UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Dottorato di Ricerca in

Ingegneria dei Sistemi ed Informatica

CICLO

XXVII

NOVEL RANKING PROBLEMS IN INFORMATION NETWORKS

Settore Scientifico Disciplinare ING-INF/05

Coordinatore: Ch.mo Prof. Sergio Greco

Firma _____

Supervisore: Andrea Tagarelli

Firma _____

Dottorando: Dott. Roberto Interdonato

Firma _____

Contents

1

Introduction

Any set of informational objects (e.g., data objects, individual agents, groups or components) interconnected with each other to form large sophisticated networks can be defined, without loss of generality, as an *information network*. Therefore, in this type of networks, the basic units being connected are pieces of information, and links join pieces of information that are related to each other in some fashion [49]. It is not difficult to enumerate examples of well-known information networks: social networks, the World Wide Web, computer networks, research publication networks, biological and chemical systems, neural networks, highway networks, and many others. Given the fact that information networks are pervasive and can be considered as the building blocks of modern information infrastructures, they gain increasing attention from researchers in computer science, social science, physics, economics, biology, and other disciplines.

The analysis of information networks is largely based on principles and algorithms from graph theory, such as short paths and giant components. The success of studies about information networks is certainly tied to the growth of the World Wide Web, which is probably the most remarkable example of such kind of networks. Research on information networks has a long history. The hypertextual structure of the Web itself has in fact a relevant precursor in the concept of *citation network* built among scholarly books and articles. The structure of a citation network naturally forms a directed graph, with nodes representing books and articles, connected by directed edges representing citations from one work to another (similar structures can be found in networks which represent connections among patents or legal decisions). A main difference between the Web and older kind of networks is in the role of the time dimension. A citation which comes from an article (or a book, a patent or a legal decision) always points to an "older" object, for the simple reason that an article is a static object which is published at a specific point in time, and can only refer to previous works. Conversely, Web pages are dynamical objects, usually updated many times during their life, leading to a structure which still contains directed links, but not necessarily from newer to older pages.

Another information network model preceding the World Wide Web can be found in the cross-references within a printed encyclopedia: cross-referencing links define explicit relations among the articles, allowing the browsing of an encyclopedia across different topics. By analyzing this cross-reference model, it can be observed that relatively distant concepts are connected by short paths, showing that the *six degrees of separation* phenomenon, often referred to in social networks (where similarly short paths link apparently distant people), also holds on this domain. This serendipitous browsing among chains of related encyclopedia articles is closely related to the way our mind rapidly associates different ideas (*stream-of-consciousness*), a model which has also been formalized through an information network, called *semantic network*. In this kind of network nodes represent concepts, and edges represent some logical or perceived relationship between two concepts.

Back to the prominent example of the World Wide Web, in the last decade a lot of radical changes happened, which made the Web evolve to an even more complex and dynamic network. This has changed so much the user perspective that the definition *Web 2.0* [133] had to be created to describe this new attitude. The major changes can be found in the growth of collective created and maintained shared content, the growth of services to store and manage personal data online, and the growth of services which are founded on on-line connections among people and not just among documents. The social aspect of the Web is constantly gaining importance, so that in most cases online web sites and services become more appealing and generally more useful as their audience grows larger. When the audience also contributes to the production of a valuable collective informational artifact (e.g., collaborative authoring of Wikipedia), we refer to a *wisdom of crowd* process, a phenomenon that explains the collective information residing in a large group. By contrast, the large amount of personally authored content produces a *long tail* effect, where there is a small amount of extremely popular content, and a long tail of content which is enjoyed by a relatively small audience. Another phenomenon which gained importance with the Web 2.0 paradigm is the *social feedback* mechanisms, which generated interesting studies concerning *reputation and trust systems*, based on a network of judgments/ratings among users (or between users and objects/services).

In the last years, research in network analysis has spanned a variety of information retrieval and knowledge discovery tasks, such as community discovery, link prediction, keyword search, expert finding, and information diffusion related tasks. Particular attention has been devoted to the understanding of the behavior of nodes, and underlying relations, that are relevant to a specific task due to their "centrality" in the network. The latter often implies a ranking task over the nodes in the network, i.e., objects in the network should be sorted according to importance, popularity, influence, authority, relevance, similarity, and proximity, by utilizing link information in the network. With the growth of the Web, which scrambled users' roles so that everyone is an author and a searcher at the same time, the ranking problems (together with

all kinds of problems surrounding information retrieval) exploded in scale and complexity. The abundance and diversity of information which can be extracted from the Web takes the typical challenges of ranking problems to new extremes: e.g., a ranking method should be able to recognize to what extent a certain document is pertinent to a desired topic and at the same time what is the level of expertise (or trust) of the document source. This scenario favors an increasing demand for new generation ranking systems that are capable of effectively taking into account the heterogeneity in the type of information, producing personalized ranking based on users' profiles and taking into account the diversity of interactions among users.

Contributions

In this thesis we address research topics that are centered around the problem of ranking in information networks. Specifically, three main research topics can be distinguished:

Ranking problems in semantic networks over semi-structured data. Labeled tree-data (e.g., XML trees) are a convenient model to describe real-life objects and their structural relationships, since they allow the representation of semantic-rich information through a natural hierarchical organization. However, the flexibility of this model is at the same time the key factor for its success in the field of knowledge management and exchange, and the source of lexical ambiguity issues: the semi-structured data model easily leads to tree data which can be diverse across different information sources and often also within the same source. Thus, coupling the syntactic information within labeled tree data with appropriate semantics, with the goal to obtain a contextual ranking of all possible meanings of the tags in a tree data, is a challenging task. The network of meanings underlying the structural constituents of tree data can conveniently be represented as a labeled (weighted) graph, therefore graph-based ranking methods are natural candidates to solve the structural sense ranking problem. In this research line, the problem has been addressed defining a multi-relational PageRank-based approach, applied to a heterogeneous information network built upon the different structural relations among the tags in the tree. Our intuition is that a multi-relational ranking method should in principle be able to better leverage the semantics of annotations in tree data that are structurally related at different levels, propagating the importance scores through different multi-typed relations. *The research line on ranking in semantic networks defined over semi-structured data is the focus of Chapter 3.*

Ranking algorithms for package recommendation systems. In the last years the advent of e-commerce and web search applications of increasing complexity, together with the fundamental need to integrate these kinds of applications with social media networks in order to enhance the degree of

personalization, made recommender systems one of the most pervasive mechanisms on the entire Web. A key challenge in this scenario is to devise systems that are capable of producing enhanced quality recommendations which take into account the heterogeneity in the type of information to personalize and deliver to the users. Moreover, an emerging trend, which showed to be a key to successfully face a number of applications, is the design of methods capable of recommending packages instead of single items. The problem is challenging due to a variety of critical aspects, including context-based and user-provided constraints for the items constituting a package, but also the high sparsity and limited accessibility of the primary data used to solve the problem. Most existing works on the topic have focused on a specific application domain (e.g., travel package recommendation), thus often providing ad-hoc solutions that cannot be adapted to other domains. In this research line, a versatile package recommendation approach has been defined that is substantially independent of the peculiarities of a particular application domain. The proposed framework is capable to exploit prior knowledge on package models (i.e., how the different item types should be grouped to form packages which meet the users' needs) in order to learn a package set that represents the input of a recommendation stage. This is finally accomplished by performing a PageRank-style method personalized w.r.t. the target user's preferences. *The research line on ranking algorithms for package recommendation is the focus of Chapter 4.*

Identification and ranking of lurkers in social networks. Despite it has been proved that the major part of members in an online community can be considered lurkers (e.g., silent users, who take benefits from others' contributions without producing anything on their own), the problem has never been formally addressed in previous literature in computer science. This research topic is characterized by the definition of computational methods for lurkers identification and ranking in social networks, which are based solely on the network's topology. Both quantitative and qualitative experimental result demonstrated the effectiveness of our approach in identifying and ranking lurkers in an online social network, unveiling lurking cases that are intuitive yet non-trivial. An extension that also integrates time-aware features (e.g., amount and freshness of users' interactions) has been also proposed. Due to the fact that social networks are by nature dynamic environments, a main part of the study is based on the study of lurking behaviors over time, analyzing their role in the network from different perspectives. In this research line, the problem of identifying and ranking silent nodes was also studied on application domains other than the online social networks, namely *research collaboration networks* and *social trust contexts*. In the contexts of research collaboration networks, the *vicarious learning* problem has been addressed, which consists in the study of expert-apprentice or advisor-advisee relationships focusing on an apprentice point of view. The study includes the definiton of methods for identification and analysis of apprentice relations and in particular of vicarious learners. The study of *lurking in social trust contexts* is

focused on the challenge to model the understanding of lurkers from a perspective of social trust computing. The study is based on a comparative evaluation of lurker ranking methods against classic approaches to trust/distrust ranking, and on a first attempt to combine the two families of methods. Results indicate that lurkers should not be a-priori tagged as untrustworthy users, and that trustworthy users can indeed be found among lurkers. *The research line on the identification and ranking of lurkers in social networks is the focus of Chapter 5.*

2

Background

2.1 Summary

This chapter provides preliminary concepts that will help the understanding of the techniques and models adopted in subsequent chapters. We first introduce some background on topological characterization of networks (Section 2.2), focusing on node degree properties (Section 2.2.1), path properties (Section 2.2.3), transitivity (Section 2.2.4), and assortativity (Section 2.2.2). In Section 2.3, we study the problem of vertex centrality in a network, first describing basic centrality measures (Section 2.3.1), then introducing concepts of eigenvector centrality and prestige in Section 2.3.2, deepening the study of fundamental eigenvector centrality algorithms like PageRank in Section 2.3.3 and HITS in Section 2.3.4. In Section 2.4, we introduce the concepts behind heterogeneous network models, followed by an overview on ranking (Section 2.4.1) and clustering (Section 2.4.2) algorithms for heterogeneous information networks. In Section 2.5, we discuss background about PageRank-based algorithms for word sense disambiguation tasks, while in Section 2.6 we review global (Section 2.6.2) and local (Section 2.6.3) trust ranking methods, with particular attention to TrustRank and Anti-TrustRank (Section 2.6.1). Chapter is concluded by Section 2.7, where the assessment criteria used in the experimental phases of subsequent chapters are briefly described.

2.2 Topological characterization

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ denote a network graph, which consists of two sets $\mathcal{V}$ and $\mathcal{E}$, such that $\mathcal{V} \neq \emptyset$ and $\mathcal{E}$ is a set of pairs of elements of $\mathcal{V}$. If the pairs in $\mathcal{E}$ are ordered the graph is said *directed*, otherwise is *undirected*. The elements in $\mathcal{V}$ are the vertices (or nodes) of $\mathcal{G}$, while the elements in $\mathcal{E}$ are the edges (or links) of $\mathcal{G}$. A graph $\mathcal{G}$ can be completely described by its $|\mathcal{V}| \times |\mathcal{V}|$ *adjacency* matrix, denoted with $\mathbf{A}$, such that $A(i,j) = 1$ if $(v_i, v_j) \in \mathcal{E}$, and $A(i,j) = 0$ otherwise. In general, we will use boldface letters to denote vectors (e.g., $\mathbf{x}$)

and boldface, capital letters to denote matrices (e.g., $\mathbf{X}$); moreover, the generic i-th entry of vector $\mathbf{x}$ (resp., (i,j)-th entry of matrix $\mathbf{X}$) will be denoted as $x(v_i)$ (resp., $X(v_i, v_j)$), for any $v_i, v_j \in \mathcal{V}$.

If $\mathcal{G}$ is undirected, for any vertex $v_i \in \mathcal{V}$, we denote with $deg(v_i) = \sum_{v_j \in \mathcal{V}} A(i,j)$ the number of edges incident with v_i, or *degree* of v_i (subsequently denoted also as k_i), and with $\mathcal{V}_i = \{v_j | (v_i, v_j) \in \mathcal{E}\}$ the set of *neighbors* of v_i. If $\mathcal{G}$ is directed, the degree and the set of neighbors have two components: we denote with $B_i = \{v_j | (v_j, v_i) \in \mathcal{E}\}$ and $R_i = \{v_j | (v_i, v_j) \in \mathcal{E}\}$ the set of *in-neighbors*, or "backward vertices", and the set of *out-neighbors*, or "reference vertices", of v_i, respectively. 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.

2.2.1 Node degree, degree distributions and correlations

The *degree* (or connectivity) k_i of a node v_i is the number of edges incident with the node, and is defined in terms of the adjacency matrix $\mathbf{A}$ as:

$$k_i = \sum_{j \in \mathcal{V}} A(i,j)$$

If the graph is directed, the degree of the node has two components: the number of outgoing links $out(v_i) = \sum_j A(i,j)$ (referred to as the out-degree of the node), and the number of ingoing links $in(v_i) = \sum_j A(i,j)$ (referred to as the in-degree of the node). The total degree is defined as $k_i = out(v_i) + in(v_i)$. A list of the node degrees of a graph is called the *degree sequence*. The most basic topological characterization of a graph $\mathcal{G}$ can be obtained in terms of the degree distribution $P(k)$, defined as the probability that a node chosen uniformly at random has degree k or, equivalently, as the fraction of nodes in the graph having degree k.

In the case of directed networks one needs to consider two distributions, $P(k^{in})$ and $P(k^{out})$. Information on how the degree is distributed among the nodes of a undirected network can be obtained by the calculation of the *moments* of the distribution. The n-moment of $P(k)$ is defined as:

$$\langle k(n) \rangle = \sum_k k^n P(k). \tag{2.1}$$

The first moment $\langle k \rangle$ is the mean degree of $\mathcal{G}$. The second moment measures the fluctuations of the connectivity distribution.

The degree distribution completely determines the statistical properties of uncorrelated networks. However a large number of real networks are correlated in the sense that the probability that a node of degree k is connected to another node of degree, say k', depends on k. In these cases, it is necessary to introduce the *conditional* probability $P(k'|k)$ being defined as the probability that a link from a node of degree k points to a node of degree k' [21].

Although the degree correlations are formally characterized by $P(k'|k)$, a direct evaluation of the conditional probability gives extremely noisy results for most of the real networks because of their finite size $N = |\mathcal{V}|$. This problem can be overcome by defining the average nearest neighbors degree of a node v_i as:

$$k_{nn,i} = 1/k_i \sum_{j \in \mathcal{V}_i} k_j = 1/k_i \sum_{j=1}^{N} A(i,j) k_j, \tag{2.2}$$

where $\mathcal{V}_i$ is the set of direct neighbors of v_i. By using Eq. 2.2 one can calculate the average degree of the nearest neighbors of nodes with degree k, denoted as $k_{nn}(k)$, obtaining an expression that implicitly incorporates the dependence on k. Such a quantity can be, indeed, expressed in terms of the conditional probability as

$$k_{nn}(k) = \sum_{k'} k' P(k'|k). \tag{2.3}$$

2.2.2 Assortativity

If there are no degree correlations, Eq. 2.3 gives $k_{nn}(k) = \langle k^2 \rangle / \langle k \rangle$, i.e., $k_{nn}(k)$ is independent from k. Correlated graphs are classified as *assortative* if $k_{nn}(k)$ is an increasing function of k, whereas they are referred to as disassortative when $k_{nn}(k)$ is a decreasing function of k. In other words, in assortative networks the nodes tend to connect to their connectivity peers, while in disassortative networks nodes with low degree are more likely connected with highly connected ones. Degree correlation are usually quantified by reporting the numerical value of the slope of $k_{nn}(k)$, or by calculating the Pearson correlation coefficient of the degrees at either ends of a link.

The existence of degree correlations among nodes is an important property of real networks. Thus, many social networks show that nodes having many connections tend to be connected with other highly connected nodes. In the literature this characteristic is usually denoted as assortativity or assortative mixing [128]. Let ε_{ij} be the probability that a randomly selected edge of the network connects two nodes, one with degree k_i and another with degree k_j. The probabilities ε_{ij} determine the correlations of the network. We say that a network is uncorrelated when:

$$\varepsilon_{ij} = (2 - \delta_{ij}) \frac{k_i P(k_i)}{\langle k_i \rangle} \frac{k_j P(k_j)}{\langle k_j \rangle} \coloneqq \varepsilon^r_{ij}, \tag{2.4}$$

i.e., when the probability that a link is connected to a node with a certain degree is independent from the degree of the attached node. Here $\langle k_i \rangle = \langle k_j \rangle$ denotes the first moment of the degree distribution.

Assortativity means that highly connected nodes tend to be connected to each other with a higher probability than in an uncorrelated network. Moreover, the nodes with similar degrees tend to be connected with larger

probability than in the uncorrelated case, i.e., $\varepsilon_{ij} > \varepsilon^r_{ij} \forall i$. The degree of assortativity of a network can thus be characterized by the quantity:

$$\mathcal{A} = \frac{\sum_i \varepsilon_{ii} - \sum_i \varepsilon^r_{ii}}{1 - \sum_i \varepsilon^r_{ii}} \tag{2.5}$$

which takes the value 0 when the network is uncorrelated and the value 1 when the network is totally assortative.

2.2.3 Shortest path lengths and diameter

Shortest paths play an important role in transferring information through a network. The shortest path between two nodes in a graph $\mathcal{G}$, defined as $d(i,j)$, is the length of the geodesic from node v_i to node v_j. The maximum value of $d(i,j)$ is called the *diameter* of the graph. A measure of the typical separation between two nodes in the graph is given by the *average shortest path length*, also known as *characteristic path length*, defined as the mean of geodesic lengths over all couples of nodes [174]:

$$L = \frac{1}{|\mathcal{V}|(|\mathcal{V}|-1)} \sum_{i,j \in \mathcal{V}, i \neq j} d(i,j) \tag{2.6}$$

A problem with this definition is that L diverges if there are disconnected components in the graph. One possibility to avoid the divergence is to limit the summation in Eq. 2.6 only to couples of nodes belonging to the largest connected component [174]. An alternative approach, that is useful in many cases is to consider the harmonic mean of geodesic lengths, and to define the so-called efficiency of $\mathcal{G}$ as:

$$E = \frac{1}{|\mathcal{V}|(|\mathcal{V}|-1)} \sum_{i,j \in \mathcal{V}, i \neq j} \frac{1}{d(i,j)} \tag{2.7}$$

Such a quantity is an indicator of the traffic capacity of a network, and avoids the divergence of formula 2.6, since any couple of nodes belonging to disconnected components of the graph yields a contribution equal to zero to the summation in Eq. 2.7 [21].

2.2.4 Transitivity

Transitivity, is a typical property of acquaintance networks, where two individuals with a common friend are likely to know each other [173]. In terms of a generic graph $\mathcal{G}$, transitivity means the presence of a high number of triangles. This can be quantified by defining the transitivity T of the graph as the relative number of transitive triples, i.e., the fraction of connected triples of nodes (triads) which also form triangles:

$$T = \frac{3 \times \#triangles\ in\ \mathcal{G}}{\#of\ connected\ triples\ of\ vertices\ in\ \mathcal{G}} \tag{2.8}$$

The factor 3 in the numerator compensates for the fact that each complete triangle of three nodes contributes three connected triples, one centered on each of the three nodes, and ensures that $0 \leq T \leq 1$, with $T = 1$ for K_N. An alternative possibility is to use the graph clustering coefficient C, a measure introduced by in [174], and defined as follows. A quantity c_i (the local clustering coefficient of node v_i) is first introduced, expressing how likely $a_{jm} = 1$ for two neighbors v_j and v_m of node v_i. Its value is obtained by counting the actual number of edges (denoted by e_i) in $\mathcal{G}_i$ (the subgraph of neighbors of v_i).Notice that $\mathcal{G}_i$ can be, in some cases, unconnected. The local clustering coefficient is defined as the ratio between e_i and $k_i(k_i - 1)/2$, the maximum possible number of edges in $\mathcal{G}_i$:

$$c_i = \frac{2ei}{k_i(k_i - 1)} = \frac{\sum_{j,m} a_{ij} a_{jm} a_{mi}}{k_i(k_i - 1)} \tag{2.9}$$

The clustering coefficient of the graph is then given by the average of c_i over all the nodes in $\mathcal{G}$:

$$C = \langle c \rangle = \frac{1}{N} \sum_{i \in \mathcal{V}} c_i. \tag{2.10}$$

By definition, $0 \leq c_i \leq 1$, and $0 \leq C \leq 1$. It is also useful to consider $c(k)$, the clustering coefficient of a connectivity class k, which is defined as the average of c_i taken over all nodes with a given degree k.

2.3 Centrality and prestige

2.3.1 Basic measures

A primary problem in network analysis is the identification of the most *central* vertices in a network. The term central 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. 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 $\mathcal{G}_1$, $\mathcal{G}_2$ are isomorphic and $m(v)$ denotes the mapping function from a node v in $\mathcal{G}_1$ to some node v' in $\mathcal{G}_2$, then the centrality of v in $\mathcal{G}_1$ needs to be the same as the centrality of $m(v) = v'$ in $\mathcal{G}_2$. 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 outdegrees of vertices, while the term *prestige* is introduced to examine the "choices received", or indegrees of vertices [173]. 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 overview well-known measures of centrality, and their definitions for undirected and directed networks. A comprehensive explanation and implication behind centrality and prestige measures can be found in [173].

Vertex-level centrality

The most intuitive measure of centrality for any vertex $v \in \mathcal{V}$ is the *degree centrality*, which is defined as the degree of v:

$$c_D(v) = degree(v) \tag{2.11}$$

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 $|\mathcal{V}| - 1$, the *relative degree centrality* is:

$$\widehat{c_D}(v) = \frac{c_D(v)}{|\mathcal{V}| - 1} = \frac{degree(v)}{|\mathcal{V}| - 1} \tag{2.12}$$

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. 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 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 [140]:

$$c_C(v) = \frac{1}{\sum_{u \in \mathcal{V}} d(v,u)} \tag{2.13}$$

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:

$$\widehat{c_C}(v) = (|\mathcal{V}| - 1)c_C(v) = \frac{|\mathcal{V}| - 1}{\sum_{u \in \mathcal{V}} d(v,u)} \tag{2.14}$$

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) [173].

Besides (shortest) distance, another important property refers to the ability of a vertex to have the 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 [56]:

$$c_B(v) = \sum_{u,z \in \mathcal{V}, u \neq v, z \neq v} \frac{m_{u,z}(v)}{m_{u,z}(\mathcal{V})} \tag{2.15}$$

where $m_{u,z}(v)$ is the number of shortest paths between u and z and passing through v, and $m_{u,z}(\mathcal{V})$ 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 $(|\mathcal{V}| - 1)(|\mathcal{V}| - 2)/2$. Analogously to the other centrality measures, it's recommended to standardize the betweenness to obtain a *relative betweenness centrality*:

$$\widehat{c_B}(v) = \frac{2c_B(v)}{(|\mathcal{V}| - 1)(|\mathcal{V}| - 2)} \tag{2.16}$$

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 $\mathcal{O}(|\mathcal{V}|^3)$ time and $\mathcal{O}(|\mathcal{V}|^2)$ space, algorithms designed for large, sparse networks require $\mathcal{O}(|\mathcal{V}|+|\mathcal{E}|)$ space and $\mathcal{O}(|\mathcal{V}||\mathcal{E}|)$ and $\mathcal{O}(|\mathcal{V}||\mathcal{E}|+|\mathcal{V}|^2 \log |\mathcal{V}|)$ time on unweighted

and weighted networks, respectively [25]. A number of variants of betweenness centrality has also been investigated, for instance related to extensions to edges and to valued networks [26].

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 indegree 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 [149] 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 $|\mathcal{V}| \times |\mathcal{V}|$ 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:

$$c_I(v_i) = \frac{1}{y_{ii} + \frac{1}{|\mathcal{V}|}(\sum_{v_j \in \mathcal{V}} y_{jj} - 2\sum_{v_j \in \mathcal{V}} y_{ij})} \tag{2.17}$$

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:

$$\frac{\sum_{v \in \mathcal{V}} c_D(v)}{|\mathcal{V}||\mathcal{V}-1|} = \frac{\sum_{v \in \mathcal{V}} \widehat{c_D}(v)}{|\mathcal{V}|} \tag{2.18}$$

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 differ in centrality [57]:

$$\mathsf{C} = \frac{\sum_{v \in \mathcal{V}} \max\mathsf{C} - \mathsf{C}(v)}{\max \sum_{v \in \mathcal{V}} \max\mathsf{C} - \mathsf{C}(v)} \tag{2.19}$$

where $\max\mathsf{C}(\cdot)$ is a function that expresses a selected measure of relative centrality, and $\max\mathsf{C}$ is the maximum value of relative centrality over all vertices in the network. 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 Eq. (2.19) is equal to $(n-1)(n-2)$, $(n-1)(n-2)/(2n-3)$, and $(n-1)$, respectively.

2.3.2 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" [144]. 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:

$$r(v) = \sum_{u \in V} w(u,v) r(u) \tag{2.20}$$

where $w(u,v)$ is equal to 1 if u points to v (i.e., u is a in-neighbor of v), and 0 otherwise. Equation (2.20) corresponds to a set of $|\mathcal{V}|$ linear equations (with $|\mathcal{V}|$ unknowns) which can be rewritten as:

$$\mathbf{r} = \mathbf{A}^{\mathrm{T}}\mathbf{r} \tag{2.21}$$

where $\mathbf{r}$ is a vector of size $|\mathcal{V}|$ 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 $|\mathcal{V}|$.

Katz [85] first recommended to manipulate the matrix $\mathbf{A}$ by constraining every row in $\mathbf{A}$ to have sum equal to 1, thus enabling Eq. 2.21 to have finite

solution. In effect, Eq. 2.21 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, Eq. 2.21 has no non-zero solution unless $\mathbf{A}^{\mathrm{T}}$ has an eigenvalue of 1.

A generalization of Eq. 2.21 was suggested by Bonacich [22], 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:

$$\lambda\mathbf{r} = \mathbf{A}^{\mathrm{T}}\mathbf{r} \tag{2.22}$$

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

$$\lambda\mathbf{R} = \mathbf{A}^{\mathrm{T}}\mathbf{R} \tag{2.23}$$

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

Katz [85] also proposed to introduce in Eq. 2.21 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:

$$\mathbf{r} = \sum_{i=1}^{\infty} \alpha^i \mathbf{A}^{\mathrm{T}i}\mathbf{r} \tag{2.24}$$

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 indegree: in fact, according to the assumption that a vertex has no status if it does not receive choices from other vertices, vertices with null indegree 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 [23], probably better known as *alpha-centrality*, is defined as:

$$\mathbf{r} = \alpha\mathbf{A}^{\mathrm{T}}\mathbf{r} + \mathbf{e} \tag{2.25}$$

where $\mathbf{e}$ is a $|\mathcal{V}|$-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 prestiges. The solution of Eq. 2.25 is:

$$\mathbf{r} = (\mathbf{I} - \alpha\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{e} \tag{2.26}$$

It can easily be proved that Eq. 2.26 and Eq. 2.24 differ only by a constant (i.e., one) [23].

