation-based feature selection for machine learning”, 1999.

[179] W. Chen, T. Y. Liu, Y. Lan, Z. Ma, and H. Li, “Ranking measures and loss functions in learning to rank”, in *Proc. Conf. on Neural Information Processing Systems (NIPS)*, 2009, pp. 315–323.

[180] K. Järvelin and J. Kekäläinen, “Ir evaluation methods for retrieving highly relevant documents”, in *Proc. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, 2000, pp. 41–48.

[181] B. Nonnecke and J. J. Preece, “Lurker demographics: Counting the silent”, in *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)*, 2000, pp. 73–80.

[182] J. J. Preece, B. Nonnecke, and D. Andrews, “The top five reasons for lurking: Improving community experiences for everyone”, *Computers in Human Behavior*, vol. 20, no. 2, pp. 201–223, 2004.

[183] H. Tsai and P. Pai, “Why do newcomers participate in virtual communities? An integration of self-determination and relationship management theories”, *Decision Support Systems*, vol. 57, pp. 178–187, 2014.

[184] R. Farzan, J. M. DiMicco, and B. Brownholtz, “Mobilizing lurkers with a targeted task”, in *Proc. Int. Conf. on Weblogs and Social Media (ICWSM)*, 2010.

[185] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a highly connected world.* Cambridge University Press, 2010.

[186] F. D. Malliaros and M. Vazirgiannis, "To stay or not to stay: Modeling engagement dynamics in social graphs", in *Proc. ACM Conf. on Information and Knowledge Management (CIKM)*, 2013, pp. 469–478.

[187] M. Rowe, "Mining user lifecycles from online community platforms and their application to churn prediction", in *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, 2013, pp. 637–646.

[188] R. Interdonato, C. Pulice, and A. Tagarelli, ""Got to have faith!": The DEvOTION algorithm for delurking in social networks", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2015, pp. 314–319.

[189] S. Martin, W. M. Brown, R. Klavans, and K. W. Boyack, "Openord: An open-source toolbox for large graph layout", in *Visualization and Data Analysis*, 2011.

[190] D. Perna, R. Interdonato, and A. Tagarelli, "Identifying users with alternate behaviors of lurking and active participation in multilayer social networks", *IEEE Transactions on Computational Social Systems*, 2017.

[191] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: quantifying influence on Twitter", in *Proc. ACM Conf. on Web Search and Web Data Mining (WSDM)*, 2011, pp. 65–74.

[192] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications", *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[193] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives", *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[194] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey", *Knowl.-Based Syst.*, vol. 151, pp. 78–94, 2018.

[195] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations", in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)"*, ACM, 2014, pp. 701–710.

[196] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks", in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)"*, ACM, 2016, pp. 855–864.

[197] N. Zhao, H. Zhang, M. Wang, R. Hong, and T. Chua, "Learning content-social influential features for influence analysis", *IJMIR*, vol. 5, no. 3, pp. 137–149, 2016.

[198] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding", 2017, pp. 203–209.

[199] S. Cavallari, V. W. Zheng, H. Cai, K. C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs", in *Proc. ACM Conf. on Information and Knowledge Management (CIKM)*, 2017, pp. 377–386.

[200] C. Morris, K. Kersting, and P. Mutzel, "Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs", in *2017 IEEE International Conference on Data Mining, ICDM*, 2017, pp. 327–336.

[201] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs", in *Proc. Int. Conf. on Machine Learning (ICML)*, 2016, pp. 2014–2023.

[202] S. F. Mousavi, M. Safayani, A. Mirzaei, and H. Bahonar, "Hierarchical graph embedding in vector space by graph pyramid", *Pattern Recognition*, vol. 61, pp. 245–254, 2017.

[203] T. Hofmann and J. M. Buhmann, "Multidimensional scaling and data clustering", in *Advances in Neural Information Processing Systems 7, [NIPS Conference*, 1994, pp. 459–466.

[204] Y. Han and Y. Shen, "Partially supervised graph embedding for positive unlabelled feature selection", in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2016, pp. 1548–1554.

[205] M. Yin, J. Gao, and Z. Lin, "Laplacian regularized low-rank representation and its applications", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 3, pp. 504–517, 2016.

[206] L. Tang, X. Wang, and H. Liu, "Uncoverning groups via heterogeneous interaction analysis", in *Proc. ICDM*, IEEE, 2009, pp. 503–512.

[207] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding", in *Proc. ACM Conf. on World Wide Web (WWW)*, 2015, pp. 1067–1077.

[208] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", *arXiv preprint arXiv:1301.3781*, 2013.

[209] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering.", in *AAAI*, 2014, pp. 1293–1299.

[210] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations.", in *AAAI*, 2016, pp. 1145–1152.

[211] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding", in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)"*, ACM, 2016, pp. 1225–1234.

[212] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs", in *International conference on machine learning*, 2016, pp. 2014–2023.

[213] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks", *arXiv preprint arXiv:1609.02907*, 2016.

[214] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model", *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[215] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs", in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[216] W. Liu, P. Chen, S. Yeung, T. Suzumura, and L. Chen, "Principled multilayer network embedding", in *IEEE International Conference on Data Mining Workshops, ICDM.*

[217] H. Zhang, L. Qiu, L. Yi, and Y. Song, "Scalable multiplex network embedding", in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2018, pp. 3082–3088.

[218] R. Matsuno and T. Murata, "MELL: effective embedding method for multiplex networks", in *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, 2018, pp. 1261–1268.

[219] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems*, 2013, pp. 3111–3119.

[220] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations", in *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, 2013, pp. 746–751.

[221] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*, 2013, pp. 3111–3119.

[222] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", in *Int. Conf. on Learning Representations, ICLR*, 2013.

[223] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics", *Journal of Machine Learning Research*, vol. 13, pp. 307–361, 2012.

[224] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models", in *Proc. Int. Conf. on Machine Learning (ICML)*, 2012.

[225] Z. Kuncheva and G. Montana, "Community detection in multiplex networks using locally adaptive random walks", in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, ACM, 2015, pp. 1308–1315.

[226] L. Rossi and M. Magnani, "Towards effective visual analytics on multiplex and multilayer networks", *Chaos, Solitons and Fractals*, vol. 72, pp. 68–76, 2015.

[227] B. L. Chen, D. H. Hall, and D. B. Chklovskii, “Wiring optimization can relate neuronal structure and function”, *Proc. of the National Academy of Sciences*, vol. 103, no. 12, pp. 4723–4728, 2006.

[228] M. De Domenico, M. A. Porter, and A. Arenas, “Muxviz: A tool for multilayer analysis and visualization of networks”, *Journal of Complex Networks*, vol. 3, no. 2, pp. 159–176, 2015.

[229] M. De Domenico, A. Lancichinetti, A. Arenas, and M. Rosvall, “Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems”, *Physical Review X*, vol. 5, no. 1, p. 011 027, 2015.

[230] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression”, *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[231] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression”, *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[232] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[233] L. Breiman, “Random forests”, *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[234] F. Rosenblatt, “Principles of neurodynamics. perceptrons and the theory of brain mechanisms”, CORNELL AERONAUTICAL LAB INC BUFFALO NY, Tech. Rep., 1961.

[235] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation”, California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[236] S. Russell, P. Norvig, and A. Intelligence, “A modern approach”, *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, no. 27, pp. 79–80, 1995.