2.3.3 PageRank

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 $|\mathcal{V}| \times |\mathcal{V}|$ matrix such that columns refer to those vertices whose status is determined by the connections received from the row vertices:

$$W(i,j) = \begin{cases} 1/out(v_i) & \text{if } (v_i, v_j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

Note that $\mathbf{W} = \mathbf{D}_{\text{out}}^{-1}\mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}_{\text{out}}$ is a diagonal matrix storing the outdegrees of the vertices (i.e., $\mathbf{D}_{\text{out}} = diag(\mathbf{A1})$). Using matrix $\mathbf{W}$, the first three ideas underlying the PageRank can be expressed as $\mathbf{r} = \mathbf{W}^{\text{T}}\mathbf{r}$, or equivalently, for every $v_i \in \mathcal{V}$:

$$r(v_i) = \sum_{v_j \in B_i} \frac{r(v_j)}{out(v_j)} \tag{2.28}$$

Therefore, vector $\mathbf{r}$ is the unique eigenvector of the matrix corresponding to eigenvalue 1. It should be noted that Eq. 2.28 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 [27]. 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. Eq. 2.25) is not necessarily equal to 1.

However, the assumption of graph connectivity behind Eq. 2.28 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., [138]): (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, Eq. 2.28 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 columns of zero, thus making $\mathbf{W}$ substochastic: as a result, by reiterating Eq. 2.28 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:

$$\mathbf{S} = \mathbf{W} + \mathbf{d}\mathbf{1}^{\mathrm{T}}/|\mathcal{V}| \tag{2.29}$$

However, Eq. 2.29 will not solve the problem of sinks. Therefore, Page and Brin [27] 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 jump:

$$\mathbf{r} = \alpha\mathbf{S}^{\mathrm{T}}\mathbf{r} + (1 - \alpha)\mathbf{p} \tag{2.30}$$

Above, vector $\mathbf{p}$, usually called *personalization vector*, is by default set to $\mathbf{1}/|\mathcal{V}|$, but it can be any probability vector. Equation 2.30 can be rewritten as:

$$\mathbf{G} = \alpha\mathbf{S} + (1 - \alpha)\mathbf{E} \tag{2.31}$$

where $\mathbf{E} = \mathbf{1}\mathbf{p}^{\mathrm{T}} = \mathbf{1}\mathbf{1}^{\mathrm{T}}/|\mathcal{V}|$. 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

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

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]

Computing PageRank

As previously indicated, the computation of PageRank requires to solve Eq. 2.31, 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 Eq. 2.30 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. [95]), 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. Eq. 2.29), and hence $\mathbf{G}$, is not required, since the power method can be rewritten as:

$$\boldsymbol{\pi}^{\mathrm{T}(k+1)} = \boldsymbol{\pi}^{\mathrm{T}(k)}\mathbf{G} = \alpha\boldsymbol{\pi}^{\mathrm{T}(k)}\mathbf{W} + (\alpha\boldsymbol{\pi}^{\mathrm{T}(k)}\mathbf{d})\mathbf{1}^{\mathrm{T}}/|\mathcal{V}| + (1-\alpha)\mathbf{p}^{\mathrm{T}} \qquad (2.32)$$

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 $\mathcal{O}(|\mathcal{V}|)$.

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 [95]. Moreover, since the second largest eigenvalue of $\mathbf{G}$ is α [119], it can be shown that the asymptotic rate of convergence of the

[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 if $\mathbf{X}^k > 0$ for some $k > 0$, which is useful to determine whether the power method applied to $\mathbf{X}$ will converge [119]. An irreducible Markov chain with a primitive transition matrix is called an *aperiodic chain*.

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 [136, 95].

Choosing the personalization vector

As previously discussed, the personalization vector $\mathbf{p}$ can be replaced with any vector whose non-negative components sum up to 1 and that can be used to boost the PageRank score for a specific subset of vertices. In particular, if we denote with $\mathcal{B} \subseteq \mathcal{V}$ a subset of vertices of interest, then $\mathbf{p} = \mathbf{1}/|\mathcal{V}|$ is replaced with another vector biased by $\mathcal{B}$, $\mathbf{p}_B$ whose entries are set to $\mathbf{1}/|\mathcal{B}|$ only for those vertices that belong to $\mathcal{B}$, and zero otherwise.

Intuitively, this way of altering the behavior of random surfing reflects the different preferences and interests that the random surfer may have, making the PageRank be query-dependent, or topic-sensitive. Google also originally used personalization mechanisms to specifically control spamming due to the so-called link farms. We shall discuss implications and relating algorithms in trust/distrust contexts later in Section 2.6.

2.3.4 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 Web pages 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 [87, 88]. Like PageRank and other eigenvector centrality methods, HITS still handles an iterative computation of a fixedpoint 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 subnetwork*. 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:

$$\mathbf{a} = \mu \mathbf{A}^{\mathrm{T}} \mathbf{h} \tag{2.33}$$

$$\mathbf{h} = \lambda \mathbf{A} \mathbf{a} \tag{2.34}$$

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: it initializes hub and authority score (e.g., to 1) for every vertex in the expanded query-focused subnetwork, then iterates through Eq. 2.33 and Eq. 2.34, normalizing both vectors at each iteration, until convergence (i.e., a termination tolerance threshold is reached). 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 Eq. 2.33 and Eq. 2.34 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 less 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 TCK 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 on the *tightly knit communities* (TCK) effect, which occurs when relatively many pages are 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" [100]. The latter phenomenon is also called *topic drift*.

Classic solutions to attenuate the TCK effect include accounting for the analysis of contents and/or the anchor texts of the Web pages (e.g., [20, 37]). However, other link analysis approaches have been developed to avoid overly favoring the authorities of tightly knit communities. Lempel and Morgan [100] 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:

$$p_a(i,j) = \sum_{v_q \in B_i \cap B_j} \frac{1}{in(v_i)} \frac{1}{out(v_k)} \tag{2.35}$$

and the Markov chain for hub scores has transition probabilities:

$$p_h(i,j) = \sum_{v_q \in R_i \cap R_j} \frac{1}{out(v_i)} \frac{1}{in(v_k)} \tag{2.36}$$

Lempel and Morgan proved that the authority stationary distribution $\mathbf{a}$ is such that $a(v_i) = in(v_i)/\bigcup_{v \in \mathcal{V}} in(v)$, and that the hub stationary distribution $\mathbf{h}$ is such that $h(v_i) = out(v_i)/\bigcup_{v \in \mathcal{V}} out(v)$. 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 [24]. Nevertheless, the TCK effect is overcome in SALSA through random walks on the hub-authority bipartite network, which imply that authorities can be identified by looking at different communities.

2.4 Heterogeneous information 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., web pages, users) and links are relationships of the same type (e.g., hypertext linkage, friendship). However, it's also the case that 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 non 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 [152], a *heterogeneous information network* (HIN) is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a vertex type mapping function $\tau : \mathcal{V} \to \mathcal{T}$ and an edge type mapping function $\phi : \mathcal{E} \to \mathcal{R}$, where each vertex $v \in \mathcal{V}$ belongs to one particular vertex type $\tau(v) \in \mathcal{T}$, each edge $e \in \mathcal{E}$ 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 $S_{\mathcal{G}} = (\mathcal{T}, \mathcal{R})$, is a meta template for a heterogeneous network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with set of vertex types $\mathcal{T}$ and set of edge types $\mathcal{R}$.

Managing and mining HINs is intuitively more difficult than the homogeneous case. Based on their previous studies in different mining tasks (e.g., ranking-based clustering [155, 158], ranking-based classification [78, 79]), meta-path-based similarity search [154], relationship prediction [151, 153], relation strength learning [150, 156] and network evolution [157]), the authors in [152] provides 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.

2.4.1 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., [15]), Web object ranking (e.g., [130]), expert search in digital libraries (e.g., [63, 183, 45]), link prediction (e.g., [42]), recommender systems and Web personalization (e.g., [73, 98, 84]). Some work has also been developed using path-level features in the ranking models, such as path-constrained random walk [96] and PathSim [154] 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 [155] and NetClus [158] 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 [15]. 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 $\mathcal{G}_D(\mathcal{V}_D, \mathcal{E}_D)$ is a labeled directed graph where every node v has a label $\lambda(v)$ and a set of keywords. Nodes in $\mathcal{V}_D$ 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 $e \in \mathcal{E}_D$ is labeled with a "role" $\lambda(e)$ which describes the relation between the connected nodes.

The *schema graph* $\mathcal{G}_S(\mathcal{V}_S, \mathcal{E}_S)$, is a directed graph which describes the structure of $\mathcal{G}_D$, i.e., it defines the set of node and edge labels. A data graph

$\mathcal{G}_D(\mathcal{V}_D, \mathcal{E}_D)$ conforms to a schema graph $\mathcal{G}_S(\mathcal{V}_S, \mathcal{E}_S)$ if there is a unique assignment μ such that:

1. for every node $v \in \mathcal{V}_D$ there is a node $\mu(v) \in \mathcal{V}_S$ such that $\lambda(v) = \lambda(\mu(v))$;
2. for every edge $e \in \mathcal{E}_D$ from node u to node v there is an edge $\mu(e) \in \mathcal{E}_S$ 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* $\mathcal{G}_A(\mathcal{V}_S, \mathcal{E}_A)$ reflects the authority flow through the edges of the graph. In particular, for each edge in $\mathcal{E}_S$ 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 $\mathcal{G}_D(\mathcal{V}_D, \mathcal{E}_D)$ and a conforming authority transfer schema graph $\mathcal{G}_A(\mathcal{V}_S, \mathcal{E}_A)$ a *authority transfer data graph* $\mathcal{G}_{AD}(\mathcal{V}_D, \mathcal{E}_{AD})$ 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:

$$\mathbf{r}^w = d\mathbf{A}\mathbf{r}^w + \frac{1-d}{|S(w)|}\mathbf{s} \tag{2.37}$$

where $S(w)$ is the base set, 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 [15], 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. In Section 3.6.2.1 of Chapter 3, we shall resort to this serialization mechanism to investigate the efficiency benefits deriving from a DAG-constrained serialization of our proposed method.

PopRank

In [130], 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:

$$\mathbf{r}_\tau = \alpha\mathbf{r} + (1-\alpha)\sum_{\tau_t} \gamma_{\tau_t\tau_0}\mathbf{M}^{\mathrm{T}}_{\tau_t\tau_0}\mathbf{r}_{\tau_t} \tag{2.38}$$

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.

Path-based ranking

The Path Ranking Algorithm (PRA) proposed in [96] is a Personalized PageRank that treats edge label sequences as features for a linear model. Given a set of query entities, for any relation path of bounded length a distribution is defined, and hence a linear model is built by assigning a weight to every path. The weight parameters are estimated using a supervised process involving optimization procedures (L-BFGS) and loss functions (binomial log-likelihood).

PRA is used as a proximity measure in various relational retrieval tasks such as venue recommendation, reference recommendation, expert finding and gene recommendation that can be formulated as *typed proximity queries*. The method also supports two additional types of experts: *query independent experts*, that returns a PageRank-like global ranking scheme, and *popular entity experts*, that allows the ranking to be adjusted for particular entities that are especially important by adding biases and query-conditioned biases to the target entities.

Closely related to multi-relational PageRank models are also the methods developed in [122, 123]. In [122], a lazy random graph walk is applied to an entity-relation heterogeneous graph to derive an extended measure of entity similarity; the model is called "lazy" since a fixed probability is set to halt the walk at each step, which makes short walks more probable while reducing the probability of reaching nodes distant from the starting point of the walk. Two learning approaches for tuning the edge weights are described: a hill-climbing method based on error back-propagation, and a re-ranking method; both methods are based on the analysis of the set of paths leading to every candidate node. The framework in [122] is also exploited in [123] for heterogeneous graphs built over a parsed text collection. A set of dependency parse

trees is modeled, where each tree has labelled directed links between words that describe relevant grammatical relations (e.g., nominal subject, indirect object). A major novelty in [123] is the introduction of a path-constrained graph walk, which relies on a dynamic evaluation of the edge weights during the walk.

2.4.2 Ranking-based clustering

RankClus

In [155], the RankClus algorithm is introduced, which integrates clustering and ranking on a bi-typed information network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, such that $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$, with $\mathcal{V}_0 \cap \mathcal{V}_1 = \emptyset$. Hence, the nodes in the network belong to one of two predetermined types, herinafter 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 [155]: *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 $\mathcal{V}_1$, $\mathbf{r}_{\mathcal{V}_1|C_k}$, can be viewed as a rank distribution of $\mathcal{V}_1$, which in fact is a measure for cluster C_k. Then, for each node $v \in \mathcal{V}_0$, the distribution of object $u \in V_1$ can be viewed as a mixture model over K conditional ranks of $\mathcal{V}_1$, and thus can be represented as a K dimensional vector in the new space [155]. 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 $\mathcal{G} = (\mathcal{V}_0 \cup \mathcal{V}_1, \mathcal{E})$, the ranking functions for $\mathcal{V}_0$ and $\mathcal{V}_1$, and a number K of clusters, RankClus produces K clusters over $\mathcal{V}_0$ with conditional rank scores for each $v \in \mathcal{V}_0$, and conditional rank scores for each $u \in \mathcal{V}_1$. The main steps of the RankClus algorithm are summarized as follows.

- Step 0: Initialization. Assign each target node with a cluster label from 1 to K randomly.
- Step 1: Ranking for each cluster. Based on current clusters, calculate conditional ranks for nodes of type $\mathcal{V}_1$ and $\mathcal{V}_0$ and within-cluster ranks for nodes of type $\mathcal{V}_0$. In this step, we also need to judge whether any cluster is empty, which may be caused by the improper initialization or biased running results of the algorithm. When some cluster is empty, the algorithm needs to restart in order to generate K clusters.
- Step 2: Estimation of the mixture model component coefficients. Estimate the parameter Φ in the mixture model, get new representations for each target object and centers for each target cluster: $\mathbf{s}_v$ and $\mathbf{s}_{C_k}$. In practice, the iteration number t for calculating Φ only needs to be set to a small number.
- Step 3: Cluster adjustment. Calculate the distance from each object to each cluster center and assign it to the nearest cluster.
- Repeat Steps 1, 2 and 3 until clusters change only by a very small ratio ϵ or the iteration number is bigger than a predefined number of iterations.

NetClus

NetClus [158] 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, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, with $T+1$ types of objects such that $\mathcal{V} = \{\mathcal{V}_t\}_{t=0}^{T}$), has a star network schema if $\forall\ e = (v_i, v_j) \in \mathcal{E}, v_i \in \mathcal{V}_0 \wedge v_j \in \mathcal{V}_t (t \neq 0)$ 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.

- 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: Build ranking-based probabilistic generative model for each net-cluster, i.e., $\{P(v|C_k^t)\}_{k=1}^K$.
- Step 2: Calculate the posterior probabilities for each target object ($P(C_k^t|v)$) and then adjust 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 cluster does 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.

2.5 PageRank for WSD problems.

In word sense disambiguation (WSD), the goal is to associate a given word in a text or discourse context with a semantic definition, or sense, which is distinguishable from other potential senses of that word. We acknowledge the existence of important studies concerning the application of PageRank to semantic networks inferred from natural language texts specifically for word sense disambiguation problems (e.g., [121, 5, 6, 43]. An early attempt to bring PageRank to WSD is proposed in [121], where traditional PageRank is applied on a graph built over WordNet synsets (vertices) and edges are drawn

using synset relations available in WordNet. Two approaches are also defined to refine the basic PageRank method or the ranking it computed: the first approach consists in using the Lesk algorithm to provide PageRank with a initial ranking of nodes, while the second approach is to combine the ranking obtained by PageRank with WordNet sense frequency information. According to the experimental results PageRank outperformed Lesk, while combining the two methods however did not bring any significant improvement over the individual methods performance when sense order is taken into account. In a further study [146], a weighted variant of PageRank is also proposed, where edges are weighted using one of classic WordNet-based word similarity measures [32].

A different graph context for PageRank-based WSD is devised in [5]. The underlying idea is to extract an undirected subgraph of WordNet which links the synsets of words in the input text, and then again apply the basic PageRank over the subgraph. This subgraph is obtained by the union of the subgraphs corresponding to the shortest paths that connect all pairs of concepts of the input tag names. An early proposal of personalized PageRank for WSD is introduced in [6], where the personalization vector is initialized with the synsets of the words in the input text. This personalized PageRank method utilizes the full (undirected) WordNet graph, where synsets are connected by WordNet relations, with the addition of a directed subgraph whose vertices are the input words and edges are links to the synset vertices of the WordNet graph. Inspired by the topic-sensitive PageRank approach [72], the initial probability mass is concentrated uniformly over the word vertices, which act as source nodes injecting mass into the corresponding synset vertices and spread their mass over the WordNet graph. Moreover, mutual reinforcement effect between semantically related synsets of the same word is alleviated by a variant called W2W. For each input word, W2W aims to concentrate the initial probability mass only over the synsets of the words surrounding that input word. Clearly, W2W is less efficient than personalized PageRank since it needs as many runs as the number of the input words. To improve the efficiency in personalized PageRank, [43] propose to exploit Latent Semantic Analysis (LSA) [44] in order to integrate latent semantic relations between words in the initialization of the personalization vector. The input text is lexically expanded by including those words in the vocabulary that have a cosine similarity with the term frequency - inverse document frequency representation of the text in the LSA space above a certain threshold.

The random walk model proposed in [139] is employed by [112] for the purpose of Web query term disambiguation. The intuition behind the approach is that the PageRank score of a Web page may serve as an indicator of how significant the dominant senses of a query term in the page are, under the assumption that the word sense usage by the Web information seeker follows the behavior of the average Web information provider. Using the WordNet graph, a weight for every sense of each term is computed according to the PageRank scores of the pages containing an instance of that term with that

specific sense. Then, a graph of the senses is constructed and used to perform a personalized PageRank with the various senses having the initial weights computed at the first step.

[147] address the problem of *contextual synonym expansion*, which can be seen as a particular WSD which aims to unsupervisedly learn the correct sense of a target word by selecting its synonyms (lexical substitutes) in a given context. The approach relies on a combination of directional similarity and graph centrality methods including TextRank, in which case the weight of the edge connecting word w_1 to w_2 is computed as directly proportional to the cosine similarity between the ESA [58] vectors of the two words and to the proportion of articles containing w_1 that also contain w_2.

It should be noted that the approaches in [121, 5, 6] were the objective of a comparative evaluation in previous works [160, 159]. These showed a poor effectiveness of word sense disambiguation methods conceived for plain text when applied to tree-structured text, and hence brought for the first time a PageRank-based approach in the context of sense ranking for labeled tree data. More precisely, in [159] the focus was on the definition of methods for determining semantic relatedness in tree data and on the assessment of the impact of these methods on structural sense ranking, whereas in [160] an evaluation framework based on various formulations of PageRank was developed for sense ranking in tree data.

However, in those studies, tree structural relations were individually used to define the ranking context graphs, and hence the ranking stage was not aware of the different types of structural relations, which is instead a major point of the study presented in Chapter 3.

2.6 Trust in online social networks

In this section we discuss previous literature pertinent to trust/distrust ranking. In a ranking task, trust can be regarded as feature either of a node or a pair of nodes in the network, i.e., a measure of how a user is seen as trustworthy either by any other user or by a specific user in the network. Therefore, trust/distrust ranking can be broadly categorized as: (i) *global* trust methods, which produce a ranking of the nodes in a network according to some notion of trust/distrust, and (ii) *local* trust methods, in which the produced trust scores are computed for any direct/indirect relationship between a trustor and a trustee.

2.6.1 TrustRank and Anti-TrustRank

Many solutions for trust ranking have been developed in the past years by resorting to PageRank-style methods [86, 170, 134, 66]. However, PageRank is vulnerable to adversarial information retrieval, i.e., link spamming techniques can enable web pages to achieve higher score than what they actually deserve.

A well-known method that was introduced to combat web spam and finally detect trustworthy pages is *TrustRank* [69]. The key assumption of this method is the approximate isolation principle, i.e., high-quality pages are unlikely to point to spam or low-quality pages. The TrustRank algorithm consists of three steps:

1. Compute a *seed set* of pages labeled by an *oracle function*, obtained by a ranking based on the *inverse-PageRank*.
2. Run the biased PageRank algorithm on the normalized graph matrix using the "good part" of the seed set as the teleportation set, with uniform probability of teleportation.
3. Rank the pages in decreasing order of TrustRank score.

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, inverse-PageRank is computed by reversing the in-links and out-links in the graph, i.e., by running PageRank on the transpose of the graph matrix; a high inverse pagerank indicates that trust can flow with a small number of hops along out-links.

Anti-TrustRank [91] follows an intuition similar to TrustRank, however it is designed to detect untrustworthy pages. It starts with a seed set of spam pages and propagates distrust in the reverse direction. Like TrustRank, Anti-TrustRank consists of three steps:

1. Compute a *seed set* of spam pages labeled by an *oracle function*, obtained by a ranking based on the PageRank.
2. Run the biased PageRank algorithm on the normalized transposed graph matrix using the seed set as the teleport set, with uniform probability of teleportation.
3. Rank the pages in decreasing order of Anti-TrustRank score.

2.6.2 Global trust ranking

Following the example of TrustRank, other PageRank-like global trust methods have been developed in the last years. In [177], Wu et al. propose trust/distrust propagation and aggregation techniques to be integrated with algorithms like TrustRank. For each node in a web graph, two scores (trust and distrust) are maintained: the former flows out of a page's out-neighbors, while the latter propagates following the in-neighbors (in the opposite direction). A method for combining trust and distrust is also introduced, using a simple linear combination of the trust and distrust values. Techniques for improving the seed set selection have also been proposed, such as the automatic seed set expansion method by Zhang et al. [184]. Here an initial seed set selection is performed based on PageRank and Inverse PageRank, to ensure the maximum number of new seeds that can be added during the expansion process.

The PageTrust algorithm by Kerchove and Van Dooren [86] is an extension of PageRank that accounts for both positive and negative links in a web graph. The key idea is that the users' opinions can be influenced by negative links. This is represented by a distrust matrix, whose diagonal represents the degree of distrust of each node. PageTrust requires two more parameters than PageRank: a binary parameter to control whether the users keep their opinions after random jumping, and another factor to control the weight of negative links.

Ortega et al. [134] define the PolarityTrust algorithm upon PolarityRank by the same authors, which is in turn an extension of PageRank to classify the nodes in a network into positive and negative ones, maintaining two different ranking scores with opposite polarities. PolarityTrust hence computes two scores for each node, representing the positive and negative reputation, respectively, of a user. Two sets of authoritative nodes, called source of trust and source of distrust, are selected to bias the ranking, and the propagation process is refined using two mechanisms designed to deal with malicious users (non-negative propagation and action-reaction propagation).

Approximation algorithms have also been proposed to solve global trust ranking problems. Graham et al. [66] propose Dirichlet PageRank to rank the nodes of a network according to a selected node, under the assumption that the node trusts a small subset of the network. The goal is to obtain a ranking that is similar to the known values of trust of the selected node. The algorithm can be adapted to penalize spam nodes, to personalize a previously computed global ranking, or to validate a ranking function for a new node. To this purpose, Dirichlet PageRank performs an iterative procedure in which the PageRank vector is updated until the residual is lower than a given threshold.

Other works exploit trust information to provide personalized recommendations to social network users. For instance, the global ranking method by Varlamis et al. [168] combines recommendations from trusted (or neighboring) users with recommendations from the influential users of a social network. The effects of link analysis metrics (e.g., degree centrality, closeness centrality, betweenness centrality, HITS, and PageRank) on the ranking process are investigated.

2.6.3 Local trust ranking

Local trust ranking has traditionally been addressed by heuristics whose common approach is to compute a trust score for a target node given a source node, under constraints on the maximum length of searching path in the trust graph and minimum trust threshold. In this regard, one of the most recent methods is the TISoN algorithm by Hamdi et al. [71], which searches all the simple paths of length at most l between two nodes that are not directly connected in the trust graph. Only the paths in which every edge has a trust value above a certain threshold are retained. Once a set of admissible trust

paths is constructed, the estimated trust value between the two nodes is inferred by computing, for each path, a weighted combination of the average trust path length, the path variance, and the path weight. The path with maximum strength between the two nodes determines the indirect trust value between them.

The notion trust path was first introduced in the Golbeck's TidalTrust algorithm [62]. Compared with TISoN, TidalTrust only considers the shortest trust paths from the source to the target, while TISoN takes into account every trust path; moreover, TidalTrust computes the trust threshold as the maximal strength over the trust paths, while in TISoN a fixed threshold is employed. TISoN also extends RN-Trust by Taherian et al. [163]. RN-Trust considers the trust network as a resistive network, modeling each relation between nodes as a resistor with resistance inversely proportional to the trust value of the relation. Another algorithm taken into account in the design of TISoN is SW-Trust by Jiang et al. [81]. SW-Trust infers the trust value between two nodes using a depth-limited breadth-first-search, multiplying the trust values in each path and averaging the strength of every path.

Eigenvector-centrality approaches have also been defined for local trust ranking. An exemplary method of this category is TrustWebRank by Walter et al. [170], which is based on the notion of feedback centrality, similar to alpha-centrality, and includes an extension to deal with the time dimension and dynamically update trust scores.

2.7 Assessment criteria

In this section are briefly described the assessment criteria we are going to use in the following chapters in order to comparatively evaluate our proposed methods' performance with respect to the competing methods.

Kendall tau rank correlation coefficient

Kendall correlation [1] 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:

$$\tau(\mathcal{L}', \mathcal{L}'') = 1 - \frac{2\Delta(\mathcal{P}(\mathcal{L}'), \mathcal{P}(\mathcal{L}''))}{M(M-1)}$$

where $\mathcal{L}'$ and $\mathcal{L}''$ are the two rankings to be compared, $M = |\mathcal{L}'| = |\mathcal{L}''|$ and $\Delta(\mathcal{P}(\mathcal{L}'), \mathcal{P}(\mathcal{L}''))$ is the symmetric difference distance between the two rankings, calculated as 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's intersection metric

Fagin measure [52] allows for determining how well two ranking lists are in agreement with each other. This is regarded as the problem of comparing "partial rankings", since elements in one list may not be present in the other list. Moreover, according to [175], a ranking evaluation measure should consider top-weightedness, i.e., the top of the list gets higher weight than the tail. Applied to any two top-k lists $\mathcal{L}', \mathcal{L}''$, the Fagin score is defined as:

$$F(\mathcal{L}', \mathcal{L}'', k) = \frac{1}{k} \sum_{q=1}^{k} \frac{|\mathcal{L}'_{:q} \cap \mathcal{L}''_{:q}|}{q}$$

where $\mathcal{L}_{:q}$ 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.

Binary Preference function (Bpref)

Bpref [29] evaluates the performance from a different view, i.e., the number of non-relevant candidates. It computes a preference relation of whether judged relevant candidates R of a list $\mathcal{L}'$ are retrieved, i.e., occur in a list $\mathcal{L}''$, ahead of judged irrelevant candidates N, and is formulated as

$$Bpref(R, N) = \frac{1}{|R|} \sum_{r} \left(1 - \frac{\#\text{of } n \text{ ranked higher than } r}{|R|} \right)$$

where r is a relevant retrieved candidate, and n is a member of the first $|R|$ irrelevant retrieved candidates.

Normalized Discounted Cumulative Gain (nDCG)

Let $\mathcal{L}^*$ and $\mathcal{L}$ denote the ideal ranking and the ranking produced by an algorithm, respectively. nDCG [76] measures the usefulness (gain) of an item based on its relevance and position in a list. Formally, nDCG is the ratio between the discounted cumulative gain to its ideal (reference) counterpart taking into account the top-k-ranked items in two lists:

$$nDCG^{(k)} = \frac{DCG^{(k)}}{IDCG^{(k)}}$$

Discounted cumulative gain is based on the assumption that highly relevant items appearing in lower positions in a list should be more penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. It is defined as:

$$DCG^{(k)} = \mathcal{L}^*[\arg \mathcal{L}(1)] + \sum_{i=2}^{k} \frac{\mathcal{L}^*[\arg \mathcal{L}(i)]}{\log_2(i+1)}$$

where symbol $\arg \mathcal{L}(i)$ is used to denote the item number of the item ranked at position i in the algorithm's ranking (i.e., $\mathcal{L}^*[\arg \mathcal{L}(i)]$ is the reference ranking value for that item), and

$$IDCG^{(k)} = \mathcal{L}^*(1) + \sum_{i=2}^{k} \frac{\mathcal{L}^*(i)}{\log_2(i+1)}$$

Mean Average Precision (MAP)

MAP is the mean value of the average precisions computed for a set of queries. The average precision for a single query is calculated as

$$AP = \frac{\sum_{n=1}^{k} P@n \cdot rel(n)}{|R|},$$

where $P@n$ is precision at step n (i.e., fraction of the top-n retrieved results that are relevant for the given query) and $|R|$ is the number of relevant candidates.

3

A multi-relational approach to structural sense ranking

3.1 Summary

In this chapter, the problem of structural sense ranking for tree data is addressed by using a multi-relational PageRank approach. By considering multiple types of structural relations, the original tree structural context is better leveraged and hence used to improve the ranking of the senses associated to the tree elements. Upon this intuition, PageRank-based formulations that exploit heterogeneity of links to address the problem of structural sense ranking in tree data are developed, representing a significant research advance in the application of PageRank-style methods to semantic graphs inferred from semistructured/plain text data. Experiments on a large real-world benchmark have confirmed the performance improvement hypothesis of a multi-relational approach.

3.2 Introduction

Tree-shaped data are pervasively used to model real-life objects and their structural relationships. Since the advent of XML, semantic-rich information with an inherent (hierarchical) logical organization has found a convenient way to be managed and exchanged. However, the flexibility in information modeling makes tree data diverse across different information sources and often also within the same source. As a consequence, the coupling of semantics with (tree structural) syntactic information in tree data is subject to lexical ambiguity issues. Disclosing the semantics underlying the structural constituents of tree data is an important task, which is essential to enable a number of applications, ranging from the mapping and integration of conceptually related information in tree-structured schemas, to the semantics-aware similarity search in heterogeneous Web data, from the organization (i.e., clustering, classification) of semantically related documents, to the definition of summaries for different semantic views over data collections.

The presence of varying degrees of structuredness that are used to explain the logical organization of the information in tree data raises a number of new challenges that make the sense disambiguation a different problem than classic lexical ambiguity issues in plain text. Moreover, it is quite common that multiple fine-grained senses may be correct (at different confidence levels) for a given term; consequently, it might be more useful for retrieval and data management purposes to produce a contextual ranking of the senses rather than to decide exactly for a single sense and regard it as the only appropriate one.

Structural sense ranking is recognized as challenging in database and information retrieval research, whereby solutions to other semantics-aware problems can be complemented or supported—these problems include schema similarity search and matching, keyword searching, feature extraction, document classification, document clustering. Previous studies [159, 160] have focused on the development of a tree-structure-aware method for sense ranking in tree data, finally demonstrating that effective solutions actually cannot ignore the structural relations existing over the tree elements.

The network of meanings underlying the structural constituents of tree data can be conveniently represented as a labeled (weighted) graph, therefore graph-based ranking methods are natural candidates to solve the structural sense ranking problem. In particular, eigenvector-centrality methods, like PageRank, have been already used to build graph-based models to support a variety of tasks for natural language processing, including not only disambiguation (e.g., [121, 6]) but also text summarization, sentence extraction, text similarity, term reweighting (e.g., [51, 120, 139, 166]). The underlying assumption is that in a cohesive text related lexical concepts (word meanings) tend to occur together and form a semantic network that can be used to build a discourse understanding model. As discussed in [121], PageRank methods on lexical semantic networks intuitively implement the concepts of text cohesion and relevance of word meanings in a text; i.e., high-ranked meanings are endorsed by related meanings, where preferred recommendations are made by most influential meanings (which in turn are highly recommended by other related meanings). Starting from the study in [121], PageRank methods have indeed shown to improve effectiveness of knowledge-based word sense disambiguation methods.

Notably, PageRank methods have also been successfully used for solving the structural sense ranking problem, and they have shown to generally outperform other graph-based approaches that do not rely on PageRank-style methods—intuitively, this can be explained since the global ranking mechanism by PageRank, which is based on propagation/attenuation properties of the importance scores, is well-suited for structural sense ranking where the semantics in portions of tree data might strongly rely on the whole logical structure of the tree data.

A major aspect that remains unexplored is related to the opportunity of modeling a semantic network for tree data as a *heterogeneous information*

network (HIN). In a HIN, the subtlety and multiplicity of tree structural relations that hold among the underlying concepts in a tree data would represent a key element to enable a ranking algorithm propagating the importance scores through different multi-typed relations. Our intuition is that a multi-relational ranking method should in principle be able to better leverage the semantics of annotations in tree data that are structurally related at different levels. For instance, translated in XML terms, the hierarchical organization of markup tags could be exploited to build a semantic HIN wherein vertices (i.e., concepts underlying the tags) would be linked to each other based on an ensemble of tree-structure relations that are defined over the tags, i.e., ancestor/descendant and sibling relations, at various levels of granularity. As a different scenario, the concepts underlying the descriptions (topic terms) of pages hierarchically organized in a Web directory could be modeled as vertices in a HIN. Edges could be drawn to reflect the Web category taxonomy (analogously to the above scenario), but also to map the hyperlinks between associated resources; such hyperlink relations could further be refined to account for different types of link endorsement (e.g., based on a set of predetermined types of blocks or information boxes of a web page), or to account for cross-lingual semantics (e.g., by following links to pages discussing the same content but written in different languages).

Contributions. While existing research has filled a lack of knowledge on the suitability of PageRank-style methods to semantic networks for the structural sense ranking problem [159, 160], no investigation on the presumed benefits deriving from a HIN representation of the structural semantics in tree data has been made so far. In this work we are hence interested in exploring the structural sense ranking problem in semantic networks inferred from tree data, when multiple types of tree structural relations are taken into account. We believe this joins an important issue due to the ever increasing demand for knowledge-driven applications to manage tree data through the emerging paradigm of dealing with mixed type information in graph models. With the purpose of pushing towards the study of multi-relational PageRank methods in multi-typed semantic networks, we propose a novel PageRank-based framework for structural sense ranking, for which different approaches are developed to deal with multiple types of tree structural relations.

We define an approach that consists of a weighted PageRank model for a tree-structure-aware semantic multidigraph. We also present two alternative formulations of the PageRank-based structural sense ranking problem, the first essentially leading to a combination of multiple independent PageRank instances for single-type tree structural relations, and the second based on the assumption of biasing the PageRank by means of multi-typed structural relations. We also provide insights into properties and complexities of the proposed methods.

A further point of novelty of this work is represented by the definition of a probabilistic Monte Carlo type approximation algorithm for our multi-

relational ranking approach. We also investigate an optimization of our main proposed method on directed acyclic graphs.

Developing a complex HIN model following the lead of prominent examples in the literature is out-of-scope of our work. However, it should be noted that the application domain in which ranking in HINs is addressed in this work (cf. Section 3.3.1) represents a point of novelty. In order to briefly highlight main differences from other works, the HIN in our framework is simple in terms of vertex set (i.e., vertices are all of the same type), however multiple structural relations induce multi-typed edges that can also be drawn between the same pair of vertices; by contrast, parallel edges are not handled in most existing HINs.

We take into account the weighting of edge types by unsupervised learning schemes (cf. Section 3.3.1.3), which require neither any training set based on a domain-expert-provided ranking [130] nor ad-hoc specified criteria [15]. Moreover, our HIN does not need to follow a particular topology like a bipartite graph, as in [155, 98], or star network schema, as in [158].

Our extensive experimentation on a large real-world benchmark of XML data has assessed the significance of a multi-relational approach to the structural sense ranking problem, and finally demonstrated that better ranking solutions are obtained when multiple types of tree structural relations are taken into account.

The remainder of this chapter is organized as follows. Section 3.3 describes our proposed structural sense ranking framework, and provides formal details about the construction of ranking context graphs and the ranking methods. Section 3.4 describes our Monte Carlo type approximation algorithm. Section 3.5 and 3.6 present experimental methodology and results. Section 3.7 shows an application of our multi-relational algorithm to a sense ranking task over plain text. Section 3.8 concludes the chapter and provides pointers for future research.

3.3 Structural sense ranking framework

Let $\mathcal{D}$ denote a labeled tree data instance rooted in a node with label t_0, and let $T(\mathcal{D}) = \{t_0, t_1, \ldots, t_n\}$ be the set of tree element labels in $\mathcal{D}$. We will refer to $T(\mathcal{D})$ as T, if the input tree data is clear from the context, and to the elements in $T(\mathcal{D})$ as *tags*. For each tag $t \in T$, the set of concepts or *senses* of t available in the reference *lexical ontology* is denoted as $\mathcal{C}(t)$. Our general task for structural sense ranking in tree data is summarized as follows:

Definition 3.1 (Structural Sense Ranking). *Assume the availability of a lexical ontology as source of information about word meanings (senses). Given a labeled tree data instance $\mathcal{D}$ and selected a set $\mathcal{T}$ of structural relations for the tag nodes in $\mathcal{D}$, the task of structural sense ranking consists in computing a ranking of all concepts associated with each tag in $\mathcal{D}$. A semantic network is*

built over the tag concepts such that it is aware of the multiple structural relations underlying the tags in $\mathcal{D}$, and a suitably defined PageRank-style method is applied on this semantic network to yield the ranking of the tags' concepts.

We present next our solutions to accomplish this goal, which adopt different approaches to handle multi-typed tree structural relations. We end this section with a discussion on computational complexity aspects of the proposed methods.

3.3.1 The Multi-Structure Semantic PageRank approach

In our first approach to solving the structural sense ranking problem, key elements are the definition of a tree-structure-aware semantic multidigraph as ranking context and a multi-relational weighted PageRank as ranking method.

3.3.1.1 Tree-structure-aware semantic multidigraph

We build the ranking context graph, named tree-structure-aware semantic multidigraph, upon the following methodology.

1. Vertices correspond to all concepts of the tags that belong to the input tree data instance.
2. Edges are drawn between two tags' concepts if a selected *structural relation* holds in the tree instance for any two nodes that are respectively labeled with the two tags. Concepts of the same tag should not be connected to each other in order to avoid undesired mutual reinforcement effects in the concept ranking; as an exception, since the same concept can in principle belong to different tags, self-loops might be drawn if the concept is shared by two structurally connected tag nodes.
3. Edge weights are computed to express the strength of association between any two connected concepts: this should rely primarily on the semantic relatedness between the concepts but should also consider the impact of the repetition of substructures across the input tree instance.

Formally, we define the ranking context graph as a directed multigraph (multidigraph) of the form $\mathcal{G} = \langle \mathcal{V}, \mathcal{T}, \mathcal{E}, w \rangle$ such that:

- $\mathcal{V} = \{c \mid c \in \mathcal{C}(t), t \in T\}$.
- $\mathcal{T} \subseteq \mathcal{T}_0$, where $\mathcal{T}_0$ denotes the domain of *structural relations* for the tag nodes in $\mathcal{D}$. Hence, $\mathcal{T}$ is regarded as the selected set of structural relations that corresponds to the set of edge-types in $\mathcal{G}$.
- $\widetilde{\mathcal{E}} = \bigcup_{\tau \in \mathcal{T}} \widetilde{\mathcal{E}}(\tau)$, such that $\widetilde{\mathcal{E}}(\tau) = \{(c_i, c_j, \tau) \mid c_i \in \mathcal{C}(t), c_j \in \mathcal{C}(t'), t, t' \in T \ \wedge \ t' \xrightarrow{\tau} t\}$. Function $t' \xrightarrow{\tau} t$ applies to a pair of tags t, t' and returns a boolean value depending on whether the structural relation $\tau \in \mathcal{T}$ holds in $\mathcal{D}$ between two nodes labeled with t' and t, respectively.

- $w : \widetilde{\mathcal{E}} \rightarrow \Re^*$ is an edge weighting function defined, for each $(c_i, c_j) \in \widetilde{\mathcal{E}}$,[1] as:

$$w(c_i, c_j) = semrel(c_i, c_j) \times sf(c_i, c_j) \tag{3.1}$$

  In (3.1), *semrel* is a non-negative real-valued function that corresponds to a selected measure of word semantic relatedness. Function sf calculates the frequency of occurrence of a direct structural relation underlying the associated tag nodes relating to two concepts, and is defined as:

$$sf(c_i, c_j) = 1 + \log_{fo(\mathcal{D})} \big(\prod_{t,t'} (1 + freqPC(t, t')) \big) \tag{3.2}$$

  where t, t' are such that $c_i \in \mathcal{C}(t), c_j \in \mathcal{C}(t')$, $fo(\mathcal{D})$ is the average fan-out of $\mathcal{D}$, and $freqPC(t, t')$ is the number of times that t' is a child node of t in $\mathcal{D}$. Function sf acts as an augmenting factor for those concept edges whose associated tag nodes are more frequently linked in the tree instance.
- $\mathcal{E} \subseteq \widetilde{\mathcal{E}}$ such that $\mathcal{E} = \{e{=}(c_i, c_j) \mid e \in \widetilde{\mathcal{E}} \wedge w(e) > 0\}$. Note that condition $w(e) = 0$ holds only if $semrel(e) = 0$, for any edge e.

Our definition of ranking context graph is general as it does not impose any particular (set of) structural relations (for drawing the edges) and semantic relatedness measures (for weighting the edges). Nonetheless, to provide a complete specification, we address the above two aspects as follows. Concerning the selection of structural relation types, we define the domain $\mathcal{T}_0$ by focusing on binary functions that capture the relative position of nodes in a subtree:

- $\tau = \textit{childOf}$: $t' \xrightarrow{\tau} t$ holds if t' is child of t;
- $\tau = \textit{descOf}$: $t' \xrightarrow{\tau} t$ holds if t' is descendant of t;
- $\tau = \textit{child}|\textit{siblchildOf}$: $t' \xrightarrow{\tau} t$ holds if t' is child of t or child of a t's sibling;
- $\tau = \textit{desc}|\textit{sibldescOf}$: $t' \xrightarrow{\tau} t$ holds if t' is descendant of t or descendant of a t's sibling.

Note that the property of domain independence holds to ensure the full applicability of $\mathcal{T}_0$. Nevertheless, underlying domain peculiarities along with side information could suggest alternative interpretations of the relations among the constituents of tree data.

The other aspect, which concerns the definition of function *semrel* for weighting the graph edges, is elaborated on in the next section. In Fig. 3.1 we show an example tree-structure-aware semantic multidigraph.

3.3.1.2 Edge weighting measures

We resort to classic measures of knowledge-based semantic relatedness known as ontology-path-based relatedness (*p-rel*), information-content-based relatedness (*ic-rel*), and gloss-overlap-based relatedness (*go-rel*), which have been

[1] Edge notation is simplified (i.e., pair of vertices) when there is no dependency on a particular structural relation type, as for the edge weighting function.

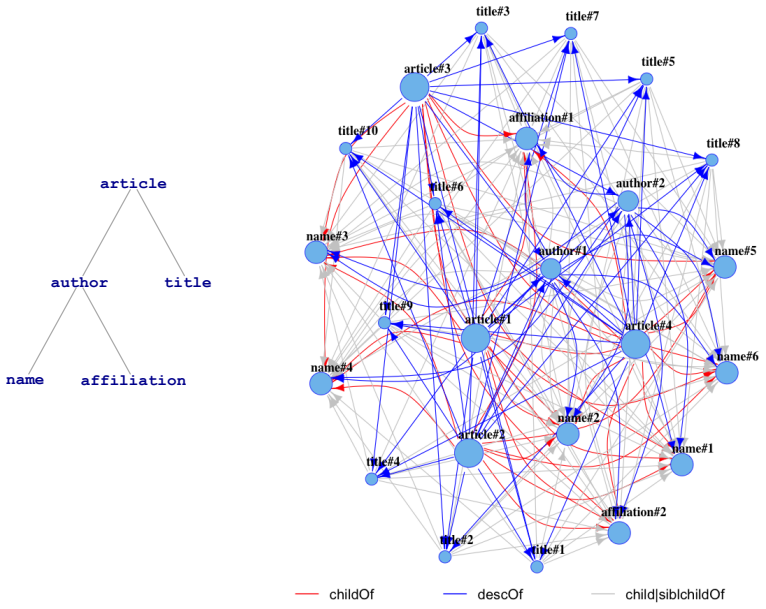


Fig. 3.1. An example tree data (on the left) and possible associated tree-structure-aware semantic multidigraph (on the right). Edges correspond to *childOf*, *descOf*, and *child|siblchildOf* structural relations. To avoid cluttering, edge weights are not shown. (The color version of this figure is available only in the electronic edition.)

widely used in the literature, including our previous works [75, 159, 160].[2] We also introduce a further measure that can be classified into the text-based category (like *go-rel*) and that compensates for the lack of use of Wikipedia as knowledge base in our previous studies.

We define a semantic relatedness measure that is based on the well-known Explicit Semantic Analysis (ESA) [58] method. ESA has shown to be very effective in computing semantic relatedness, and it's often regarded as current state-of-the-art. ESA relies on the intuition that the semantics of a text can be understood in terms of its affinity with respect to a predefined set of concepts that are explicitly defined and described by humans. For this purpose, ESA takes advantage of the human knowledge encoded in Wikipedia,[3] and hence Wikipedia articles are equated with concepts. Each Wikipedia concept is modeled as a tf-idf vector of words occurring in the corresponding article. An input text is represented as a weighted vector of Wikipedia concepts, where

[2] Other measures (e.g., [65, 166]) could certainly be used in alternative, however identifying the best-performing existing measure(s) is out-of-scope of this work.
[3] http://en.wikipedia.org

concepts are ranked based on their relevance to the text via a centroid-based text classification algorithm. Semantic relatedness between two input texts is finally computed by comparing their corresponding concept vectors using the cosine similarity measure.

However, unlike the other measures we employ in our study, ESA is designed to apply to pairs of individual words (or entire documents) only, and hence it cannot be directly used to disambiguate or rank the senses of an input word/text. In order to employ ESA in our setting, we define an ESA-based semantic relatedness measure which, for any two input word senses c_1, c_2, it first retrieves their corresponding WordNet glosses g_1, g_2 and then applies ESA to compare the two glosses; formally: $\textit{esa-rel}(c_1, c_2) = ESA(g_1, g_2)$.

3.3.1.3 Structural relation weighting schemes

To deal with multiple structural relations, we define *weighting schemes* (alternative to uniformly weighting) which, assuming the unavailability of user-specified requirements or prior knowledge, are based on characteristics of the input tree. One approach would rely on the assumption that the most frequent instances of a structural relation are the most important ones; this obviously implies that more complex (i.e., indirect) structural relations would be assigned with higher weights, since the frequency of occurrence is a non-decreasing function for increasing complexities. However, this approach might have the shortcoming of further penalizing the score propagation through graph edges that belong to simpler yet direct relations (e.g., *childOf*), which already have a lower support in the tree data instance. The opposite approach would hence assign higher weights to less frequently occurring relations, thus aiming to balance the properties of rarity (low support) and locality that a structural relation has in the tree when propagating the ranking score in the context graph. We hereinafter refer to the two weighting approaches as *support-aware* and *locality-aware* weighting schemes, respectively.

Given a structural relation $\tau \in \mathcal{T}$, if we denote with $n(\mathcal{D}, \tau)$ the number of edges in $\mathcal{D}$ of type τ, the support-aware weight of τ is defined as:

$$\omega_\tau{}^{(s)} = \frac{n(\mathcal{D}, \tau)}{\sum_{\tau' \in \mathcal{T}} n(\mathcal{D}, \tau')} \tag{3.3}$$

whereas the locality-aware weight of τ is defined as:

$$\omega_\tau{}^{(l)} = \frac{\sum_{\tau' \in \mathcal{T},\ \tau' \neq \tau} n(\mathcal{D}, \tau')}{(|\mathcal{T}| - 1) \sum_{\tau' \in \mathcal{T}} n(\mathcal{D}, \tau')} \tag{3.4}$$

Note that both the above definitions are such that $\sum_{\tau \in \mathcal{T}} \omega_\tau = 1$, which is a requirement in the application of the weighting scheme to the ranking models that will be presented next.

Algorithm 1 MSSPR

Input: tree data instance $\mathcal{D}$, set of structural relations $\mathcal{T}$, name of structural relation weighting scheme S, semantic relatedness measure *semrel*, damping factor α.

Output: the stationary distribution π.

$\mathcal{G} = \langle \mathcal{V}, \mathcal{T}, \mathcal{E}, w \rangle \leftarrow buildGraph(\mathcal{D}, \mathcal{T}, semrel)$ /* *cf. Sect. 3.3.1.1* */
if $(|\mathcal{T}| > 1)$ **then**
  $\omega_\tau \leftarrow buildWeightingScheme(S, \mathcal{D})$ /* *uniform or cf. Eqs. (3.3)–(3.4)* */
else
  $\omega_\tau \leftarrow 1$
end if
$it \leftarrow 0$
$r_i^{(it)} \leftarrow 1/|\mathcal{V}|$, **for all** $i \in \mathcal{V}$
repeat
  for all $i \in \mathcal{V}$ **do**
    $r_i^{(it+1)} = \alpha\left(\sum_{\tau \in \mathcal{T}} \omega_\tau \sum_{j \in B_\tau(i)} \frac{w(j,i)}{out_\tau(j)} r_j^{(it)}\right) + \frac{1-\alpha}{|\mathcal{V}|}$ /* *cf. Eq. (3.5)* */
  end for
  $it \leftarrow it + 1$
until *convergence*
$\pi \leftarrow r^{(it)}$

3.3.1.4 Multi-structure semantic PageRank

Our proposed ranking method, named *multi-structure semantic PageRank* (MSSPR), adapts a weighted PageRank formulation to deal with a multi-relational, edge-typed graph. Essentially, the underlying random-walk model is expressed by as many transition probability matrices as the different edge types. Given the ranking context graph $\mathcal{G}$ with structural relation set $\mathcal{T}$ and corresponding $|\mathcal{T}|$ weighting coefficients ω_τ, the ranking score of any concept c_i is computed as:

$$r_i = \alpha\left(\sum_{\tau \in \mathcal{T}} \omega_\tau \sum_{j \in B_\tau(i)} \frac{w(j,i)}{out_\tau(j)} r_j\right) + \frac{1-\alpha}{|\mathcal{V}|} \tag{3.5}$$

where $B_\tau(i)$ is the set of concepts that are linked to c_i through τ, $out_\tau(j)$ is the sum of weights on outgoing edges of type τ for c_j, and α is a damping factor ($\alpha \in [0,1]$, commonly set to 0.85). Equivalently, the matrix form of MSSPR is:

$$\mathbf{r} = \alpha\left(\sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}\right) + (1-\alpha)\mathbf{v} \tag{3.6}$$

where $\mathbf{v} = \frac{1}{|\mathcal{V}|}\mathbf{1}$ is the teleportation vector, and $\mathbf{S}_\tau$ denotes the column-stochastic transition probability matrix associated to the structural relation τ, i.e., only edges of type τ are considered in $\mathbf{S}_\tau$. Note that (3.6) can also be written as $\mathbf{r} = \alpha\mathbf{S}_\mathcal{T}\mathbf{r} + (1-\alpha)\mathbf{v}$, where $\mathbf{S}_\mathcal{T}$ is a convex combination of all the $\mathbf{S}_\tau$ matrices weighted by the corresponding ω_τ.

Upon the above MSSPR formulation, we introduce a variant into the definition of $\mathbf{v}$ to bias MSSPR according to the usage frequency of the concepts in $\mathcal{V}$. The rationale here is that a-priori importance of the concepts can be estimated based on their linguistic popularity as known from annotated text corpora, and hence the probability of moving to a concept-vertex c_i might be defined as proportional to its usage frequency. Formally, the ith element of the teleportation vector, for each $c_i \in \mathcal{V}$, is computed as: $v_i = (usage\_freq(c_i) + 1)/(\sum_{c \in \mathcal{V}} usage\_freq(c) + |\mathcal{V}|)$, where $usage\_freq(c)$ is the c's frequency of usage as stored in the reference lexical ontology, and the Laplace smoothing is introduced to handle unavailability of information about a concept's usage count. We will refer to the biased version of MSSPR as MSSPR-uf.

3.3.2 Alternative multi-relational methods

We devise two alternative approaches to structural sense ranking in tree data, whose common characteristic is a relaxation of the assumption of multidigraph definition of the context graph while maintaining the information on all selected types of tree structural relations. In particular, we raised two generic questions:

(Q-1) What if multiple instances of a basic PageRank model are separately built and performed over all structural relation types?

(Q-2) What if information on all structural relation types is used only to bias a single instance of a basic PageRank model?

3.3.2.1 Weighted combination of PageRank vectors

To answer question (Q-1), we develop the following method. Given the set $\mathcal{T}$ of selected structural relation types, MSSPR is performed for each of the types in $\mathcal{T}$. Upon convergence of all such $|\mathcal{T}|$ instances of MSSPR, the final ranking is obtained as a weighted linear combination of the PageRank stationary vectors p_τ:

$$p = \sum_{\tau \in \mathcal{T}} \omega_\tau p_\tau \tag{3.7}$$

where ω_τ ($\tau \in \mathcal{T}$) denote the coefficients of the structural relation weighting scheme. We will refer to this method as the pSSPR method; analogously, if (3.7) corresponds to MSSPR-uf instead, we will refer to the method as pSSPR-uf.

Interestingly, a relation between the rankings produced by MSSPR and pSSPR can be determined observing their behavior through the random walk convergence. In the following, we state a property on the conditioned equivalence between MSSPR and pSSPR.

Proposition 3.2. *For a given tree data instance $\mathcal{D}$ and a set $\mathcal{T}$ of selected structural relation types, the pSSPR vector is equivalent to the MSSPR vector at every iteration $n \geq 0$ if and only if each of the independent $|\mathcal{T}|$ MSSPR rankings is taken at the nth iteration.*

Algorithm 2 pSSPR

Input: tree data instance $\mathcal{D}$, set of structural relations $\mathcal{T}$, name of structural relation weighting scheme S, semantic relatedness measure *semrel*, damping factor α.

Output: the stationary distribution π.

1: **for all** $\tau \in \mathcal{T}$ **do**
2: $\quad \pi_\tau \leftarrow \mathsf{MSSPR}(\mathcal{D}, \{\tau\}, S, semrel, \alpha)$
3: **end for**
4: $\omega_\tau \leftarrow buildWeightingScheme(S, \mathcal{D})$ /*either uniform or cf. Eqs. (3.3)-(3.4)*/
5: $\pi \leftarrow \sum_{\tau \in \mathcal{T}} \omega_\tau \pi_\tau$

Proof. The "if" part of the proof is trivially by induction on the number of iterations. For the base case, take $n = 1$. Since the pure PageRank vectors, hereinafter denoted as $\mathbf{r}_{\mathsf{MSSPR}}$ and $\mathbf{r}_{\mathsf{pSSPR}}$, are initialized ($n = 0$) by the same way in both methods, we have:

$$\begin{aligned}
\mathbf{r}^{(1)}_{\mathsf{MSSPR}} &= \mathbf{r}^{(1)}_{\mathsf{pSSPR}} \Leftarrow \\
\Leftarrow \alpha \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(0)}_{\mathsf{MSSPR}} + (1-\alpha)\mathbf{v} &= \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{r}_\tau{}^{(0)}_{\mathsf{pSSPR}} \\
&= \sum_{\tau \in \mathcal{T}} \omega_\tau (\alpha \mathbf{S}_\tau \mathbf{r}^{(0)}_{\mathsf{pSSPR}} + (1-\alpha)\mathbf{v}) \\
&= \alpha \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(0)}_{\mathsf{pSSPR}} + (1-\alpha)\mathbf{v} \sum_{\tau \in \mathcal{T}} \omega_\tau \\
&= \alpha \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(0)}_{\mathsf{pSSPR}} + (1-\alpha)\mathbf{v} \Leftarrow \\
\Leftarrow \mathbf{r}^{(0)}_{\mathsf{MSSPR}} &= \mathbf{r}^{(0)}_{\mathsf{pSSPR}}
\end{aligned}$$

For the inductive step, suppose the hypothesis holds at the nth iteration, i.e., $\mathbf{r}^{(n)}_{\mathsf{MSSPR}} = \mathbf{r}^{(n)}_{\mathsf{pSSPR}}$, then at the $(n+1)$th iteration we have the same algebraic steps as for the base case:

$$\begin{aligned}
\mathbf{r}^{(n+1)}_{\mathsf{MSSPR}} &= \mathbf{r}^{(n+1)}_{\mathsf{pSSPR}} \Leftarrow \\
\Leftarrow \alpha \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(n)}_{\mathsf{MSSPR}} + (1-\alpha)\mathbf{v} &= \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{r}_\tau{}^{(n)}_{\mathsf{pSSPR}} \Leftarrow \\
\Leftarrow \mathbf{r}^{(n)}_{\mathsf{MSSPR}} &= \mathbf{r}^{(n)}_{\mathsf{pSSPR}}
\end{aligned}$$

The "only if" part of the proof is given by contradiction: assume there is an iteration $h > n$ for a certain $\tau_i \in \mathcal{T}$ such that

$$\mathbf{r}^{(n)}_{\mathsf{MSSPR}} = \mathbf{r}^{(h)}_{\mathsf{pSSPR}} \Leftarrow \tag{3.8}$$
$$\Leftarrow \alpha \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(n-1)}_{\mathsf{MSSPR}} + (1-\alpha)\mathbf{v} = \sum_{\tau \in \mathcal{T}, \tau \neq \tau_i} \omega_\tau \mathbf{r}_\tau {}^{(n-1)}_{\mathsf{pSSPR}} + \omega_{\tau i} \mathbf{r}^{(h-1)}_{\tau_i, \mathsf{pSSPR}}$$

If we take $h = n+1$, then we have:

$$\begin{aligned}
&\alpha \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(n-1)}_{\mathsf{MSSPR}} + (1-\alpha)\mathbf{v} = \\
&\quad = \alpha \sum_{\tau \in \mathcal{T}, \tau \neq \tau_i} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(n-1)}_{\mathsf{pSSPR}} + \alpha \omega_{\tau i} \mathbf{S}_{\tau i} \mathbf{r}^{(n)}_{\mathsf{pSSPR}} + (1-\alpha)\mathbf{v} \sum_{\tau \in \mathcal{T}} \omega_\tau \\
&\quad = \alpha \sum_{\tau \in \mathcal{T}, \tau \neq \tau_i} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(n-1)}_{\mathsf{pSSPR}} + \alpha \omega_{\tau i} \mathbf{S}_{\tau i}{}^2 \mathbf{r}^{(n-1)}_{\mathsf{pSSPR}} + (1-\alpha)\mathbf{v} \\
&\quad = \alpha \sum_{\tau \in \mathcal{T}} \omega_\tau \mathbf{S}_\tau \mathbf{r}^{(n-1)}_{\mathsf{pSSPR}} + \alpha \omega_{\tau i} \mathbf{S}_{\tau i} (\mathbf{S}_{\tau i} - 1) \mathbf{r}^{(n-1)}_{\mathsf{pSSPR}} + \omega_{\tau i}(1-\alpha)\mathbf{v} \\
&\quad \Leftarrow \mathbf{r}^{(n)}_{\mathsf{MSSPR}} = \mathbf{r}^{(n)}_{\mathsf{pSSPR}} + \omega_{\tau i}(\alpha \mathbf{S}_{\tau i}(\mathbf{S}_{\tau i} - 1)\mathbf{r}^{(n-1)}_{\mathsf{pSSPR}} + (1-\alpha)\mathbf{v})
\end{aligned}$$

Since there must be that $\mathbf{r}^{(n)}_{\mathsf{MSSPR}} = \mathbf{r}^{(n)}_{\mathsf{pSSPR}}$, the above equality obviously does not hold, which contradicts (3.8). This concludes the proof.

As we experimentally found (cf. Section 3.6), the above stated property does not hold in general. In fact, aside significant differences in time efficiency, the rankings of the two methods tend not to be equivalent in most cases.

3.3.2.2 Multi-structure aware personalized PageRank

To answer question (Q-2), we develop an adaptation of personalized PageRank, named mS-PPR, in which the bias in the ranking model relies on the tree structural relations of various types.

Let $\mathcal{G}_p = \langle \mathcal{V}_p, \mathcal{T}, \mathcal{E}_p, w_p \rangle$ be the ranking context graph with vertex set $\mathcal{V}_p$ coinciding with $\mathcal{V}$ of MSSPR, and edge set

$$\mathcal{E}_p = \{(c_i, c_j) \mid c_i \in \mathcal{C}(t), c_j \in \mathcal{C}(t'), t, t' \in T \ \wedge \ t' \rightarrow t\},$$

where $t' \rightarrow t$ means that t' is child of t. For each $(c_i, c_j) \in \mathcal{E}_p$, a weight $w_p(i,j)$ is computed to express the probability that any tag associated to c_i implies any tag associated to c_j through a direct structural relation in the tree; formally, $w_p(i,j) = \text{avg}_{t,t' \in T} \Pr(t, t') = \Pr(t \cap t') / \Pr(t) = freqPC(t,t')/freq(t)$ such that $c_i \in \mathcal{C}(t), c_j \in \mathcal{C}(t')$, where $freq(t)$ is the total number of occurrences of tag t in $\mathcal{D}$ (and $freqPC(t,t')$ is defined as for MSSPR). If we denote with $out_p(j)$ the sum of weights w_p on out-going edges of c_j, and with $R_\tau(i)$ the set of concept vertices that are pointed by c_i through edges of type τ, the mS-PPR score of any c_i is computed as:

$$r_i = \alpha \sum_{j \in B(i)} \frac{w_p(j,i)}{out_p(j)} r_j + (1-\alpha)\left(1 - \frac{\sum_{\tau \in \mathcal{T}} |R_\tau(i)|}{\sum_{h \in \mathcal{V}} \sum_{\tau \in \mathcal{T}} |R_\tau(h)|}\right) v_i \qquad (3.9)$$

with $v_i = 1/(|\mathcal{V}|-1)$ if $R_\tau(i) \neq \emptyset$, otherwise $v_i = 0$. The teleportation factor in (3.9) is defined to ensure that the teleportation matrix is stochastic, and that the probability of teleportation increases with smaller τ-specific out-neighbor sets.

3.3.3 Computational complexity aspects

We discuss here computational complexity aspects of the proposed multi-relational methods, broken down into two main phases, namely construction of the ranking context graph and execution of the ranking method.

Complexity of each of the proposed methods relies on the number and type of selected tree structural relations. For a given tree data instance $\mathcal{D}$, if we denote with $|\mathcal{D}|$ the size, with $\textsc{Max\_Depth}_{\mathcal{D}}$ the maximum depth, and with $\textsc{Max\_Branch}_{\mathcal{D}}$ the maximum branch (fan-out) of $\mathcal{D}$, we can express the costs C_τ of searching $\mathcal{D}$ for structural relations of type $\tau \in \mathcal{T}$ as follows: $\mathcal{O}(|\mathcal{D}|)$ for $\tau =$ *childOf*, $\mathcal{O}(|\mathcal{D}| \times \textsc{Max\_Depth}_{\mathcal{D}})$ for $\tau =$ *descOf*, $\mathcal{O}(|\mathcal{D}| \times \textsc{Max\_Branch}_{\mathcal{D}})$ for $\tau =$ *child*$|$*siblchildOf*, and $\mathcal{O}(|\mathcal{D}| \times \textsc{Max\_Depth}_{\mathcal{D}} \times \textsc{Max\_Branch}_{\mathcal{D}})$ for $\tau =$ *desc*$|$*sibldescOf*.

Let us denote with $\textsc{Max\_Poly}$ the maximum degree of polysemy of a term in the reference lexical ontology, which is an upper-bound for the maximum size of set $\mathcal{C}(t)$, with $t \in T(\mathcal{D})$. Also, let us denote with C_{semrel} the cost of computing the semantic relatedness for any pair of terms (tags), and with $\textsc{Max\_Tags}$ the maximum number of tags in $T(\mathcal{D})$ that share a concept (which in practical cases is a low constant, smaller than $|T(\mathcal{D})|$) such that computing function sf is $\mathrm{C}_{sf} = \mathcal{O}(|\mathcal{D}| \times \textsc{Max\_Tags}^2)$. The cost of building the context graph $\mathcal{G}$ in MSSPR is:

$$\mathrm{C}^{\mathcal{G}}_{\mathsf{MSSPR}} = \mathcal{O}(\sum_{\tau \in \mathcal{T}} \mathrm{C}_\tau \times \textsc{Max\_Poly}^2 \times \mathrm{C}_{semrel} \times \textsc{Max\_Tags}^2)$$

and the cost of a single iteration of MSSPR is $\mathcal{O}(\sum_{\tau \in \mathcal{T}} \mathrm{C}_\tau \times \textsc{Max\_Poly}^2)$. Similarly, the upper-bound in computing the $|\mathcal{T}|$ context graphs in pSSPR is:

$$\mathrm{C}^{\mathcal{G}}_{\mathsf{pSSPR}} = \mathcal{O}(\max_{\tau \in \mathcal{T}} \mathrm{C}_\tau \times \textsc{Max\_Poly}^2 \times \mathrm{C}_{semrel} \times \textsc{Max\_Tags}^2)$$

whereas the cost of a single iteration of pSSPR is the same as that of MSSPR.

As concerns mS-PPR, the number of edges in $\mathcal{G}_p$ is $\mathcal{O}(|\mathcal{D}| \times \textsc{Max\_Poly}^2)$ (since only *childOf* relation is considered), computing the edge weights is C_{sf}, and hence constructing $\mathcal{G}_p$ is $\mathcal{O}(|\mathcal{D}| \times \textsc{Max\_Poly}^2 \times \textsc{Max\_Tags}^2)$. Since the costs of computing sets $R_\tau(\cdot)$ in (3.9) follow the costs C_τ defined above, and hence running a single iteration of mS-PPR is $\mathcal{O}(\sum_{\tau \in \mathcal{T}} \mathrm{C}_\tau \times \textsc{Max\_Poly}^2)$.

Algorithm 3 Monte Carlo MSSPR

Input: Tree-structure-aware semantic multidigraph $\mathcal{G} = \langle \mathcal{V}, \mathcal{T}, \mathcal{E}, w \rangle$, distribution of structural relation weighting coefficients ω_τ over $\mathcal{T}$, probability α, number $m \geq 1$ of random walks to be started from each vertex in $\mathcal{G}$.

Output: An estimation $\hat{\pi}$ of the stationary distribution of MSSPR.

```
1: nVisited ← 0                                   /* total count of visited vertices */
2: visits[] ← {}                                  /* array of vertex visit counts */
3: for all i = 1..m do
4:    for all v ∈ V do
5:       nVisited ← MCRandomWalk(G, ω_τ, v, visits, α) + nVisited
6:    end for
7: end for
8: for all v ∈ V do
9:    π̂[v] ← visits[v]/nVisited
10: end for
```

3.4 Approximation of MSSPR based on Monte Carlo method

In this section we focus on how the computation of the rankings produced by MSSPR can be efficiently estimated, which is in principle helpful when dealing with *big data*, and the resulting ranking-context graphs can have very large size.

Approximation of PageRank algorithms can be addressed in different ways due to the many interpretations allowed by PageRank through expectations. A particularly appealing interpretation of PageRank is that it can be estimated as the average number of random walks visiting a given vertex at a given time provided that at each time, a walk can stop visiting with probability $(1 - \alpha)$. This interpretation has been shown to be conveniently modeled via absorbing Markov chains, such that the end-point of a random walk that starts from a random vertex and can be terminated at each step with probability $1 - \alpha$, is found as a sample from the stationary distribution of PageRank. After repeating the process many times, the estimate of the ranking score of a given vertex at convergence can be determined by aggregating partial information from the simulated walks.

In [12], the above intuition has been formalized as a series of probabilistic Monte Carlo type methods, which have a principled advantage (over the deterministic power iteration method) of an inherently parallel implementation, since the random walks simulated starting from different vertices can be considered as independent stochastic variables. Focusing on the most effective method, named Monte Carlo complete path stopping at dangling nodes, a simulated walk is supposed to be *short* since at each step it terminates with probability $1 - \alpha$ (i.e., a random walk has average length $1/(1 - \alpha)$), or when it reaches a vertex without out-neighbors (i.e., no teleportation is performed). After simulating m random walks from each vertex, the ranking score of a

Algorithm 4 MCRandomWalk - Monte Carlo random walk simulation

Input: Tree-structure-aware semantic multidigraph $\mathcal{G} = \langle \mathcal{V}, \mathcal{T}, \mathcal{E}, w \rangle$, distribution of structural relation weighting coefficients ω_τ over $\mathcal{T}$, start vertex v_0, array *visits* of vertex visit counts, probability α.
Output: integer $nVisited$.

```
1: nVisited ← 0, Q ← ∅
2: Q.insert(v_0)
3: repeat
4:    nVisited ← nVisited + 1
5:    v ← Q.remove()
6:    visits[v] ← visits[v] + 1
7:    rnd ← random(0,1)                    /* generate a random number in [0..1] */
8:    if (rnd ≥ 1 − α) and (R(v) ≠ ∅) then
9:       rnd ← random(0,1)
10:      cpr ← 0                                          /* cumulated probability */
11:      R ← R(v)                                  /* current vertex's out-neighbors */
12:      while (!R.isEmpty() and Q.isEmpty()) do
13:         u ← R.removeMin()
14:         for all e = (v,u,τ) ∈ E do
15:            pr(v,u) ← w(e) * ω_τ
16:            if (cpr < rnd < cpr + pr(v,u)) then
17:               Q.insert(u)                    /* the random walk moves to u */
18:               break
19:            end if
20:            cpr ← cpr + pr(v,u)
21:         end for
22:      end while
23:   end if
24: until (Q.isEmpty())
```

given vertex is estimated as the total number of visits to that vertex divided by the total number of visited vertices.

Algorithm 3 sketches our Monte Carlo type version of MSSPR, which extends the PageRank approximation in [12] to cope with a weighted, multi-relational PageRank. We provide a formulation of Monte Carlo MSSPR which exploits the property that the global connectivity matrix for a tree-structure-aware semantic multidigraph can be seen as a convex combination of the edge-type-specific transition probability matrices, each weighted by the corresponding ω_τ (cf. Section 3.3.1.4). As shown in Algorithm 4, our key idea is that, when a random walk started from a given vertex v continues through one of the out-neighbors u of v, the probabilistic move is not decided uniformly, rather it is biased by the strength of the edge between v and u, and also by the weight of the structural relation type corresponding to that edge (lines 14-17 in Algorithm 4). Note that however, this solution does not impact on the visiting order of the current vertex's out-neighbors; in effect, the choice at each step is in principle random, or it can be made deterministic by selecting

Table 3.1. Main notations used in this chapter and their descriptions.

notation	description	notation	description
$\mathcal{T}$	structural relation set	c	$\mathcal{T} = \{childOf\}$
u, l, s	uniform/locality-/support-aware weighting schemes	d	$\mathcal{T} = \{descOf\}$
ω_τ	weighting scheme coefficients	c,d	$\mathcal{T} = \{childOf, descOf\}$
MSSPR	Multi-Structure-aware Semantic PageRank	sc	$\mathcal{T} = \{child\|siblchildOf\}$
pSSPR	weighted combination of MSSPR solutions	c,sc	$\mathcal{T} = \{childOf, child\|siblchildOf\}$
mS-PPR	multi-Structure-aware Personalized PageRank	c,d,sc	$\mathcal{T} = \{childOf, descOf, child\|siblchildOf\}$
-uf	usage-frequency-based bias	sd	$\mathcal{T} = \{desc\|sibldescOf\}$
α	damping factor	sc,sd	$\mathcal{T} = \{child\|siblchildOf, desc\|sibldescOf\}$

the out-neighbor with lowest vertex-id (line 13). A more refined alternative could be sorting the out-neighbors by decreasing semantic relatedness with the current vertex (edge weight $w(e)$), regardless of the edge-type weight ω_τ.

A natural question arises on how to choose the number m of iterations of the algorithm. In this regard, the study in [12] gives evidence of the fact that setting $m = 1$ can be deemed as "sufficient" in practice, and that after m iterations, the relative error of the Monte Carlo method will reduce on average only by a factor $1/\sqrt{m}$. Concerning the computational complexity of Monte Carlo MSSPR, we acknowledge the cost $\mathcal{O}(m|\mathcal{V}|/(1-\alpha))$ of the original Monte Carlo type algorithm for the basic PageRank [12]. However, in our multi-relational ranking setting, the presence of multiple structural relation types (which may lead to multiple edges between the same pair of vertices) should not be discarded. Therefore, a factor proportional to the average (out-)degree of vertices, although generally not of the same order of magnitude of the vertex set size, would refine the above cost expression, while not changing the asymptotic behavior of the algorithm.

3.5 Experimental evaluation

3.5.1 Data

We used a dataset of 1,289,309 XML documents, which was originally used in [75, 159]. This dataset is part of the official INEX 2009 collection,[4] a corpus of semantically annotated Wikipedia articles, which perfectly fits our evaluation needs due to its semantic and structural heterogeneity. A tag is coupled with two attributes: *wordnetid*, whose value corresponds to a unique sense id in WordNet 3.0, and *confidence*, whose value (typically within 0.6 and 1) expresses the confidence the annotator originally had in assigning that wordnetid to the tag.

The document trees in our dataset contain 159,094,497 tags (5,203 distinct), with average (resp. maximum) depth of 9.01 (resp. 74) and average (resp. maximum) fan-out of 1.36 (resp. 4,643). The average polysemy of the tags is 2.45, which increases to 4.01 if monosemous tags are excluded. We

[4] http://www.mpi-inf.mpg.de/departments/d5/software/inex/

Table 3.2. Statistics on the tree-structure-aware semantic multidigraphs for the heterogeneous evaluation case.

	c	*d*	*sc*	*sd*
# vertices	11,477	11,477	11,477	11,477
# edges	4,249,867	5,982,983	10,925,758	12,747,330
avg in-degree	370.29	521.30	951.97	1110.68
avg path length	2.39	2.26	2.19	2.10
diameter	8	8	8	7
clustering coefficient	0.36	0.39	0.50	0.50
# strongly CCs	129	196	105	141
# vertices in the largest CC	11,349	11,282	11,373	11,337
# edges in the largest CC	4,247,704	5,979,251	10,915,969	12,738,020

Table 3.3. Statistics on the tree-structure-aware semantic multidigraphs for the homogeneous evaluation case. Values correspond to averages (and standard deviations in brackets).

	c	*d*	*sc*	*sd*
# vertices	94.62 (90.60)	96.46 (90.17)	94.20 (90.75)	97.99 (90.86)
# edges	403.07 (531.16)	403.79 (548.59)	1,225.10 (3,625.10)	1,131.28 (3,303.20)
avg in-degree	3.57 (1.60)	3.49 (1.61)	7.60 (6.62)	6.88 (5.89)
avg path length	3.64 (1.48)	2.55 (1.09)	3.29 (1.22)	2.46 (0.99)
diameter	9.25 (4.60)	6.07 (3.29)	8.68 (4.27)	6.04 (3.19)
clustering coefficient	0.06 (0.10)	0.05 (0.09)	0.16 (0.18)	0.11 (0.13)
# strongly CCs	70.01 (59.54)	82.06 (72.51)	50.09 (41.49)	70.81 (58.93)

processed the articles to keep only the structure information, so to obtain trees of tags, rooted in `article`. Note that tags in the INEX 2009 collection were already provided as normalized to match WordNet entries. As for the text-based relatedness approach (i.e., *go-rel* and *esa-rel* measures), the tags' synset glosses were subject to standard text preprocessing operations, such as removal of stopwords and word stemming.

In order to build the ranking context graphs, we treated the tree data instances in two different ways: either all the article document trees are considered separately or they are merged into a single huge tree (rooted in a fictitious tag node `articles`). We will refer to the above situations as *homogeneous evaluation* case and *heterogeneous evaluation* case, respectively. Note that by considering such two complementary situations, we could assess our approach on different conditions concerning the domain-specificity of the vocabulary of tags. In this regard, we expect that a relatively conceptual homogeneity of the tags in a tree would justify the use of structural contexts that rely on more complex relations; conversely, for a tree covering a larger variety of topics (i.e., tag labels), building the context graph over (directly) related tags would reduce the disambiguation "noise" which might be produced by more complex structural contexts.

3.5.2 Tree-structure-aware semantic multidigraphs

We analyzed the topological properties of the ranking context graphs for MSSPR by varying the type of structural relation.[5] Table 3.2 refers to the heterogeneous evaluation case, whereas Table 3.3 refers to the homogeneous evaluation case; in this case, all statistics were averaged over the graphs corresponding to the single document trees in the INEX collection. Note that, to avoid cluttering the presentation in the result tables, we use hereinafter abbreviated notations for the selected structural relations, as reported in the right-hand side of Table 3.1.

Table 3.2 shows that some statistics may vary significantly with the structural relation type. The average (in-)degree of nodes evidently increases with the complexity of structural relation, but it is rather high even with *c*. Moreover, as expected, from *c* (resp. *d*) to *sc* (resp. *sd*) the number of strongly connected components decreases, whereas the clustering coefficient increases. Note that however the clustering coefficient is generally high regardless of the structural relation type, which means that vertices in the graph tend to form tightly connected, localized cliques with their immediate neighbors. Therefore, the relatively high clustering coefficients combined with the low average path length (between 2.1 and 2.4) and network diameter (7-8) in our data, is a strong indication that the small-world network hypothesis holds for the structural semantic network setting.

The above remark is also confirmed in the homogeneous evaluation case (Table 3.3), although all statistics are subject to significant variations due to the large structural and semantic variety of document trees that compose the INEX collection. Furthermore, the clustering coefficient values are quite lower and the average path length values are higher than in the heterogeneous case. This was expected since in a huge document tree (i.e., heterogeneous case), the concept vertices that are linked to one concept vertex are more likely to be linked to each other, i.e., the clustering coefficient is higher.

It should be noted that in both evaluation cases, and regardless of the structural relation type, the average path length is rather low (always below 4). Moreover, in the homogeneous case, it may significantly vary over the input document trees and w.r.t. the structural relation type. This raises a natural question whether and to what extent properties related to the average path length in the network can influence the global ranking mechanism. In Section 3.6.1.2, we will attempt an answer to this question showing how information on the average path length can be exploited to estimate the damping factor in our proposed ranking methods.

3.5.3 Evaluation methodology

The INEX 2009 collection lends itself particularly well to the generation of a gold standard for the structural sense ranking task, thanks to the availability

[5] We used the igraph package for the R environment (http://igraph.sourceforge.net/).

of explicit indicators on the meaning assigned by a human annotator to each occurrence of a tag, along with the amount of trust expressed by the annotator in disambiguating the tag. In order to generate a reference ranking for our evaluation dataset, we followed the methodology proposed in [159]. Each tag is associated with a vector of probabilities, which has length equal to the number of possible senses for that tag, such that positions in the tag's vector correspond to the tag's sense numbers. Each of the tag's senses that has been annotated in the document instance is assigned a score in the vector which is proportional to the confidence value(s) of its annotation(s); each of the possibly remaining senses without annotation in the document instance is assigned a score in the vector which is inversely proportional to its sense number—the rationale is that word senses are usually ordered according to an estimation of their common usage of frequency. We refer the interested reader to [159] for a detailed description.

To assess the effectiveness of the proposed methods, we used criteria that are standard in ranking tasks: *normalized discounted cumulative gain* (*nDCG*), *Binary preference function* (*Bpref*), and *Fagin's intersection metric* (F)(cf. Chapter 2, Section 2.7); for each of them, the higher the score the better the ranking evaluation. Such criteria needed to be adapted to our setting, as described next.

As regards *nDCG*, we first compute a tag-based $nDCG_t^{(k)}$ taking into account the sense ordering over the top-k senses of a single tag (i.e., comparing the ordering of the tag-senses in the reference ranking with the one given by the algorithm's scores). Hence we obtain the final $nDCG$ as the average of the $nDCG_t^{(k)}$ computed over all tags t.

We defined two variants of the Fagin measure F, henceforth denoted as F_1 and F_2. In F_1, the actual reference ranking is obtained by simply sorting all scores in the original reference ranking, whereas the algorithm's ranking scores are first normalized by tag (to resemble the tag-specific probability distributions in the original reference ranking), and then sorted. In F_2, for both the reference and algorithm's rankings, each concept's score is multiplied by the logarithm of the number of senses of the unique tag associated to the concept (recall that a concept is treated as a pair tag-IDsense, i.e., a synset in WordNet).

As concerns *Bpref*, we used as queries the root-to-leaf tag-paths in $\mathcal{D}$, judging the top-1 ranked senses of each tag in the path as relevant candidates, and all the other senses of these tags as not relevant. The overall $Bpref$ score was obtained as a weighted average over the tag-path $Bpref$ scores weighted by the number of occurrences of a particular path.

3.6 Results

We discuss our experimental evaluation in terms of effectiveness and efficiency.[6] It should be noted that the methods that apply to graphs built on singleton sets of structural relations (i.e., *c*, *d*, *sc*, and *sd*) actually play the role of competitors (cf. Section 4.3) against our proposed multi-relational setting. Note also that our definition of structural contexts reported in Table 3.1 is not a complete lattice over the selected types of structural relations, which is explained by our intention not to further increase the complexity of the evaluation framework.

3.6.1 Effectiveness

We present performance results obtained by our proposed methods, distinguishing between the heterogeneous and homogeneous evaluation cases (cf. Section 3.5.1); in the latter case, results correspond to averages over the individual trees. Note that we decided not to include monosemous tags in the ranking evaluation in order to avoid a bias in the result presentation. Parameter k of $nDCG$ was set equal to 3, as this value is close to the average polysemy of the input data (cf. Section 3.5.1). Parameter k of F was set equal to 5000, as this value allows taking into account a reasonably large portion of the global rankings produced by the methods—about 10% of the size of the vertex set in a ranking context graph.

3.6.1.1 Evaluation of MSSPR, pSSPR, and mS-PPR

We organize our discussion on the evaluation of effectiveness of the proposed MSSPR and alternative ranking methods according to three main aspects: composite (multi-typed) vs. singleton structural contexts, heterogeneous vs. homogeneous evaluation case, and biased versions of the MSSPR and pSSPR methods. We will elaborate on each of these aspects in the following paragraphs.

Multi-typed vs. singleton structural contexts

A major result of our study was to confirm that the performance improvement hypothesis actually holds when the structural context for the ranking algorithm consists of multiple tree relations. This was generally observed for all algorithms and evaluation cases. In Table 3.4, the best-performing results by MSSPR correspond to the structural contexts *c,d* followed by *c,sc* or *c,d,sc*, for all criteria with the exception of *Bpref* (for which the best scores were achieved for *sc,sd*). In the homogeneous evaluation case (Table 3.5), *c,sc* followed by *c,d,sc* or *c,d* were prevalent on the other structural contexts. Interestingly, in the homogeneous case, the performance gains corresponding

[6] Experiments were carried out on an Intel Core i7-3960X CPU @ 3.30GHz, 64GB RAM machine.

Table 3.4. Performance of MSSPR methods: heterogeneous evaluation case.

		MSSPR				MSSPR-uf			
$\mathcal{T}$	ω_τ	$nDCG$	$Bpref$	F_1	F_2	$nDCG$	$Bpref$	F_1	F_2
c	–	0.935	0.266	0.402	0.204	0.938	0.283	0.443	0.243
d	–	0.937	0.284	0.405	0.205	0.939	0.296	0.447	0.244
c,d	u	**0.938**	0.277	0.408	0.209	**0.940**	0.292	**0.449**	**0.247**
	l	**0.938**	0.279	**0.409**	0.209	**0.940**	0.293	**0.449**	**0.247**
	s	0.937	0.277	0.408	0.208	0.938	0.292	**0.449**	0.246
sc	–	0.936	0.288	0.398	0.207	0.938	0.291	0.428	0.242
c,sc	u	0.937	0.277	0.400	0.209	0.939	0.292	0.431	0.244
	l	0.937	0.278	0.401	0.209	**0.940**	0.292	0.431	0.244
	s	0.936	0.277	0.400	0.209	0.939	0.292	0.431	0.243
c,d,sc	u	0.937	0.283	0.406	0.208	0.939	0.296	0.438	0.244
	l	0.937	0.283	0.406	0.208	**0.940**	0.296	0.438	0.244
	s	0.937	0.283	0.405	0.208	0.937	0.296	0.438	0.244
sd	–	0.936	**0.294**	0.402	0.209	0.938	0.311	0.429	0.242
sc,sd	u	0.937	0.291	0.403	**0.210**	0.939	**0.313**	0.431	0.243
	l	0.937	0.292	0.402	**0.210**	0.939	0.311	0.430	0.243
	s	0.937	0.291	0.403	0.209	0.939	**0.313**	0.431	0.243

Results correspond to best performance over the various semantic relatedness measures. Bold values refer to the best scores per assessment criterion.

Table 3.5. Performance of MSSPR methods: homogeneous evaluation case.

		MSSPR				MSSPR-uf			
$\mathcal{T}$	ω_τ	$nDCG$	$Bpref$	F_1	F_2	$nDCG$	$Bpref$	F_1	F_2
c	–	0.806	0.570	0.460	0.316	0.916	0.627	0.558	0.377
d	–	0.761	0.511	0.445	0.296	0.907	0.544	0.546	0.351
c,d	u	0.810	0.572	0.458	0.319	0.917	0.627	0.559	0.378
	l	0.810	0.580	0.460	0.324	0.921	0.625	0.558	0.379
	s	0.810	0.574	0.458	0.320	0.917	**0.629**	0.559	0.377
sc	–	0.789	0.557	0.469	0.330	0.922	0.587	0.564	0.372
c,sc	u	0.807	0.571	0.469	0.328	0.916	0.620	0.566	0.376
	l	0.806	**0.585**	**0.471**	**0.334**	0.922	0.624	**0.567**	**0.380**
	s	0.805	0.560	0.468	0.326	0.915	0.609	0.565	0.371
c,d,sc	u	**0.812**	0.569	0.462	0.325	0.921	0.622	0.562	0.379
	l	**0.812**	0.579	0.461	0.329	**0.923**	0.623	0.560	**0.380**
	s	0.811	0.560	0.463	0.326	0.915	0.612	0.563	0.374
sd	–	0.697	0.490	0.443	0.300	0.909	0.511	0.541	0.349
sc,sd	u	0.795	0.551	0.459	0.324	0.922	0.587	0.556	0.375
	l	0.794	0.546	0.453	0.320	0.921	0.582	0.554	0.371
	s	0.806	0.554	0.460	0.326	0.921	0.592	0.560	0.375

Results correspond to best performance over the various semantic relatedness measures. Bold values refer to the best scores per assessment criterion.

to multi-typed structural contexts were more evident. Analyzing the average performance over the various semantic relatedness measures also confirmed the benefits deriving from using composite contexts in MSSPR; Fig. 3.2(a)–(b) show how the $nDCG$ scores achieved by MSSPR on multi-typed contexts were consistently higher than those obtained on single-type contexts.

The pSSPR approach (Tables 3.6–3.7) performed comparably to MSSPR on multi-typed structural contexts in terms of $Bpref$ and Fs. However, $nDCG$ values were significantly lower (e.g., gaps of the order of 1.0E-1); furthermore, as shown in Fig. 3.2(a), average $nDCG$ scores were even lower than those obtained for single-typed structural contexts in the heterogeneous case—we further investigated this point and found that it did not depend on using a

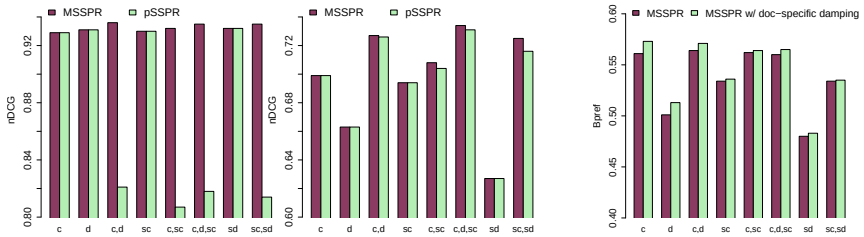


Fig. 3.2. Average performance over semantic relatedness measures: MSSPR vs. pSSPR on (a) heterogeneous evaluation case and (b) homogeneous evaluation case, using $nDCG$; (c) MSSPR vs. MSSPR with document-specific damping factor (homogeneous evaluation case), using $Bpref$.

particular semantic relatedness measure. Therefore, it should be emphasized that the multi-relational approach adopted by MSSPR might be considered as a preferred method w.r.t. the combination of single-type structure based ranking solutions, which is peculiar of pSSPR.

Concerning mS-PPR (Table 3.8), there is a less clear evidence of the benefits that can derive from using multi-typed structural contexts. However, this method was generally outperformed by MSSPR and pSSPR (even by the non-biased MSSPR in most cases) for all criteria, with gaps of the order of 3.0E-1 $nDCG$, 6.0E-2 $Bpref$, 5.0E-2 F_1 and 8.0E-2 F_2.

We also evaluated the ranking performance of the MSSPR methods when only the *semrel* term would be considered in the edge-weighting function, i.e., $sf = 1$ for all edges. We observed a general slight decrease in the average performance, of the order of 1.0E-3 or above on each assessment criterion, which indicates that the sf term in the edge-weighting function serves for the purpose of weighing the impact of the repetition of substructures across the input tree instance.

Heterogeneous vs. homogeneous evaluation case

The two evaluation cases turned out to be different scenarios for our methods. In general, the higher cohesiveness of the tags in the input data enables a ranking method equipped with a multi-typed structural context to significantly improve upon the performance corresponding to each of its subsets of structural relations. As a result, in the homogeneous case the performance improvement due to a multi-typed structural context was more evident, and also scores were generally higher than in the heterogeneous case.

The weighting schemes impacted differently over the various criteria in the heterogeneous case, for both MSSPR and MSSPR-uf, with s never improving upon u and, except for some cases corresponding to *sc,sd*, upon l (gaps around 2.0E-3). However, in the homogeneous case, relative differences among the weighting schemes were more consistent over the criteria. As we expected,

Table 3.6. Performance of pSSPR methods: heterogeneous evaluation case.

		pSSPR				pSSPR-uf			
$\mathcal{T}$	ω_τ	*nDCG*	*Bpref*	F_1	F_2	*nDCG*	*Bpref*	F_1	F_2
c,d	*u*	**0.844**	0.278	**0.407**	**0.210**	**0.898**	0.292	**0.449**	**0.247**
	l	**0.844**	0.278	**0.407**	**0.210**	**0.898**	0.292	**0.449**	**0.247**
	s	**0.844**	0.278	**0.407**	**0.210**	**0.898**	0.292	**0.449**	**0.247**
c,sc	*u*	0.827	0.276	0.402	0.208	0.892	0.290	0.430	0.243
	l	0.826	0.278	0.402	**0.210**	0.892	0.290	0.430	0.245
	s	0.818	0.287	0.398	0.208	0.884	0.292	0.427	0.242
c,d,sc	*u*	0.835	0.282	0.406	0.207	0.895	0.294	0.438	0.243
	l	0.834	0.282	**0.407**	0.208	0.894	0.293	0.438	0.245
	s	0.821	0.287	0.399	0.207	0.884	0.295	0.429	0.242
sc,sd	*u*	0.822	**0.290**	0.403	0.209	0.884	**0.308**	0.431	0.242
	l	0.822	**0.290**	0.403	0.209	0.884	0.307	0.432	0.242
	s	0.822	**0.290**	0.403	0.209	0.884	0.307	0.431	0.242

Results correspond to best performance over the various semantic relatedness measures. Bold values refer to the best scores per assessment criterion. Results corresponding to singleton sets $\mathcal{T}$ are the same as in Table 3.4, hence are not reported.

Table 3.7. Performance of pSSPR methods: homogeneous evaluation case.

		pSSPR				pSSPR-uf			
$\mathcal{T}$	ω_τ	*nDCG*	*Bpref*	F_1	F_2	*nDCG*	*Bpref*	F_1	F_2
c,d	*u*	0.810	0.562	0.460	0.313	0.917	0.615	0.558	0.370
	l	0.808	0.567	0.460	0.317	0.921	0.611	0.557	0.372
	s	**0.811**	0.561	0.460	0.314	0.917	0.616	0.558	0.369
c,sc	*u*	0.807	**0.570**	0.470	0.326	0.917	0.622	0.565	0.375
	l	0.804	0.582	**0.471**	**0.332**	0.922	**0.623**	**0.566**	**0.380**
	s	0.804	0.556	0.468	0.324	0.914	0.608	0.564	0.370
c,d,sc	*u*	**0.811**	0.563	0.463	0.320	**0.923**	0.615	0.561	0.376
	l	0.810	**0.570**	0.462	0.324	**0.923**	0.613	0.560	0.375
	s	0.809	0.554	0.464	0.322	0.915	0.607	0.561	0.370
sc,sd	*u*	0.794	0.545	0.460	0.318	0.922	0.578	0.556	0.368
	l	0.792	0.536	0.454	0.313	0.920	0.572	0.554	0.364
	s	0.804	0.546	0.461	0.319	0.920	0.582	0.559	0.369

Results correspond to best performance over the various semantic relatedness measures. Bold values refer to the best scores per assessment criterion. Results corresponding to singleton sets $\mathcal{T}$ are the same as in Table 3.5, hence are not reported.

scheme *l* mostly led to better performance than *s* and *u* for contexts that involve the *c* relation.

As concerns the impact of the semantic relatedness measures, in the heterogeneous case similar performances were generally achieved by *go-rel* and *esa-rel*, with the latter leading to slightly better F_2 and *Bpref* values. Different behavior was observed in terms of *nDCG*, where *esa-rel* scores were far from the best (e.g., of the order of 1.0E-1, for MSSPR), and differences were scarcely significant (e.g., 2.0E-4, for MSSPR) between *go-rel* and *p-rel*. The latter turned out to be the best-performing measure in terms of *nDCG* and *Bpref*. All measures led to very close F_1 scores, with *go-rel* often obtaining better results (of the order of 1.0E-3). In the homogeneous case, comparison situation was much simpler, since *p-rel* always led to better performance than the other measures, with gains around 2.0E-1 in terms of *nDCG* and lower in the other cases (e.g., of the order of 4.0E-2). This predominance of *p-rel* can be explained by its total coverage over the relations, that allows

Table 3.8. Performance of mS-PPR.

$\mathcal{T}$	*heterogeneous evaluation*				*homogeneous evaluation*			
	nDCG	*Bpref*	F_1	F_2	*nDCG*	*Bpref*	F_1	F_2
c	0.634	0.238	0.390	0.188	0.443	0.562	0.387	0.285
d	**0.638**	0.231	0.377	**0.194**	0.449	0.542	0.392	0.285
c,d	0.637	**0.239**	**0.394**	**0.194**	**0.451**	**0.566**	**0.406**	**0.291**
sc	0.635	0.232	0.391	0.190	0.447	0.560	0.402	0.279
c,sc	0.636	0.235	**0.394**	0.192	0.448	0.563	0.404	0.280
c,d,sc	0.637	0.237	0.392	0.193	0.450	0.564	0.405	0.281
sd	0.635	0.231	0.377	0.193	0.450	0.540	0.404	0.288
sc,sd	**0.638**	0.232	0.393	0.192	0.449	0.561	0.401	0.283

Bold values refer to the best scores per assessment criterion.

discriminating among different synsets also with tag-sets of relatively small size. Performances of *go-rel* and *esa-rel* were always very similar in the homogeneous case, with the latter performing slightly better.

Usage-frequency-based bias

MSSPR-uf and pSSPR-uf outperformed MSSPR and pSSPR, respectively, in terms of all criteria and evaluation cases. Performance gaps between MSSPR-uf and MSSPR in the heterogeneous (resp. homogeneous) case were up to 1.9E-3 (resp. 1.0E-1) $nDCG$, 1.2E-2 (resp. 2.0E-2) $Bpref$, 2.6E-2 (resp. 9.3E-2) F_1, and 3.1E-2 (resp. 4.2E-2) F_2. This supports our expectation that exploiting information on the concepts' usage frequency is beneficial to the ranking performance. Moreover, in both MSSPR-uf and pSSPR-uf, the effect of usage-frequency-based bias led to subtle differences among the performance scores corresponding to the various semantic relatedness measures.

3.6.1.2 Impact of damping factor on the MSSPR performance

In our proposed methods, the damping factor α is chosen to be 0.85, in analogy with the default setting of the parameter in the original PageRank algorithm. Recall that this finds an explanation 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 Section 3.5.2, we have however observed that the concept vertices in a semantic multidigraph tend to connect to each other following shorter paths of average length which is always below 4. This prompted us to conjecture that a different setting of α, i.e., a value lower than 0.85, might be more appropriate for the structural sense ranking context. In order to assess our hypothesis, we focused on the homogeneous evaluation case, for which the semantic graphs corresponding to the input document trees exhibit high variability in their structural properties, including average path length (cf. Table 3.3). Therefore, rather than fixing a unique value of α for the ranking method to apply over all document graphs, we decided to set α specifically for each document graph as $\alpha = 1 - (1/apl)$, being apl the average path length of the particular document graph.

Table 3.9. Performance of MSSPR methods with document-specific damping factor (homogeneous evaluation case).

$\mathcal{T}$	MSSPR				MSSPR-uf			
	nDCG	*Bpref*	F_1	F_2	*nDCG*	*Bpref*	F_1	F_2
c	0.804	0.579	0.453	0.318	0.922	0.628	0.570	0.398
d	0.753	0.525	0.439	0.291	0.918	0.549	0.568	0.386
c,d	0.809	**0.587**	0.461	0.320	**0.926**	**0.632**	0.571	**0.399**
sc	0.789	0.560	0.463	0.327	0.924	0.593	0.573	0.387
c,sc	0.805	0.585	**0.464**	**0.329**	0.925	0.621	**0.576**	0.395
c,d,sc	**0.812**	0.585	0.457	0.326	**0.926**	0.626	0.572	0.398
sd	0.697	0.493	0.438	0.291	0.912	0.516	0.565	0.379
sc,sd	0.795	0.553	0.453	0.321	0.924	0.593	0.567	0.390

Results correspond to best performance over the various semantic relatedness measures, and refer to the uniform weighting scheme. Bold values refer to the best scores per assessment criterion.

Table 3.9 reports on the performance scores achieved by MSSPR methods equipped with document-specific damping factor. It is noticeable that the best-performing results again correspond to multi-typed structural contexts, in all cases. Comparing Table 3.9 with Table 3.5, the following maximum gains were obtained between the respective best scores on composite structural contexts (and ω_τ fixed to u): 9.0E-3 on $nDCG$, 1.5E-2 on $Bpref$, 1.0E-2 on F_1, 1.9E-2 on F_2. Overall, when equipped with document-specific damping factor, MSSPR behaved as good as or, for $Bpref$ (Fig. 3.2(c)) and on some composite structural contexts for F_1 and F_2, better than MSSPR with α fixed to 0.85. MSSPR-uf consistently outperformed its counterpart with α fixed to 0.85.

3.6.1.3 Evaluation of Monte Carlo MSSPR

In Table 3.10, we summarize effectiveness results achieved by our probabilistic Monte Carlo version of MSSPR. For this session of experiments, we referred to the heterogeneous evaluation case, for which we set the structural relation weighting scheme as uniform, and the probability α to 0.85. We varied the number m of iterations from one to ten—given the small fluctuations on the results produced, in the table we present only results obtained for $m = 1$ and $m = 10$.

From the table, it can be seen that better results tend to be in favor of complex structural relation sets, which once again justifies our multi-relational approach to the structural sense ranking problem, even in a probabilistic approximation setting. Monte Carlo MSSPR achieved moderately high values of $nDCG$ (up to around 0.75 for $m = 1$ and 0.79 for $m = 10$) which, compared to the results shown in Table 3.4, corresponds to a gap of at least 0.19 (in terms of best scores). On the other hand, Monte Carlo MSSPR performed much closely to the deterministic MSSPR according to the other assessment criteria, with gaps equal to or less than 2.0E-2 F_1, 1.0E-2 F_2, and 5.0E-3 $Bpref$. As expected, increasing the number m of iterations was only marginally beneficial to the performance of Monte Carlo MSSPR, as it led to scores that

Table 3.10. Performance of Monte Carlo MSSPR (heterogeneous evaluation case).

$\mathcal{T}$	$m=1$				$m=10$			
	nDCG	*Bpref*	F_1	F_2	*nDCG*	*Bpref*	F_1	F_2
c	0.733	0.235	0.385	0.191	0.781	0.236	0.395	0.194
d	0.733	0.249	0.384	0.191	0.779	0.250	0.396	0.194
c,d	0.739	0.241	**0.386**	**0.193**	0.789	0.246	**0.400**	**0.195**
sc	0.741	0.257	0.382	0.192	0.781	0.260	0.390	0.194
c,sc	0.739	0.249	0.380	0.191	0.788	0.252	0.393	0.193
c,d,sc	0.743	0.251	0.383	**0.193**	**0.790**	0.251	0.396	**0.195**
sd	0.741	0.258	0.383	0.192	0.782	**0.262**	0.394	0.192
sc,sd	**0.748**	**0.259**	0.385	0.191	0.785	0.261	0.391	0.192

Results are averaged over fifty runs. Bold values refer to the best scores per assessment criterion.

for each of the criteria corresponded to small improvements w.r.t. the single-iteration case. Instead, it came to our surprise to find out that the vertex selection scheme for the random walk simulation based on an ordering of the out-neighbors by decreasing semantic relatedness (cf. Section 3.4), did not significantly impact on the quality of approximations, with increments that were on average of the order of 1.0E-3 for each of the assessment criteria.

As a final remark, it was interesting to investigate whether the relatively good behavior of Monte Carlo MSSPR as approximation of MSSPR also holds under a crisp evaluation in terms of accuracy, i.e., proportion of correctly disambiguated tags (top-1 ranked senses). We calculated accuracy over all structural contexts and semantic relatedness measures, with uniform weighting scheme, and by varying m as in the previous evaluation. On median (resp. average), the minimum gap of Monte Carlo MSSPR from MSSPR was 0.052 (resp. 0.055). Considering the best accuracy scores, the minimum gap of Monte Carlo MSSPR was 0.074 w.r.t. MSSPR, and about 0.1 w.r.t. MSSPR-uf. Therefore, we can fairly conclude that the accuracy evaluation still confirms the usefulness of Monte Carlo MSSPR as an approximation of MSSPR.

3.6.2 Efficiency

We analyzed the time performances of MSSPR and pSSPR, divided in graph building times and ranking times for each of the methods, as shown in Fig. 3.3(a); results were averaged over the semantic relatedness measures and corresponded to the heterogeneous evaluation case. It can be noted that the graph building times in MSSPR increased for increasing structural complexity of the corresponding sets $\mathcal{T}$, i.e., for increasing size of the edge set. Moreover, graph building time was generally slightly higher when using *esa-rel* (e.g., about ten seconds more than time corresponding to *go-rel*), which might be due to a slower access time to the Wikipedia database w.r.t. WordNet.[7] By contrast, variations in the ranking times were nearly negligible, with average

[7] We used the full English Wikipedia dump in XML available at http://dumps.wikimedia.org/enwiki/ (last accessed September 4, 2013).

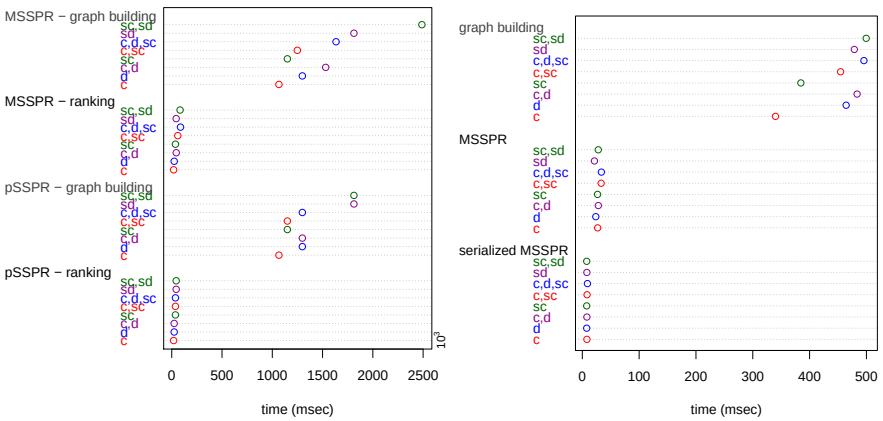


Fig. 3.3. Time performances: (a) MSSPR vs. pSSPR, and (b) MSSPR vs. serialized-MSSPR, on a DAG version of the evaluation dataset.

rate of convergence ranging from 25 iterations for *p-rel* and *ic-rel* to 60 iterations for *go-rel*. In Fig. 3.3(a), we also reported the pSSPR time performance under a "parallel" runtime configuration, i.e., in which the maximum runtimes per structural context were considered. The comparison between MSSPR and pSSPR was clearly in favor of the latter, albeit both methods' runtimes were of the same order of magnitude.

Concerning mS-PPR, the time required for building the context graph in mS-PPR was comparable to the MSSPR graph time with relation c (especially when *p-rel* or *ic-rel* measures were used); this was expected since the graph for mS-PPR considers only relation c, being independent on the choice of $\mathcal{T}$. However, the ranking time of mS-PPR, which instead depends on $\mathcal{T}$, was always slower than the the ranking times of MSSPR methods on the corresponding set $\mathcal{T}$, usually about one order of magnitude.

3.6.2.1 Optimization of MSSPR on DAGs

We investigated the benefits obtained by properly *serializing* the MSSPR calculation for application to a directed acyclic graph (DAG).

As previously mentioned in Section 4.3, we based this evaluation by following the lead of the serialization of the ObjectRank method described in [15]. Recall that the partitioning of a graph into subsets $L_1, \ldots, L_q$, and hence the serialized calculation, is possible if and only if the graph is DAG. In our setting, it is supposed that the structural semantic graph is never DAG in practical cases, since not only the same tag can occur multiple times in the same tree but also cycles can be formed by concepts that are shared among different tags. Nevertheless, we tested a serialized form of MSSPR on a DAG-aware version of the input dataset, for the homogeneous evaluation case. Such

a dataset version was obtained by performing a depth-first search of each input tree in the original dataset and pruning the tree whenever it was encountered a node labeled with a tag sharing a concept with one or more tags found at lower depth; in other terms, given $\mathcal{D}$, all tag-nodes $t \in \mathcal{D}$ are kept such that $\nexists\ t' \in \mathcal{D} \mid depth_{\mathcal{D}}(t) < depth_{\mathcal{D}}(t') \wedge \mathcal{C}(t) \cap \mathcal{C}(t') \neq 0$. Trees with depth lower than three were filtered out.

As a result of the DAG-aware pruning, about 12M tree nodes were retained, whereas 3% of distinct tags was left out. Note that about 14% of the trees did not require any pruning, i.e., their corresponding graphs were already DAGs. Comparing the trees before and after the pruning, we found that the maximum depth and maximum fan-out were reduced of about 60% and 95%. By contrast, no significant variation was observed in the average depth and in the average polysemy of tags.

To generate a DAG semantic multidigraph from each pruned tree, the disjoint subsets $L_1, \ldots, L_q$ forming the set of concepts in the DAG were computed by partitioning the tree on a per-level basis; more precisely, in order to ensure the existence of parent-child and ancestor-descendant relations among the concepts to assign to each subset L_h ($h = 1..q$), the key idea was to partition the tree into groups of at least three levels. Formally, given $\mathcal{D}$ with $maxdepth(\mathcal{D}) \geq 3$, we set $q = maxdepth(\mathcal{D})/3$ and compute each L_h ($h = 1..q$-1) as $L_h = \{c | c \in \mathcal{C}(t) \wedge 3h\text{-}2 \leq depth(t) \leq 3h\}$, and $L_q = \{c | c \in \mathcal{C}(t) \wedge maxdepth(\mathcal{D}) - (2 + maxdepth(\mathcal{D})\%3) \leq depth(t) \leq maxdepth(\mathcal{D})\}$.

Figure 3.3(b) shows the time performances of MSSPR and serialized-MSSPR on the obtained DAG-based dataset. Besides the expected drastic reduction in both graph building and ranking times for MSSPR (3-4 orders of magnitude w.r.t. those shown in the upper group of Fig. 3.3(a)), the effect of serialization made MSSPR significantly improve in ranking time, ranging from 168% (*sd*) to 296% (*c,sc*).

3.6.2.2 Evaluation of Monte Carlo MSSPR

We assessed the efficiency of Monte Carlo MSSPR by varying the structural relation types and the number m of random walks. Since the Monte Carlo type approximation has an inherent parallel nature, we estimated the running times by cumulating over the number of random walks (i.e., number of iterations of Monte Carlo MSSPR) the largest times observed for all vertices in the semantic graph.

We observed that the execution time increases with the complexity of structural relation, and hence with the average degree of vertices; in detail (milliseconds), with m=1: 0.70 for *c*, 1.01 for *d*, 1.82 for *c,d*, 2.29 for *sc*, 2.41 for *sd*, 3.06 for *c,sc*, 4.29 for *c,d,sc*, and 5.03 for *sc,sd*. This behavior is explained since, at each step of random walk simulation, the next vertex is chosen by evaluating a larger number of out-going links, which might also involve multiple edges of different types to the same out-neighbors. This result confirms our intuition that the computational cost for the Monte Carlo type

PageRank might be adjusted by also including a factor proportional to the average degree of vertices in a multi-relational ranking context (cf. Sect. 3.4). We also observed a clearly linear increase in the running time w.r.t. m (results not shown). Moreover, compared to MSSPR and pSSPR time performances, Monte Carlo MSSPR was from 3 to 4 orders of magnitude faster, as we expected thanks to the potential of a Monte Carlo method parallel implementation; moreover, the average length of a single simulated random walk $(1/(1-\alpha))$, is generally much shorter than random walks performed by a power-iteration method (like in MSSPR), where dangling vertices do not correspond to termination conditions and teleportation is taken into account.

3.6.3 Summary of findings

In this section we summarize our major findings, with the purpose of shedding light on the various interrelated aspects of our evaluation study.

Our HIN-based definition of semantic network for sense ranking in tree data, and associated multirelational PageRank approach, has turned out to be suitable for improving the performance of the structural sense ranking. In terms of ranking quality w.r.t. ground-truth, this holds for the multi-structure semantic PageRank algorithm (MSSPR) as well as for the weighted combination of structural-type-specific PageRank stationary vectors (pSSPR), according to all assessment criteria. The best-performing structural contexts are *c,d*, *c,sc*, and *c,d,sc*; more precisely, the selection of *c,sc* is preferable to the selection of *c,d* when the input tree instance is single-source (i.e., homogeneous evaluation case), and vice versa when the input tree instance is multi-source (i.e., a relatively deeper and wider tree that encompasses many trees with independent tag-sets). Overall, the advantage in using multi-typed structural contexts (quantitatively expressed in terms of better scores of the assessment criteria) appears as more evident as the conceptual cohesiveness of the tags constituting the input tree instance is higher. Intuitively, this is explained by a higher likelihood of significant semantic relatedness among tags (which implies increased effectiveness in ranking performance) when they are compared through indirect relations in the homogeneous evaluation case. Furthermore, in this case, the locality-aware weighting scheme (which accounts for low support and locality of a structural relation in the tree) can be advantageous over the uniform weighting, which is otherwise preferable.

A theoretical relation between the rankings produced by MSSPR and pSSPR has been identified and proved (cf. Proposition 1), however empirical evidence has shown that in general the two approaches lead to different results. More precisely, pSSPR can achieve results comparable with MSSPR when taking into account a precision measure based on tag-paths ($Bpref$) and an agreement measure based on partial ranking lists (Fs), but it behaves worse than MSSPR in terms of tag-specific sense ranking ($nDCG$). This means that pSSPR, while behaving similarly to MSSPR in detecting the most important tag senses in the input tree, it may fail in disambiguating some tags, or ranking their senses at higher positions. Therefore, we would tend to prefer

MSSPR to pSSPR, unless the efficiency aspect is regarded as primary; in this respect, pSSPR takes some advantage w.r.t. MSSPR due to its parallel nature, although MSSPR runtimes can be of the same order of magnitude as that of the pSSPR runtimes. Moreover, the structural sense ranking problem cannot be addressed by the approach adopted in mS-PPR, that is, by exploiting information on the structural relation types solely to bias a single instance of PageRank: this further confirms the improvement hypothesis that relies on a multi-relational random walking approach applied to a (tree-structure-aware) semantic network.

The ranking performance of our methods is generally boosted by introducing a bias based on the synsets' usage frequency into the personalization vector. Moreover, varying the damping factor in function of topological characteristics (specifically, average path length) of the document-specific network can also lead to further increased performance.

The effect of semantic relatedness measures on the ranking performance is again clearer in the homogeneous evaluation case, where the ontology-path-based approach generally leads to better performance due to its full applicability to any pair of synsets. By contrast, the latter aspect is not guaranteed by the gloss-overlap-based approach, despite the improvements obtained by exploiting ESA for the calculation of the text affinity between glosses.

Interesting observations also stand out as concerns our two proposed optimizations, namely Monte Carlo MSSPR and serialized-MSSPR. Monte Carlo MSSPR drastically reduces the runtime (up to 4 orders of magnitude faster than the deterministic algorithms), and requires a small number of iterations to still yield relatively good ranking performance (with gaps from MSSPR of the order of tenths, for $nDCG$, and cents for the other criteria). Our analysis of serialization of the MSSPR calculation over DAGs has shed light on the significant improvement in ranking time (up to about 300%) obtained on a DAG-aware version of the input dataset; we have also found that the DAG property might not be extremely rare (e.g., 14% of the trees in our evaluation dataset did not require any pruning, i.e., their corresponding graphs were already DAGs), which suggests that a DAG-based approach to multi-relational ranking is worthy of further research as we shall discuss in the next section.

3.7 MSSPR for dependency tree based word sense ranking

In this section, we show how our MSSPR algorithm can be used for a sense ranking task over plain text. We devise a sense ranking framework for plain text, which is based on a semantic heterogeneous information network built using dependency trees. The network is used as ranking context for our multi-relational PageRank algorithm. We perform a preliminary evaluation on a word sense disambiguation task which shows significance of our proposal.

3.7.1 Motivations

As discussed in Section 2.5 of Chapter 2, graph-based ranking methods have been widely applied to semantic networks inferred from plain text. Dependency trees have been successfully applied to Natural Language Processing (NLP) tasks as well. Approaches to WSD which use knowledge about grammatical dependencies in combination with tree matching techniques [164] and graph-based ranking methods [127] can be found in literature.

Nevertheless, to the best of our knowledge, no investigation has been done regarding the possibility to exploit information deriving from different grammatical relations for the construction of a heterogeneous information network. We proved in previous sections the effectiveness of MSSPR on a sense ranking task over semi-structured data (e.g., XML trees). The aim of this study is to devise a technique which allows to build a HIN starting from a plain text input, exploiting parsing-tree-based grammatical relations, and to apply a multi-relational sense ranking method (MSSPR) over it.

3.7.2 Parsing-tree based heterogeneous information network

In order to build a HIN starting from plain text, we resorted to the Stanford Parser[8], a well-known and reliable NLP tool, for the parsing-tree generation phase, and to the WordNet lexical ontology as a knowledge base from which to extract words' concepts (corresponding to WordNet synsets).

The first step of the HIN building process regards the identification of the vertex set $\mathcal{V}$. We now formalize two alternative methods for the construction of the set $\mathcal{V}$:

- *Standard Set.* We build this set upon the set of all concepts (e.g., WordNet synsets) belonging to the words in the input text. Consequently, we had to filter out all terms corresponding to parts of speech which do not have a concept set (e.g., an entry in WordNet), like articles and prepositions.
- *Enriched Set.* It is the *Standard Set*, enriched with the concepts of words in the input terms' glosses extracted from WordNet.

Once a vertex set $\mathcal{V}$ has been selected, a set of structural relations $\mathcal{T}$ (and the corresponding set of typed edges $\mathcal{E}$) have to be defined in order to complete the HIN building process. We devised two different techniques in order to identify significant relations among the concepts in $\mathcal{V}$, which lead to the construction of two different multi-relational graphs:

- *SD Graph.* Each Stanford Dependency[9] in the parsing tree is considered as a different structural relation τ. A edge of type τ is added among each pair of concepts belonging to terms on which the corresponding Stanford Dependency holds in the parsing tree.

[8] http://nlp.stanford.edu/software/lex-parser.shtml
[9] http://nlp.stanford.edu/software/dependencies_manual.pdf

- *NVA Graph*. To reduce the size of the set $\mathcal{T}$(in order to obtain significantly connected edge-type-specific subnetworks), the detailed information about grammatical relations is discarded, and the dependencies are grouped by the parts of speech of the words to which they refer to. We grouped the dependencies in five macro-types:
  - **NN** refers to a dependency between two *noun* terms
  - **VV** refers to a dependency between two *verb* terms
  - **NV** refers to a relation between a *noun* and a *verb* (no matter the direction of the relation)
  - **NVAA** refers to all relations involving *adjective* or *adverb* terms
  - **any** refers to all dependencies. This structural relation was inserted to have at least a structural relation which represents the whole graph structure with an homogeneous edge-type, favoring the score propagation during the random walk process.

  An edge of one of the above-mentioned types is added among all concepts of two terms which are linked by a Stanford Dependency in the parsing tree, based on their parts of speech.

Figure 3.4 shows an example of the parsing tree labeled with the *NVA relations* (left) and a particular of the resulting NVA Graph (right), on an input sentence extracted from the task 17 of SemEval-2010 [4].

3.7.3 Preliminary experimentation

We conducted a preliminary experimentation of our proposed framework on a WSD task, i.e., SemEval-2010 task 17 (*"All-words word sense disambiguation on a specific domain"*). We compared our work with the TreeMatch algorithm [164] as competitor, and with classic baselines like first sense assignment and random sense assignment. We configured our framework as follows: *Standard Set* $\mathcal{V}$, *NVA Graph* relations, unitary edge weights and uniform structural relation weights. This setup has been chosen to reduce the noise coming from an enriched node set, and to avoid the sparsity in the transition matrices which would derive from an oversized set $\mathcal{T}$. Furthermore, we also tested our MSSPR-uf variant of the algorithm, in which the teleportation vector is defined to bias MSSPR according to the usage frequency of the concepts in $\mathcal{V}$. Moreover, we indicate with the subscript *bs* the variants in which the base set (non-zero values in the personalization vector) corresponds to the concepts of the SemEval target words. Among our formulations, MSSPR-uf was the best performing method (precision 0.426-recall 0.370), largely outperforming the random assignment (precision 0.23-recall 0.23), but not first sense assignment (precision 0.505-recall 0.505) and TreeMatch (precision 0.506-recall 0.493). MSSPR-uf$_{bs}$ (precision 0.410-recall 0.355) and MSSPR$_{bs}$ (0.354-recall 0.307) obtained lower but comparable performance.

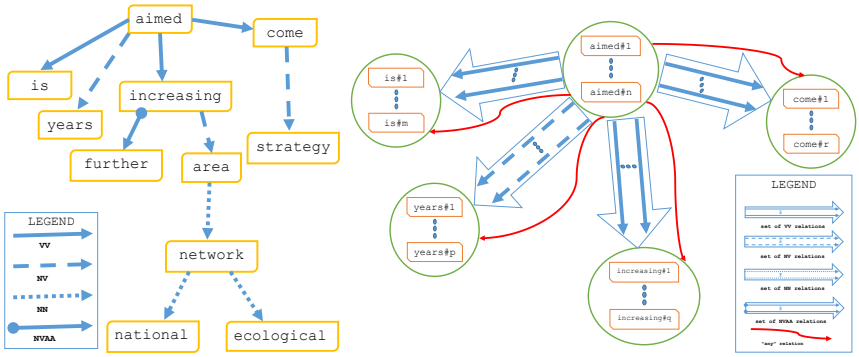


Fig. 3.4. Parsing Tree labeled with NVA relations (left) and particular of the NVA Graph (right) for the sentence: "In the years to come the strategy is aimed at further increasing the area of the National Ecological Network.".

3.8 Chapter review

We addressed the problem of structural sense ranking in tree data proposing a multi-relational PageRank framework over a structure-aware semantic network. We developed different formulations of the problem, focusing on the modeling of a semantic multidigraph as ranking context graph and on PageRank methods that differently handle multi-typed structural relations in tree data. Results have demonstrated the expected improvements in performance w.r.t. the case of single-type structural contexts.

We also discussed an application of the structural sense ranking problem to the unstructured text domain. We represented a plain text by means of dependency trees, which is a common tool in computational linguistic and natural language processing tasks. In this context, the ranking context graph corresponds to a semantic network built on the multiple types of dependencies (e.g., syntactic, morphological, semantic) among the constituents of sentences.

While we have taken XML document trees as a case in point for the experimental evaluation, our approach to structural sense ranking can be readily applied to any kind of domain where semantic-rich data attributes have an inherent hierarchical organization.

As for the identification of possible cases in which the DAG-based approach could be useful, we believe that our DAG-constrained serialization of MSSPR might be suitable for a selective, or targeted, ranking task, i,e, a structural sense ranking task applied only to a (small) subset of tree nodes. In this case, by focusing on the ranking of senses for one or few target tags only, the semantic multidigraph to build would be smaller than in the full-tags ranking task, and hence the likelihood of DAG-admissibility should be increased in general. An index could be developed to conveniently store, during the construction of the semantic multidigraph, those tag-synsets for which the DAG property does not hold. A conditional test could be finally applied to verify

that, if the DAG part of the semantic multidigraph contains the synsets of the target tag(s), then serialized-MSSPR can be carried out instead of the original MSSPR.

Yet, the potential of Monte Carlo type approximation could be further investigated in distributed computing solutions (e.g., [178]). Addressing the structural sense ranking problem under a rank aggregation framework (e.g., [104]) or a cross-view random walk paradigm (e.g., [172]) would also be promising developments.

4

Package recommendation

4.1 Summary

An emerging trend in research on recommender systems is the design of methods capable of recommending packages instead of single items. The problem is challenging due to a variety of critical aspects, including context-based and user-provided constraints for the items constituting a package, but also the high sparsity and limited accessibility of the primary data used to solve the problem. Most existing works on the topic have focused on a specific application domain (e.g., travel package recommendation), thus often providing ad-hoc solutions that cannot be adapted to other domains. By contrast, in this chapter we describe a versatile package recommendation approach that is substantially independent of the peculiarities of a particular application domain. A key aspect in the framework addressed in this chapter is the exploitation of prior knowledge on the content type models of the packages being generated that express what the users expect from the recommendation task. Packages are learned for each package model, while the recommendation stage is accomplished by performing a PageRank-style method personalized w.r.t. the target user's preferences, possibly including a limited budget. The developed method has been tested on a TripAdvisor dataset and compared with a recently proposed method for learning composite recommendations.

4.2 Introduction

Recommender systems are essential part in a variety of information-providing services that aim to satisfy their users' personalized needs. Emerging applications in e-commerce, web search, and web services integrated with social media networks are demanding for systems that are capable of producing enhanced quality recommendations which take into account the heterogeneity in the type of information to personalize and deliver to the users. As a matter of fact, recommending groups of items is the key to successfully face a number of

applications, ranging from business and leisure (e.g., trip planning) to education (e.g., course combination), from finance (e.g., stock market investing) to health-care (e.g., diet planning). In all such applications the items of interest are naturally associated with different types; for example, hotels, restaurants, and points-of-interest in a travel scenario. Therefore, it is highly desirable that recommendations are provided in the form of multi-typed sets of items, or *packages*.

While the known issues in recommender systems extend to the recommendation of packages, providing suggestion lists of packages instead of single items undergoes a number of new challenges. The primary information used to drive the recommendation process still corresponds to the user-item ratings. These are characterized by high sparsity in many domains whereby items are associated with a cost (i.e., price, time) besides a value/score. The volume and quality of the primary data used to learn the packages is also negatively affected by such a high sparsity, but also by the intrinsic difficulty in satisfying different kinds of constraints, which involve compatibility and correlations among items as well as user-specified constraints (e.g., limited budget). After all, several problems for package recommendation have been shown NP-hard, as discussed in [46].

Majority of existing approaches to package recommendation have focused on a particular application domain, usually motivated by very attractive application fields such as tourism. In that case, the design of an effective recommender system has to rely on how well the specific domain challenges have been addressed, often resulting in the development of ad-hoc, hardly generalizable solutions (e.g., [109, 17, 39]). By contrast, there has also been a host of work dealing with set/package recommendation for a generic class of data (e.g., [179, 46, 10, 90]); however, while in some cases being able to express complex constraints and focusing on efficiency or optimization aspects, such studies propose overly sophisticated and rigid systems, and hence how much they could be easy-to-set and really applicable is not clear. Moreover, a common tendency in addressing the package recommendation problem is to develop solutions that are mainly based on top-k query processing (e.g., [10, 90]), combinatorial optimization (e.g., [179]), and statistical models (e.g., [109]), but surprisingly with a limited use of collaborative filtering and graph-based authority ranking techniques. While collaborative filtering should be an essential element in any information personalization task, authority-based ranking methods, such as PageRank, are ever-increasingly applied in a number of information networks, including those supporting recommender systems [98, 74, 108, 64].

We conceive the *package recommendation* problem as follows: Given a set of items of different types along with contextual information, a set of users along with their item ratings, and given prior knowledge on a set of package models of interest: learn how items can be grouped to form context-aware packages that conform to the specified models, and rank the learned package instances specifically for any target user. Our framework, named *PackRec*,

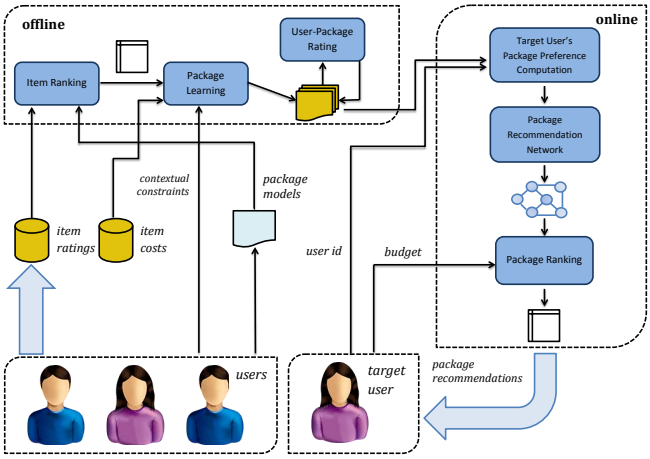


Fig. 4.1. Proposed package recommendation framework.

consists of two stages, as sketched in Fig. 4.1: (i) an offline stage which is centered on the notion of package model(s) with the objective of inducing packages for each of the specified package models, and (ii) an online stage which is in charge of recommending packages tailored to the target user's preferences and budget.

A major novelty of our proposal concerns the definition of a package recommendation framework that integrates well-established paradigms in information retrieval such as expert finding, collaborative filtering, and graph-based ranking methods: expert finding is used to estimate the relevance of items w.r.t. a package model in a collaborative fashion, user-based collaborative filtering is employed in an original odds-ratio based method that models the user's package preferences (on which the recommendation network is built), and a biased PageRank method is finally used to produce a ranked list of recommendations in the form of packages. Moreover, our approach features a certain *versatility* as it can deal with a wide range of application domains, and can work under different settings concerning the structure of the packages to be recommended and even the lack of critical information like the costs of items and/or the limited budget of users.

A preliminary experimentation on a TripAdvisor dataset has shown the recommendation ability of our approach despite the evident criticality of the selected domain. Furthermore, a comparative evaluation with a recently proposed method for composite recommendations has emphasized the superiority of our hypothesis of integration of different information retrieval techniques against a strategy based on an instance-optimal cost/value-item-driven knapsack.

Table 4.1. Main notations used in this chapter

(in) var.	*description*	*(out) var.*	*description*
u, U	user, set of users	p	package
e, I	item, set of items	P	set of packages
$\mathbf{R}_I$	user-item rating matrix	$\mathbf{R}_P$	user-package rating matrix
R_{ue}	rating of item e by user u	R_{ui}	rating of package p_i by user u
$\mathcal{I}$	item type	*primary param.*	*description*
$\boldsymbol{\mathcal{I}}$	set of item types	α	damping factor in Step 5
$\mathcal{P}$	package model	$Usim$	user similarity thr. in Step 4
$\boldsymbol{\mathcal{P}}$	set of package models	*secondary param.*	*description*
$\mathcal{C}$	set of contextual constraints	$\omega_{u\mathcal{I}}, \omega_{u\mathcal{P}}$	weights in Step 3
$\mathcal{S}$	set of constants	λ_u, λ_{ps}	smoothing factors in Steps 1 and 3

4.3 Definitions and notation

We are given a set of users $U = \{u_1, \ldots, u_m\}$ and a set of items $I = \{e_1, \ldots, e_n\}$. Users' rating information is stored into a matrix $\mathbf{R}_I$, where each entry $R_{ue} \in \mathbb{R}^*$ corresponds to the rating given by user u to item e (zero in case of no rating). Moreover, each item might be associated with a real value denoting a cost; similarly, each user could specify a budget, in relation to a given context and time. Optionally, there may be temporal information about when users have rated/reviewed the items; in this case, under the usual assumption that more recent ratings better match the user's current preferences, the importance of a rating would be decreased inversely proportional to its timestamp, for which purpose we will opt for a logarithmic function: for an item e rated by user u with value v at time t, the actual R_{ue} will be set as $R_{ue} = v + v \log(1/(1 + \Delta t))$, where Δt is the difference between the current timestamp and t.

Each item is associated to one type $\mathcal{I}$ from a predefined set of item types $\boldsymbol{\mathcal{I}}$. We will use $\mathcal{I}_e$ to indicate the type associated to item e, and U_e to indicate the set of users related to e.

Constraints might be specified to allow for checking whether an item satisfies certain contextual conditions (e.g., same location for travel items). Therefore, assuming there can be recognized type-independent yet relevant attributes of the items (e.g., location), a set of predicate symbols $\mathcal{C}$ and a set of constants $\mathcal{S}$ are defined that correspond to the sets of attribute names and attribute values, respectively. An atom is an expression of the form $c(e, s)$ with boolean truth values, where c is a predicate symbol and s is a constant; for example, $location(e,''Rome'')$ evaluates to true if e (hotel, restaurant, or any type of attraction) is located in Rome.

Upon the basic notion of item, we define a *package* as a group of items which might be of different types. Specifically, a package p can be seen as a subset of I that conforms to a predefined *package model* $\mathcal{P}$. We assume the existence of a set of package models denoted as $\boldsymbol{\mathcal{P}}$, where each package model is a multiset over $\boldsymbol{\mathcal{I}}$, i.e., $\mathcal{P} = \langle \mathcal{I}(\mathcal{P}) \subseteq \mathcal{I}, f_{\mathcal{P}} \rangle$ where $\mathcal{I}(\mathcal{P})$ denotes the set of *valid item types* and $f_{\mathcal{P}} : \mathcal{I}(\mathcal{P}) \to \mathbb{N}^+$ indicates the number of repetitions for

each item type in $\mathcal{P}$. Table 4.1 summarizes main notations that will be used throughout this chapter.

4.4 Package recommendation framework

The proposed package recommendation framework (Fig. 4.1) takes an input set of users, items, and corresponding ratings along with knowledge on the types of items, desired package models and contextual constraints, and performs the following main steps: offline ranking of the items, learning of the packages, computation of the user-package ratings, and online ranking of the packages tailored to any target user. Each of these steps will be described in the next sections.

4.4.1 Step 1: Collaborative, package-model-aware item ranking

The first step is in charge of identifying the best candidate items to form packages for each of the known models. Given a package model $\mathcal{P}$, the relevance of each item e having type $\mathcal{I}_e$ valid for $\mathcal{P}$ is computed using a probabilistic model, which takes into account three main criteria: (i) the likelihood of e given $\mathcal{P}$, (ii) the likelihood of $\mathcal{I}_e$ given $\mathcal{P}$, and (iii) the a-priori likelihood of e. We now elaborate on each of these terms.

The likelihood of e given $\mathcal{P}$ is expressed by an expert finding model, where $\mathcal{P}$ is assumed to be conditionally independent to e given u:

$$\Pr(e|\mathcal{P}) = \sum_{u \in U_e} \Pr(e|u) \Pr(u|\mathcal{P}) \tag{4.1}$$

Probability $\Pr(e|u)$ measures the relevance of item e for user u and is estimated as

$$\Pr(e|u) = \frac{R_{ue}}{\sum_{e' \in I, \mathcal{I}(e') = \mathcal{I}_e} R_{ue'}} \tag{4.2}$$

Probability $\Pr(u|\mathcal{P})$ can be rewritten as $\Pr(\mathcal{P}|u) \Pr(u)$. $\Pr(u)$ expresses the strength of a user based on her/his degree of activity of rating, which is simply estimated as

$$\Pr(u) = \frac{\sum_{e \in I} \delta_u(e)}{\sum_{u' \in U} \sum_{e \in I} \delta_{u'}(e)} \tag{4.3}$$

where $\delta_u(e) = 1$ if u has rated item e, and zero otherwise. Probability $\Pr(\mathcal{P}|u)$ is determined using a zeroth-order Markov model:

$$\Pr(\mathcal{P}|u) = \Pr(\mathcal{P}|\theta_u) = \prod_{\mathcal{I} \in \mathcal{P}} \Pr(\mathcal{I}|\theta_u) \tag{4.4}$$

where $\Pr(\mathcal{I}|\theta_u)$ can be estimated as: $\Pr(\mathcal{I}|\theta_u) = (1-\lambda_u) \Pr(\mathcal{I}|u) + \lambda_u \Pr(\mathcal{I}|U)$, with smoothing factor $\lambda_u \in [0, 1]$ as an input parameter (set to 0.1 by default).

The likelihood of $\mathcal{I}_e$ given $\mathcal{P}$, i.e., $\Pr(\mathcal{I}_e|\mathcal{P})$, is estimated by linear interpolated smoothing (a.k.a. Jelinek-Mercer):

$$\begin{aligned}\Pr(\mathcal{I}_e|\mathcal{P}) \propto \Pr(\mathcal{I}_e|\theta_{\mathcal{P}}) = \\ = (1-\lambda_{\mathcal{P}})\Pr(\mathcal{I}_e|\mathcal{P}) + \lambda_{\mathcal{P}}\Pr(\mathcal{I}_e|\boldsymbol{\mathcal{P}})\end{aligned} \tag{4.5}$$

where $\Pr(\mathcal{I}_e|\mathcal{P}), \Pr(\mathcal{I}_e|\boldsymbol{\mathcal{P}})$ are maximum likelihood estimators, and $\lambda_{\mathcal{P}} \in [0,1]$ is a smoothing factor; recall that, for a generic element x, a multi-set $\mathcal{X}$, and a set of multi-sets $\boldsymbol{\mathcal{X}}$, the maximum likelihood estimators are determined as $f_{\mathcal{X}}(x)/|\mathcal{X}|$ and $\sum_{\mathcal{X}\in\boldsymbol{\mathcal{X}}} f_{\mathcal{X}}(x)/\sum_{\mathcal{X}\in\boldsymbol{\mathcal{X}}}|\mathcal{X}|$, respectively. The smoothing factor $\lambda_{\mathcal{P}}$ is set based on the following hypothesis: the degree of smoothing should be higher for package models that are less frequent and more complex; formally, it is defined as $\lambda_{\mathcal{P}} = \frac{|\mathcal{P}|}{|\mathcal{P}|+mc(\mathcal{P},\boldsymbol{\mathcal{P}})}$, where $mc(\mathcal{P},\boldsymbol{\mathcal{P}})$ expresses the relative frequency of *containment* of model $\mathcal{P}$ in $\boldsymbol{\mathcal{P}}$, i.e., the fraction of package models of which $\mathcal{P}$ is subset.

Finally, prior $\Pr(e)$ is expressed analogously to (4.3):

$$\Pr(e) = \frac{\sum_{u\in U}\delta_u(e)}{\sum_{e'\in I}\sum_{u\in U}\delta_u(e')} \tag{4.6}$$

Overall, the first step is accomplished by iteratively performing the following set of equations ($\forall e \in I \; s.t. \; \mathcal{I}_e \in \mathcal{I}(\mathcal{P}), \forall \mathcal{P} \in \boldsymbol{\mathcal{P}}$) which determines a ranking of all items for the selected package model:

$$rank_{\mathcal{P}}(e) = [\Pr(e|\mathcal{P}) + \Pr(\mathcal{I}_e|\mathcal{P})]\Pr(e) \tag{4.7}$$

Intuitively, given $\mathcal{P}$, the importance of e according to (4.7) relies on the relevance of the item as a candidate to constitute $\mathcal{P}$ ($\Pr(e|\mathcal{P})$) as well as on the relevance of the item's type w.r.t. $\mathcal{P}$ ($\Pr(\mathcal{I}_e|\mathcal{P})$). Yet, $\Pr(e)$ accounts for the popularity of e independently of the generation of packages, and hence acts to penalize items that are less rated by the users.

4.4.2 Step 2: Context-driven package learning

The item rankings computed at the previous step are used to construct a set of package instances for all package models in $\boldsymbol{\mathcal{P}}$. Besides the type compatibility to ensure for the items w.r.t. any given package model $\mathcal{P}$, contextual constraints at the instance level might also be satisfied. Therefore, predefined contextual predicates could be here applied to extract a subset of $\mathcal{P}$-compatible items, for each of the $|\mathcal{P}|$ item types that conform to $\mathcal{P}$. Given a DNF formula $\mathcal{A}$ over a subset $C \subseteq \mathcal{C}$ of predicates and a subset $S \subseteq \mathcal{S}$ of constants, a set of items $I_{\mathcal{I}}^{\mathcal{A}}$ is derived for each item type $\mathcal{I} \in \mathcal{I}(\mathcal{P})$ such that $I_{\mathcal{I}}^{\mathcal{A}}$ contains the items of type $\mathcal{I}$ that also satisfies $\mathcal{A}$.

We automatically select the top-ranked items for the various types valid for a given package model by pipelining *Pareto-frontier* computations over the $I_{\mathcal{I}}^{\mathcal{A}}$. For each $I_{\mathcal{I}}^{\mathcal{A}}$, alternatives among the items are evaluated according to their

normalized cost/normalized score ratio, where the item scores are computed by (4.7); i.e., upon an ordering of the items by increasing cost/value ratios, the next item with smallest rank is added to the Pareto-frontier if it is not dominated by any item already in the Pareto-frontier. To check whether a candidate item is dominated by any alternative in the Pareto-frontier, it will suffice to find the most expensive alternative which is still cheaper than the candidate item, since it does not dominate the candidate item, neither can any other alternative in the Pareto-frontier. It should be however noted that for some types of items, either the cost information is not applicable or the value (score) dimension should weight more than the cost of items; consequently, a cost/value based strategy for the selection of top items might not necessarily lead to the best choice in terms of item coverage. For this purpose, we also used a cost-free selection strategy such that an item will be included in the top-ranked list for type $\mathcal{I}$ if its score has a percentage decrease from the top-1 item which is not greater than the average of the percentage decreases of all items from the top-1 item.

The set of packages for each model $\mathcal{P}$ is finally computed as the result of the Cartesian product of the $|\mathcal{P}|$ sets of selected type-specific items for $\mathcal{P}$. Each package p is provided in the form of a set of items.

4.4.3 Step 3: Package rating

Once a set P of packages for all package models has been learned, user ratings for the packages need to be computed. Such ratings can certainly be inferred from the ratings of the items that belong to a package: however, the usual sparsity of the user-item matrix is expected to be further exacerbated when selecting the small bunch of items that constitutes any given package. In order to alleviate this issue, we introduce a smoothing scheme that refines the actual R_{ue}, with $e \in p$: $(1 - \lambda_{ps})R_{ue} + \lambda_{ps}\overline{R_e}$, where $\overline{R_e}$ denotes the average rating of e over all users, with λ_{ps} set to 0.1 by default.

Besides users' ratings of the items constituting a package, other information might be taken into account when computing the rating of a user for a package. A user may want to differentiate among the types of items constituting a package: given user u and package p, this can be expressed by a set of weights $\omega_{u\mathcal{I}} \in [0, 1]$, with $\mathcal{I} \in \boldsymbol{\mathcal{I}}$ each of which might act as a damping (or even as nullifying) factor for a particular item type; by default, such weights are set to 1 meaning that all constituent items are fully considered when rating the package. Moreover, a user may specify prior preferences over package models, based on her/his individual exogenous source of information: this is expressed by a coefficient $\omega_{u\mathcal{P}} \in [0, 1]$ (set to 1 by default), for each $\mathcal{P} \in \boldsymbol{\mathcal{P}}$.

The above types of information are combined to define a *package rating* function $p\text{-}score : U \times P \to \mathbb{R}^*$, which is computed for any given user u and package p as:

$$p\text{-}score(u,p) = \omega_{u\mathcal{P}} \frac{\sum_{e \in p} \omega_{u\mathcal{I}_e} [(1-\lambda_{ps}) R_{ue} + \lambda_{ps} \overline{R_e}]}{\sum_{e \in p} \omega_{u\mathcal{I}_e}} \tag{4.8}$$

A *user-package rating matrix*, denoted as $\mathbf{R}_P$, is derived by storing the computed $p\text{-}score(u,p)$ values, for all $u \in U$ and $p \in P$. Hereinafter we will use notation R_{ui} to denote the rating given by user u to package p_i.

4.4.4 Step 4: Target user's package recommendation network

A *package recommendation network* will be designed for any given target user, and used to produce a ranking (probability distribution) over the set of packages obtained from the offline stage, thus providing recommendations to the target user. The network is modeled as a directed graph in which vertices are the learned packages and edges are drawn according to a function that models the user's preference on pairs of packages. The role of this preference function is to define the transition probabilities in the package ranking model discussed later. The sign of the preference function will determine the orientation of the edge, while a pair of reciprocal edges will be drawn in case of no preference expressed for any two packages. The strength of the connection between two packages is set to be proportional to the value of the preference function. Formally, the package recommendation network for the target user u is defined as $\mathcal{G}_u = \langle \mathcal{V}, \mathcal{E}, w_u \rangle$, with set of vertices $\mathcal{V} = P$, set of edges $\mathcal{E} = \{(p_i, p_j) | p_i, p_j \in P \wedge \pi_u(p_j, p_i) \geq 0\}$, and edge weighting function $w_u : \mathcal{E} \to \mathbb{R}^+$ such that $w_u(i,j) = e^{\pi_u(p_j, p_i)}$.

Package preference function

Given a user u, our objective is to model a function of the form $\pi_u : P \times P \to \mathbb{R}$, with $\pi_u(p_i, p_j) > 0$ when package p_i is considered as more preferable to package p_j for user u, and vice versa, whereas $\pi_u(p_i, p_j) = 0$ means that there is no preference between the two packages. We require that $\pi_u(p_i, p_i) = 0$, for all $p_i \in P$ and $\pi_u(p_i, p_j) = -\pi_u(p_j, p_i)$, for all $p_i, p_j \in P$. Moreover, to adequately determine the strength of preference, we are interested in modeling the target user's package preferences in a collaborative setting.

Let us denote with U_{ij} the set of users that have rated p_i or p_j, or both. We define in terms of *odds ratio* the strength of association of two random variables: the one expressing the condition "users in U_{ij} are similar to u" and the other one expressing the event "p_i is rated higher than p_j". Hence, an odds ratio greater than (resp. lower than) 1 indicates that the event of rating p_i higher than p_j is more likely to occur in the group of users similar (resp. not similar) to u, while an odds ratio equal to 1 indicates that whether or not users are similar to u is irrelevant to prefer p_i over p_j.

To determine user similarity, we can resort to various similarity measures used in collaborative filtering [108]. The extent to which two users are considered as similar to each other can be controlled by a minimum-similarity threshold, henceforth denoted as $Usim$, which can be experimentally varied.

The collaborative-based preference odds ratio computed for each pair p_i, p_j is then combined with the variation in u's ratings of p_i, p_j to model the preference function.

$$\pi_u(p_i, p_j) = (R_{ui} - R_{uj})(OR_{u,ij})^{1-2\chi_{u,i,j,OR}} \tag{4.9}$$

where $\chi_{u,i,j,OR}$ is the indicator function for the event "$(R_{ui} - R_{uj}) < 0$", and

$$OR_{u,ij} = \frac{N_u^{>}}{N_u^{<}} \cdot \frac{notN_u^{<}}{notN_u^{>}} \tag{4.10}$$

with $N_u^{>}$ (resp. $notN_u^{>}$) indicating the add-one smoothed number of users similar (resp. not similar) to u that have rated p_i strictly higher than p_j, $N_u^{<}$ (resp. $notN_u^{<}$) indicating the add-one smoothed number of users similar (resp. not similar) to u that have rated p_i strictly lower than p_j.

It can be noted that the two terms in (4.9) play a different role in modeling the preference function: while the "versus" of preference is determined by $(R_{ui} - R_{uj})$, the odds ratio acts as a strengthening (resp. damping) factor if there is a concordance (resp. discordance) in sign between the difference of u's ratings and the logarithm of the odds ratio. Note also that, as required, $\pi_u(p_i, p_j) = -\pi_u(p_j, p_i)$ since it holds that $OR_{u,ij} = OR_{u,ji}^{-1}$ and $\chi_{u,i,j,OR} = 1 - \chi_{u,j,i,OR}$, which ensure that the overall odds ratio term is identical in both $\pi_u(p_i, p_j)$ and $\pi_u(p_j, p_i)$.

4.4.5 Step 5: Package ranking

We develop a PageRank-style method for ranking the set of packages specifically for each target user. PageRank-style methods have been already successfully applied to item recommendation problems [98, 74, 108, 64]. A major motivation is that the underlying Markov chain model is effective to face the usual lack of rating or preference information that characterize many users. In other terms, if preferences between packages p_i, p_j have been expressed by some users, and preferences between packages p_i, p_k have been expressed by other users, the preferences regarding p_j, p_k are not known and hence can be inferred through an iterated random walk. In our package recommendation setting, an intuitive interpretation of the classic PageRank idea is that the importance of a package (i.e., the likelihood of being preferred to other packages) both relies on and influences the importance of neighboring packages in the network $\mathcal{G}_u$. This is captured by the following equation that computes the ranking score r_i for a given package p_i:

$$r_i = \sum_{j \in In(i)} \frac{w_u(j,i)}{\sum_{h \in Out(j)} w_u(j,h)} r_j \tag{4.11}$$

where, for any vertex i, $In(i)$ and $Out(i)$ denote the in-neighbor and out-neighbor sets, respectively. The above equation can be written in the equivalent matrix notation as $\mathbf{r} = \mathbf{S}^{\mathrm{T}}\mathbf{r}$, where $\mathbf{S}$ is the *package connectivity* stochastic

matrix defined as $\mathbf{S} = \mathbf{D}_{out}{}^{-1}\mathbf{W} + \mathbf{a}\mathbf{e}^{\mathrm{T}}/|\mathcal{V}|$, such that $\mathbf{W}$ is the weighted adjacency matrix of $\mathcal{G}_u$, $\mathbf{D}_{out} = diag(\mathbf{W}\mathbf{e})$ is the diagonal matrix storing the (weighted) out-degrees with $\mathbf{e}$ denoting a $|\mathcal{V}|$-dimensional column vector of ones, and $\mathbf{a}$ is the dangling-vertex vector such that $a_i = 1$ if vertex i has zero out-degree (i.e., package p_i is not rated), and 0 otherwise. To ensure the convergence of the Markov chain with $\mathbf{S}$ to a stationary distribution, the usual primitivity adjustment is introduced as a convex combination of $\mathbf{S}$ with another stochastic matrix defined as $\mathbf{B} = \mathbf{v}\mathbf{v}^{\mathrm{T}}/|\mathcal{B}|$. Vector $\mathbf{v}$ and set $\mathcal{B}$ are equal by default to $\mathbf{e}$ and $\mathcal{V}$, respectively, but $\mathbf{v}$ can be replaced with any vector whose non-negative components sum up to 1 and that can be used to personalize the PageRank to boost a specific subset of vertices (base-set $\mathcal{B}$). $\mathbf{B}$ is also known as teleportation matrix, since the random walker can decide not to follow the link structure by selecting a vertex with relevance $1/|\mathcal{B}|$. A parameter α between 0 and 1 (commonly set to 0.85) controls the proportion of the random walk based on the link structure as opposed to teleporting: $\mathbf{r} = \alpha\mathbf{S}^{\mathrm{T}}\mathbf{r} + (1-\alpha)\mathbf{v}/|\mathcal{B}|$.

We define a *cost-sensitive personalization* of our PageRank-based method, according to the following two criteria: (i) $\mathcal{B}$ corresponds to the subset of packages in P having the same model as the packages that have been rated by u, and (ii) each of the selected packages in $\mathcal{B}$ must have a cost (computed over its constituting items) not greater than a specified budget b_u.

4.5 Experimental evaluation

4.5.1 Data and evaluation methodology

To assess our approach we chose to focus on the travel planning domain, given the increasing interest it has produced as a major application for package recommendation tasks. We used the popular TripAdvisor.com data as case in point for our evaluation. During April 2013, we crawled information about hotels, restaurants, and all available types of attractions along with the associated users' ratings, starting from the Top-Destinations section of the website[1]. This resulted in 48,131 hotels, 19,802 B&Bs, 159,716 restaurants, and 21,661 attractions classified in 133 categories, with a total of 249,310 items, and 12,622,091 ratings made by 4,004,926 users over 230,814 items. However, the very high sparsity of the rating matrix (above 99%) prompted us to restrict our selection to a subset of locations in the attempt of reaching a good tradeoff between salience of the venue in TripAdvisor (in terms of user popularity), diversity in terms of the attractions a venue usually offers (i.e., diversity in the set of item types), and suitability of the locations to perform queries related to different travel topics (i.e., nature, business, historical sites, etc.). As a result, we selected 15 locations which are shown in Table 4.4.

[1] http://www.tripadvisor.com/TravelersChoice-Destinations

To assess the proposed and competing methods, we assigned a reference ranking score to each package p, which takes into account the TripAdvisor-supplied rankings and costs of the constituting items as well as a user-provided budget (b):

$$rank(p,b) = \text{avg}_{\mathcal{I}\in\mathcal{P}} \left(\frac{\sum_{e:\mathcal{I}_e=\mathcal{I}} rank(e)}{\sum_{i=1..m(\mathcal{I}_e)} i} \right) \frac{1}{(cost(p)-b)+1}$$

where $rank(e)$ is the rank of item e w.r.t. the item-type specific ranking provided by TripAdvisor, $cost(p)$ is the total cost of package p calculated over its items' costs, b equals the budget specified by the target user (i.e., $b = b_u$), and $|p|$ denotes the number of item-types (with duplicates) for the package model of p. Note that the formula penalizes out-of-budget packages.

We used four assessment criteria that are standard in ranking tasks, namely *mean average precision* (MAP), *Kendall rank correlation coefficient*, *normalized discounted cumulative gain* (nDCG), and *Fagin's intersection metric* (cf. Chapter 2, Section 2.7). For each of them, higher scores correspond to better ranking evaluation.

In our setting of the MAP measure, the personalized ranking produced for a given user is considered as a query, and the number of relevant and retrieved candidates is obtained by taking into account the top-k-ranked packages.

4.5.2 Experimental settings

PackRec parameters were setup using the default values as declared in their definitions; moreover, cosine similarity and $Usim$ set to 0.6 were used for the user neighborhood computations (cf. Sect. 4.4.4). We however undertook a preliminary investigation on how the PackRec performance was influenced by λ_{ps} (cf. Sect. 4.4.3) and by $Usim$: in summary, we observed that a value for λ_{ps} close to zero (e.g., the default 0.1) was enough to alleviate the sparsity issue but also to avoid a loss of discrimination between the individual users' ratings in scoring the packages; similar neighborhood sizes were observed when setting $Usim$ to $0.6 \div 0.8$, and even down to 0.5 in a few locations, while lower (resp. higher) values would lead to a nearly total (resp. null) coverage of the users.

We devised two evaluation stages, the first focused on the assessment of our PackRec performance under different settings, and the second devoted to a comparative evaluation with the instance-optimal algorithm in [179], hereinafter referred to as CompositeRec, as anticipated at the end of Sect. 4.6.

In the first stage, the selected locations were grouped into four categories according to their attractions. As shown in Table 4.4, we defined a set of four package models based on the attraction types available in the corresponding locations and with the attempt of configuring four types of trip, namely cultural, family, business, and fun trip. We specified the input budgets for each location trying to simulate different price ranges, according to the following

strategy: we calculated the total cost of each package learned contextually to a given location, then we analyzed the package costs in increasing order, fixing a budget threshold when the next cost was a certain percentage (set to 20%) higher than the lower cost in the current range. Note that the setting corresponding to the highest budget (i.e., budget set equal to the maximum package cost for the location) will correspond to no budget-driven personalization of the ranking.

In the second stage, since CompositeRec requires a number of recommender systems in input, we exploited a state-of-the-art method, called ItemRank [64], to generate personalized item rankings for the five most active users for each location. Since the package models of the packages returned by CompositeRec cannot be known in advance, we carried out CompositeRec to recommend the top-10 packages for each of the selected users per location, by setting the budget per location as equal to the corresponding highest budget used by our PackRec in the first stage. Then, the models of the packages produced by CompositeRec were used to drive the package learning step of PackRec. For the comparative evaluation of the recommendations, we needed to generate a different reference ranking for each user, containing both the packages returned by CompositeRec and by PackRec, constrained to the same budget per location.

4.5.3 Results

Table 4.2 reports on performance results obtained by PackRec over the various locations; parameter k in Fagin, nDCG and MAP was set to cover 10% of the package set. For each location and assessment criterion, results correspond to averages over the different budgets selected for the location. Moreover, for each location, we selected two different sets of users, corresponding to the upper-quartile and the lower-quartile, respectively, based on the users' activity in terms of total ratings on items belonging to the location. Note that no distinction was made between positive (high) and negative (low) ratings of the users, which clearly affected the overall quality of the average performance results: these were indeed generally low due to a partial agreement with the TripAdvisor-driven reference ranking generated for each location (cf. Sect. 4.5.1), which by definition tends to rank higher positively rated items, and hence packages.

Looking at the results per location in each group, lower scores corresponded to average Fagin and MAP, which however behaved quite closely. Generally higher scores were achieved in terms of Kendall, which would indicate a higher overall alignment w.r.t. the reference ranking than in the cases where the head of lists was taken into account, and in terms of nDCG, which means that the shared packages between the top-k list of PackRec and the top-k list of the reference ranking obtained similar ranks and relevance scores. By comparing the corresponding group average performances in the two cases of user activity, improved results could also be obtained in the lower-quartile

Table 4.2. Average performances of PackRec.

Location	**Fagin**		**Kendall**		**nDCG**		**MAP**
	avg	*max*	*avg*	*max*	*avg*	*max*	
upper-quartile users							
Amsterdam	.228	.393	.432	.557	.503	.625	.193
BuenosAires	.024	.218	.165	.641	.173	.945	.038
Istanbul	.175	.503	.417	.574	.440	.783	.143
group avg	.129	.342	.315	.597	.348	.796	.116
Barcelona	.112	.246	.271	.435	.317	.473	.123
Chicago	.153	.602	.223	.689	.416	.871	.152
Honolulu	.125	.398	.296	.422	.385	.707	.113
SanFrancisco	.115	.609	.172	.617	.464	.822	.123
Sydney	.184	.646	.156	.419	.479	.836	.160
WashingtonDC	.128	.402	.210	.447	.437	.779	.112
group avg	.135	.462	.225	.485	.411	.727	.129
Edinburgh	.170	.552	.349	.608	.422	.924	.205
Rome	.165	.497	.426	.646	.308	.628	.120
Venice	.096	.302	.262	.399	.304	.509	.071
group avg	.142	.441	.350	.550	.334	.655	.123
NiagaraFalls	.117	.231	.182	.327	.361	.518	.099
PlayadelCarmen	.185	.455	.192	.549	.428	.768	.182
SharmElSheikh	.103	.366	.219	.440	.379	.748	.102
group avg	.139	.363	.198	.451	.393	.692	.132
lower-quartile users							
Amsterdam	.191	.252	.443	.525	.474	.609	.161
BuenosAires	.099	.389	.324	.555	.221	.595	.069
Istanbul	.148	.336	.466	.604	.383	.571	.122
group avg	.142	.329	.397	.555	.346	.595	.114
Barcelona	.102	.120	.272	.411	.267	.318	.123
Chicago	.079	.261	.155	.531	.356	.656	.108
Honolulu	.147	.351	.377	.532	.448	.796	.162
SanFrancisco	.146	.649	.253	.487	.545	.784	.130
Sydney	.053	.098	.305	.522	.341	.496	.058
WashingtonDC	.154	.373	.274	.533	.444	.741	.125
group avg	.114	.293	.281	.498	.393	.613	.118
Edinburgh	.180	.520	.439	.664	.501	.869	.204
Rome	.220	.522	.495	.651	.400	.646	.192
Venice	.065	.233	.241	.391	.256	.543	.051
group avg	.155	.419	.393	.562	.373	.662	.149
NiagaraFalls	.156	.787	.481	.592	.766	.925	.161
PlayadelCarmen	.123	.239	.321	.594	.348	.512	.102
SharmElSheikh	.166	.454	.341	.574	.489	.778	.180
group avg	.148	.463	.372	.587	.511	.715	.148

Table 4.3. Best performances (Fagin and MAP scores) of PackRec and CompositeRec over different locations.

Location	PackRec		CompositeRec		Location	PackRec		CompositeRec	
	Fagin	MAP	Fagin	MAP		Fagin	MAP	Fagin	MAP
Amsterdam	.530	.659	.089	.090	PlayadelCarmen	.219	.254	.022	.028
Barcelona	.589	.841	.184	.307	Rome	.487	.735	.099	.089
BuenosAires	.528	.828	.112	.111	SanFrancisco	.486	.314	.193	.184
Chicago	.496	.839	.135	.124	SharmElSheikh	.466	.793	.092	.085
Edinburgh	.514	.586	.086	.069	Sydney	.493	.913	.119	.106
Honolulu	.520	.682	.072	.061	Venice	.607	.931	.229	.524
Istanbul	.589	.887	.101	.112	WashingtonDC	.296	.494	.057	.032
NiagaraFalls	.623	.954	.071	.072	*avg*	.496	.714	.111	.133

case (i.e., for groups 3 and 4), where the sparsity in the user's ratings is generally higher.

PackRec also appeared to be moderately robust w.r.t. the various queries (i.e., package models) and location group settings, since the differences among the group averages were relatively low in terms of average Fagin and MAP. By analyzing the personalization of the recommendations obtained for different budgets (results not shown), we observed that in several cases smaller budgets corresponded to better top-rankings (i.e., higher Fagin scores, nDCG and MAP), which might be explained by the fact that smaller budgets led to filter out a larger number of packages, facilitating the ranking task.

Comparison with CompositeRec

Table 4.3 shows the results obtained by PackRec and CompositeRec. Comparison was limited to the Fagin and MAP criteria for two main reasons: both the PackRec and the CompositeRec outputs are incomplete rankings w.r.t. the merged reference ranking described above (which prevents the use of Kendall), and the CompositeRec output ranking is produced without scores (which prevents the use of nDCG). Parameter k for Fagin was set to 10, since we let CompositeRec to recommend the top-10 packages for each of the selected users per location.

As we expected, CompositeRec produced packages significantly different from those involved in our first stage of evaluation: over the various locations, the number of distinct models varied from 5 to 10, and the size (i.e., number of item types) from 2 to 35 (average of 13.9). Moreover, packages with multiple items of the same type were also taken into account—while the average multiplicity was always close to 1, in each location there was however at least one item type with more than 10 occurrences in a package model.

Results in Table 4.3 show that PackRec achieved significant correlation with the combined reference ranking in all locations. It came to our surprise that, despite PackRec was trained under a setting driven by CompositeRec recommendations, PackRec clearly outperformed CompositeRec in all cases

and for both criteria, with average gains of 0.385 Fagin and 0.581 MAP. Qualitatively, worse performance by CompositeRec was indeed explained by the fact that most of the 10 packages returned by CompositeRec per location have poor match with the top-10 packages in the reference ranking. This would give evidence that PackRec can effectively deal with packages of varying structure and that its combination of expert finding, collaborative filtering and graph-based ranking allows for better performance than a cost/value knapsack strategy.

4.6 Related work

In the last few years recommendation of sets/packages of items has been studied from various perspectives and with different levels of expressiveness. [28] proposes a general decision support system for the definition of composite alternative recommendations. While different user-specified requirements are taken into account, the recommendation process is highly interactive, as it utilizes the target user's feedback to feed a preference learner module, and hence it might continue iteratively until the user is satisfied with one of the alternatives. [10] approaches the problem from a top-k query processing perspective, by introducing the class of entity package finder query. Such queries are used to identify the top-k tuples of entities, according to the relevance of each entity w.r.t. a given set of keywords. Like our approach, associations among the entities (i.e., item types) are known in advance; however, entity package finder queries cannot directly handle user-specified constraints such as budget to control the identification of most relevant packages and, in general, they do not consider a collaborative-based support to provide a personalized recommendation to a target user. In [90], complex yet customized recommendations are defined declaratively as a high-level workflow over relational data, in which traditional and enhanced relational algebra operators are used to specify user-requirements and to generate virtual nested relations. The approach in [90] is designed to maximize flexibility in recommendation, however at the cost of higher complexity of the recommendation engine and difficulty to control contextual and cost requirements.

Unlike our work, the approaches in [39, 17] exploit geo-temporal and multimedia data to provide recommendations about popular touristic places. Both studies however do not take item/package rating into consideration, thus they are limited to provide composite recommendations for a set of user-specified temporal constraints. The latter differences w.r.t. our work are also present in [165], which studies the general problem of set-based queries with aggregation constraints.

In [109], travel packages are generated through the TAST (tourist-area-season topic) model, based on latent topic distributions of tourist-season pairs. Such topic distributions are used to find seasonal nearest neighbors for each tourist, and collaborative filtering based ranking is employed to personalize suggestions. An extension of the TAST model, named TRAST (tourist-

Table 4.4. Package models and statistics about the learned packages.

Group	Location	# Ratings per Location	# Learned Packages	Percentage of packages covered by each budget range	Package Models
1	Amsterdam	58,746	126	3.97%, 28.57%, 54.76%, 84.92%	$\mathcal{P}_1$: Hotel, Restaurant, Museums, HistoricSites, ArchitecturalBuildings
	Buenos Aires	56,926	64	15.63%, 18.75%, 51.56%, 56.25%, 81.25%	$\mathcal{P}_2$: Hotel, Restaurant, Landmarks/PointsofInterest, Parks, Entertainment
	Istanbul	59,538	64	15.63%, 60.94%	$\mathcal{P}_3$: B&B, Restaurant, Performances, Entertainment
					$\mathcal{P}_4$: Hotel, Restaurant, HistoricSites, Civic/ConventionCenters
2	Barcelona	73,389	68	4.41%, 7.35%, 19.12%, 54.41%, 82.35%	$\mathcal{P}_5$: Hotel, Restaurant, Museums, HistoricSites, ReligiousSites
	Chicago	60,852	64	75.00%, 87.50%	$\mathcal{P}_6$: Hotel, Restaurant, HistoricSites, Gardens
	Honolulu	38,049	80	11.25%, 18.75%, 45.00%, 62.50%	$\mathcal{P}_7$: B&B, Restaurant, Museums, Theaters
	San Francisco	69,041	64	62.50%, 87.50%, 93.75%	$\mathcal{P}_8$: Hotel, Restaurant, HistoricSites
	Sydney	39,276	68	5.88%, 14.71%, 75.00%, 98.53%	
	WashingtonDC	46,403	64	14.06%, 25.00%, 76.56%, 95.31%	
3	Edinburgh	52,382	112	21.43%, 57.14%, 77.68%	$\mathcal{P}_9$: Hotel, Restaurant, Museums, ArchitecturalBuildings, ReligiousSites
	Rome	28,428	104	15.38%, 23.08%, 30.77%, 65.38%, 82.69%, 88.46%	$\mathcal{P}_{10}$: Hotel, Restaurant, Parks, Entertainment
	Venice	51,873	64	17.19%, 37.50%, 50.00%, 70.31%, 79.69%	$\mathcal{P}_{11}$: B&B, Restaurant, Performances, Entertainment
					$\mathcal{P}_{12}$: Hotel, Restaurant, ArchitecturalBuildings, ReligiousSites
4	Niagara Falls	17,843	96	3.13%, 11.46%, 40.63%, 90.63%	$\mathcal{P}_{13}$: Hotel, Restaurant, Theaters
	Playa del Carmen	28,629	88	5.68%, 12.50%, 15.91%, 18.18%, 72.73%, 86.36%	$\mathcal{P}_{14}$: Hotel, Restaurant, Entertainment, SportsCamps/Clinics
	Sharm El Sheikh	21,431	64	6.25%, 21.88%, 40.63%, 59.38%, 75.00%	$\mathcal{P}_{15}$: B&B, Restaurant, Entertainment, SportsCamps/Clinics
					$\mathcal{P}_{16}$: Hotel, Restaurant, Theaters, Parks

relation-area-season topic) is also defined to model the tourist relationships in a travel group. The approach in [109] can provide recommendations only for users who have traveled at least once in the existing travel records, i.e., users who have rated at least one package, while our approach can even handle target user's partial ratings about the items constituting the candidate packages to recommend. While in our model a package might contain different item types, each one with arbitrary multiplicity, in [109] a package is substantially an array of landscapes. Also, the TAST/TRAST models are in principle applicable to other scenarios but only provided that certain assumptions hold at the basis of the topic model.

The composite recommendation system proposed in [179] is centered on a domain-independent approach that extends the knapsack problem. It assumes the availability of multiple recommender systems to score the items for a specific user and of an external information source that provides the items' costs. Given a cost budget and an integer k, the method computes top-k variable-length packages to recommend. Two approximation algorithms are developed, an instance-optimal, pseudo-polynomial algorithm and a greedy algorithm. Unlike our work, packages are learned regardless of the specific type and compatibility constraints of the constituting items; moreover, the knapsack-like approach adopted in [179] is by nature not particularly tailored to the various target user's preferences, which are in fact simply summarized in the specification of the budget constraint and whose understanding cannot easily be refined in a collaborative fashion.

Nevertheless, we involved the approach by [179] in a comparative evaluation with our PackRec, not only because the two works are both domain-independent and do not rely on a text processing via language modeling (like [109] does), but also because this evaluation allowed us to stress our PackRec under a different setting in which the schema of the learned packages can be highly varying and is not user-provided.

4.7 Chapter review

We presented an application-independent framework for the recommendation of packages of items. While being fully unsupervised, the framework relies on a-priori (user-provided) knowledge on the structure and types of the packages to be learned and recommended in a personalized fashion. Testing on datasets of other domains is certainly needed to fully support our claim of application-domain independence and versatility. Moreover, besides deepening the evaluation of the PackRec sensitivity to the various parameters, we have identified a number of points to be studied, including the definition of a regularization framework to address a possible item correlation issue in the package learning step, and alternative graph models for the package ranking step.

5

Ranking lurkers in social networks

5.1 Summary

The massive presence of silent members in online communities, the so-called *lurkers*, has long attracted the attention of researchers in social science, cognitive psychology, and computer-human interaction. However, the study of lurking phenomena represents an unexplored opportunity of research in data mining, information retrieval and related fields. In this chapter, a formal specification of lurking is provided and the new problem of *lurker ranking* is addressed through the discussion of the first centrality methods specifically conceived for ranking lurkers in social networks. An extensive analysis on static and dynamic contexts has shown the significance of the lurker ranking approach, and its uniqueness in effectively identifying and ranking lurkers in an online social network. Experiments were carried out using real-world social networks like Twitter, Flickr, FriendFeed, GooglePlus and Instagram as cases in point, in order to evaluate the lurker ranking methods' performance against data-driven rankings as well as existing centrality methods, including the classic PageRank and alpha-centrality. Analyses concerning the application of the lurker rank algorithms in social trust contexts and in other domains(e.g., research collaboration networks), which point out the flexibility of the discussed approach are provided as well.

5.2 Introduction

The majority of members of online communities play a passive or silent role as individuals that do not readily contribute to the shared online space. Such individuals are called *lurkers*, since they belong to a community but remain quite unnoticed while watching, reading or, in general, benefiting from others' information or services without significantly giving back to the community.

Lurking characterization in online communities has been a controversial issue from a social science and computer-human interaction perspec-

tive [50]. Since the early works on social motivations and implications of lurking [131, 135], one common perception of lurking is that based on the infrequency of active participation to the community life, but other definitions have been given under the hypotheses of free-riding [89], legitimate peripheral participation [97, 70], individual information strategy of microlearning [82], and knowledge sharing barriers (e.g., interpersonal or technological barriers) [11]. Lurkers might also be perceived as a menace for the cyberspace as they maliciously feed on others' intellects. For instance, in P2P file sharing systems [47], lurking may correspond to a leeching behavior whenever a user wastes valuable bandwidth by downloading much more than what s/he uploads. In the realm of online social networks (OSNs), negative views of the lurkers have been however supplanted with a neutral or even marginally positive view. A neutral perception of lurkers is related to the fact that their silent presence is seen as harmless and reflects a subjective reticence (rather than malicious motivations) to contribute to the community wisdom; half of times, a lurker simply feels that gathering information by browsing is enough without the need of being further involved in the community [135]. However, lurking can be expected or even encouraged because it allows users (especially newcomers) to learn or improve their understanding of the etiquette of an online community before they can decide to provide a valuable contribution over time.

Lurking is responsible for a *participation inequality phenomenon* that is shared by all large-scale online communities. This phenomenon is explained by the so-called "1:9:90" rule, which states that while 90% of users do not actively contribute, 9% of users may contribute (i.e., comment, like or edit) from time to time, and only 1% of users create the vast majority of social content [131, 135]. Consequently, such inequities lead to a biased understanding of the community, whereby a major risk is that we will never hear from the silent majority of lurkers. Therefore, a challenge is to attract, or *de-lurk*, the crowd of lurkers, whereby online advertising strategies should be tailored to the lurkers' behavioral profile. Moreover, since lurkers have knowledge about the online community (as a result of the substantial time they dedicate towards learning from the community), delurking can mainly be seen as a mix of strategies aimed at encouraging lurkers to return their acquired social capital, through a more active participation to the community life.

Understanding user behaviors has long been studied in online social networks. A key element that is shared by all studies is the use of a social graph model as the basic tool to represent relationships among users [173]. Relationships, or *ties* [67], can vary over a spectrum that include friendships and followships [92, 7, 125, 93, 36, 176], visible interactions [40, 102, 169, 176, 118], and latent interactions (based on, e.g., browsing profiles or clickstream data) [142, 18, 80].

Surprisingly, despite the fact that lurking has been recognized and surveyed in social sciences, at the time of writing of our initial study [162], we were not aware of any previous study on lurking in social networks from a

graph data management or mining perspective. Particularly, no computational method has been so far conceived to determine, and eventually, rank lurkers in an OSN graph. Note that, beyond the frequent yet trivial case of users that exhibit a peripheral unstructured membership, hidden forms of lurking are massively present in OSNs, which make it challenging to mine lurkers. While lurking is hard to track from a personal dispositional viewpoint, it appears that ranking lurkers is still possible by handling the situational variables that are related to the network of relationships between members. Moreover, a well-founded principle of eigenvector centrality, which is adopted in this work, will enable the determination of each node's lurking score in function of the lurking scores of the nodes that it is connected to.

One may notice that ranking influential people is clearly valuable as we naturally tend to follow leaders and learn from them, and conversely wonder *"why ranking lurkers?"*. We argue that scoring community members as lurkers, rather than limiting to solely recognize (potential or actual) lurkers, should be seen as essential to determine the contingencies in the network under which different lurking behaviors occur, and ultimately to aid devising both generic and ad-hoc de-lurking plans and strategies. In effect, ordering members by decreasing lurking score would enable to manage priority in de-lurking applications, to identify the sub-communities particularly affected by lurkers, and to define personalized triggers of active participation. For example, lurkers of a given sub-community developed around an entity of interest (e.g., a person, or theme) would welcome messages that highlight the key topics (a service that is already delivered to its users by Twitter, for example), social events that describe how to approach a discussion in a forum or to start off your own project in a collaboration network, or introduce the role of forum moderators or team leaders. Moreover, in order to alleviate information overload, which is recognized as a major negative factor for participation, various mechanisms of filtering (e.g., recommending threads of discussion, providing visual maps of the categories of activities) or promotion of lightweight contribution tasks (e.g., [53]) could be applied with the ultimate goal of revealing the lurker's value (i.e., ideas, opinions, expertise) to the community.

Contributions. In this chapter we address the new problem of mining lurkers in OSNs. We scrutinize the concept of lurking in OSNs to determine the essential criteria that can be taken as the basis for mining lurkers. We lay out a *topology-driven lurking* definition upon a network representation modeling the directed relationships from information-producer to information-consumer. Our lurking definition is based on three principles that respectively express in/out-degree related properties of a given node, its in-neighborhood, and its out-neighborhood. We also define a lurking coefficient to characterize the topology of a network in terms of lurking degree.

The proposed lurking definition lends itself naturally to score the users in an OSN according to their lurking behavior, thus enabling the development of ranking mechanisms. We hence focus on the problem of *lurker ranking*, and define three formulations of it that rely on the different aspects of our

topology-driven lurking concept. By resorting to classic link-analysis ranking algorithms, PageRank and alpha-centrality, we provide a complete specification of lurker ranking methods. We also propose a randomization-like model that simulates a mechanism of "self-delurking" of a network, and a lurking-oriented percolation analysis to unveil possible relations between lurkers and users that act as bridges over subnetworks.

We also extend our topology-driven definitions by devising time-aware lurker ranking methods which integrate different temporal aspects concerning both the production and consumption of information. More specifically, we measure temporal properties for individual users that account for the timeliness and the temporal evolution of the information produced (i.e., posts), and analogously we measure for every interaction between users, the timeliness and the temporal evolution of the information consumed by one user w.r.t. the other (i.e., comments, like/favorite-marks). Both types of temporal measures are the key ingredients in the proposed time-aware lurker ranking methods. Two different approaches to the ranking are followed that correspond to either a transient ranking, which is restricted to a particular snapshot graph of the network, or a cumulative ranking, which encompasses a sequence of snapshots based on a time-evolving graph model.

We conducted experiments on Twitter, Flickr, FriendFeed, GooglePlus and Instagram networks, analyzing both static and dynamic network contexts. For what concerns the static context, quantitative and qualitative results have shown the effectiveness of our lurker ranking approach, highlighting superior performance against PageRank, alpha-centrality and the Fair-Bets model, which conversely might fail to correctly identify and rank presumed lurkers. The experimentation on dynamic contexts provided insights into the understanding of lurkers from different perspectives along the time dimension in the SN environment, by addressing four research questions concerning the temporal dimension in the analysis of lurking behaviors in a SN; these questions span different topics concerning the relation between lurkers and inactive users, the relation between lurkers and active users, the responsiveness behavior of lurkers, and the evolution of lurking trends across time. Furthermore, evaluation of the ranking performance of time-aware lurker ranking methods on Flickr, FriendFeed, and Instagram network datasets have yielded interesting results, as our methods improved upon existing methods to which we compared them.

We also provide an investigation on the possibility to apply the proposed lurker ranking model to domains different than the classic OSNs. By applying the lurker rank methods in a social trust context we aim at understanding and quantifying relations between lurkers and trustworthy/untrustworthy users in ranking problems. We conduct this study by focusing purely on user behaviors that can be inferred from the network structure, using no content or contextual information. Results obtained on Advogato, Epinions, Flickr and FriendFeed networks indicate that lurkers should not be a-priori flagged as untrustworthy users, and that trustworthy users can indeed be found among lurkers. Concerning the domain of Research Collaboration Neworks (RCNs),

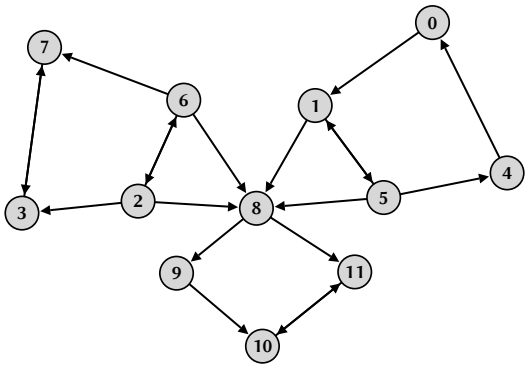


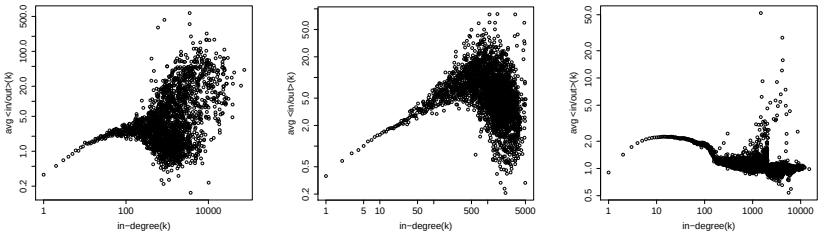
Fig. 5.1. An example OSN graph for our lurking-oriented ranking analysis.

we use our model to represent a vicarious-learner-oriented co-authorship network, introducing the concept of vicarious learner into research on analysis and mining in RCNs and providing a formal definition of vicarious learner. We conducted experiments using DBLP as case in point, as both a static and a dynamic network. Moreover, we also resorted to ArnetMiner for purposes of ground-truth evaluation of the competing methods. Quantitative as well as qualitative results have indicated the significance of the proposal.

The remainder of this chapter is organized as follows. Section 5.3 introduces our definitions of topology-driven lurking and lurking coefficient of a network. Our topology-driven lurker ranking methods are described in Section 5.4, while their time-aware extentions are described in Section 5.5. Section 5.6 presents methodology and results of the experimentation on static contexts, while in Section 5.7 the experimentation on dynamic contexts is discussed. Section 5.8 discusses the application of our lurker ranking model to other domains, e.g., trust networks and research collaboration networks. Section 5.9 discusses related work. Section 5.10 concludes the chapter.

5.3 In-degree, out-degree and lurking

User interactions in an OSN are typically modeled as influence relationships, whose varying strengths are used to determine and rank the influential users. In effect, ranking methods, such as PageRank, follow the conventional model of *influence graph*, which implies that the more incoming links a node has the more important or authoritative it is; for example, translated to Twitter terms, the more followers a user has, the more interesting his/her published tweets might be. Actually, as is well-known in spam detection, a node's in-degree can easily be affected by malicious manipulation, and hence the number of incoming links is not to be trusted as unique estimator of the node's importance score. Rather, as discussed in [59] in the Twitter scenario, the follower-to-



(a) *FriendFeed* (b) *GooglePlus* (c) *Twitter-UDI*

Fig. 5.2. Average in/out-degree as function of the in-degree k, on double-logarithmic scale. Sink and source nodes are discarded.

followee ratio should in principle be considered: if the number of followers exceeds those of followees then the user is likely to be an opinion-maker, otherwise her/his tweets are not that interesting.

We however observe that classic authority-based ranking methods (i.e., PageRank and related methods) cannot be directly applied to lurking analysis because they assume that links across users carry the meaning of node influence propagation, which is related to the *amount of information (number of walks) a node produces*. By contrast, lurking behaviors build on the *amount of information a node consumes*; again, in Twitter terms, if user v follows user u, then v is benefiting from u's information (i.e., v is receiving u's tweets).

A question might arise whether there is any evident correlation between the in/out-degree ratio and the in-degree distribution in an OSN graph. To roughly answer the question, we empirically investigated this aspect on the networks we used for our experimental evaluation (cf. Section 5.6.1); Figure 5.2 displays the average in/out-degree for each in-degree k, on some selected datasets. While the charts show substantially different trends, they all provide evidence on the poor correlation between in/out-degree ratio and the in-degree distribution. For the *FriendFeed* and *GooglePlus* cases, it can be observed a slightly upward trend for low in-degree values, while for *Twitter-UDI*, the initial uptrend rapidly decreases for low-mid in-degrees. All cases however present high dispersion of in/out-degrees for mid-high in-degrees.

5.3.1 Topology-driven lurking

Upon the in/out-degree ratio intuition, we now provide a basic definition of lurking which aims to lay out the essential hypotheses of a lurking status based solely on the topology information available in an OSN.

Definition 5.1 (Topology-driven lurking). *Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ denote the directed graph representing an OSN, with set of nodes (members) $\mathcal{V}$ and set of edges $\mathcal{E}$, whereby the semantics of any edge (u, v) is that v is consuming information produced by u. A node v with infinite in/out-degree ratio (i.e., a*

sink node) is trivially regarded as a lurker. A node v with in/out-degree ratio not below 1 shows a lurking status, whose strength is determined based on:

Principle I: *Overconsumption*. *The excess of information-consumption over information-production. The strength of v's lurking status is proportional to its in/out-degree ratio.*

Principle II: *Authoritativeness of the information received.* *The valuable amount of information received from its in-neighbors. The strength of v's lurking status is proportional to the influential (non-lurking) status of the v's in-neighbors.*

Principle III: *Non-authoritativeness of the information produced.* *The non-valuable amount of information sent to its out-neighbors. The strength of v's lurking status is proportional to the lurking status of the v's out-neighbors.*

To support this intuition, let us consider the example of network in Figure 5.1. Nodes $3, 7, 8, 10, 11$ have the highest in/out-degree ratio (i.e., 2), and as such they are candidate lurkers in the network. However, node 8 should be scored higher than others, since it benefits from information coming from two connected components, which are likely to contain influential nodes in the network (i.e., 5, 6). By contrast, nodes $10, 11$ should be scored as lurkers lower than node 8, since they are mainly fed by 8 itself; similarly, nodes $3, 7$ should be scored higher than $10, 11$ but lower than 8, since they receive information that propagates from a smaller subgraph. Note that the example allows us to shed light on a crucial aspect related to the role that node 8 has in the network. In effect, one may say that 8 is a "bridge" as it allows readers 9, 10, and 11 to peek into two otherwise separated communities. However, in our network model oriented to information consumption, the notion of bridge is also revised: the communication received from 9, 10, and 11 is likely to be less significant (in terms of amount and/or quality) than the bandwidth of information flow originated from the two largest components and received from 8. In Section 5.6.6, we shall investigate the relationship between lurkers and bridges, which will confirm that it's correct to regard node 8 as top-lurker.

5.3.2 Lurking Coefficient of a network

The participation inequality "1:9:90" rule loosely tells us that the majority of users shows a potential lurking behavior, in any generic online community. But, can we have a more precise indication of the presence of lurkers given a particular network? To answer this question we introduce here a measure, named *Lurking Coefficient*, as a basic lurking-related property of the topology of a network.

Given the directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ representing an OSN, for any node $i \in \mathcal{V}$ let $B_i = \{j | (j,i) \in \mathcal{E}\}$ and $R_i = \{j | (i,j) \in \mathcal{E}\}$ denote the set of in-neighbors (i.e., backward nodes) and out-neighbors (i.e., reference nodes) of i, respectively. The sizes of sets B_i and R_i are the in-degree and the out-degree

of i, denoted as $in(i)$ and $out(i)$, respectively. The *local Lurking Coefficient* of a node is first introduced to measure how likely any given node i is a lurker within its neighborhood. We define this quantity as:

$$lc_i = \frac{1}{|\mathcal{V}_i|}\left(\sum_{j\in B_i}\mathbb{1}\left\{\frac{in(j)}{out(j)} < \frac{in(i)}{out(i)}\right\} + \sum_{j\in R_i}\mathbb{1}\left\{\frac{in(j)}{out(j)} \geq \frac{in(i)}{out(i)}\right\}\right) \quad (5.1)$$

where $\mathcal{V}_i$ is the set of neighbors of i, and $\mathbb{1}\{A\}$ is the *indicator* function, which is equal to 1 when the event A is true, 0 otherwise. Note that the two additive terms in Eq. (5.1) are in accordance with Principle II and Principle III, respectively, of Def. 1. The *Lurking Coefficient* of a graph $\mathcal{G}$ is then given by the weighted average of the local Lurking Coefficients over the nodes in $\mathcal{G}$:

$$LC_{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} p_i \cdot lc_i \quad (5.2)$$

where p_i is the weight of lc_i. This weight, unitary by default, can be set in accordance with Principle I, hence it is defined as the in/out-degree ratio of i normalized over all nodes in its neighborhood. We will refer to the variant of LC with non-unitary weights as weighted Lurking Coefficient (wLC).

5.4 Lurker Ranking

In this section we formulate our solutions to the problem of lurker ranking. To this aim, we will capitalize on the three principles stated in our topology-driven lurking definition. Note that, as a general premise valid for all lurker ranking methods that we shall present, we introduce a Laplace smoothing factor in the calculation of both in-degree and out-degree of node, i.e., $in(i)$ (resp. $out(i)$) is meant hereinafter as the actual in-degree (resp. out-degree) of node i plus one. This allows us to deal with sink nodes and avoid infinite in/out-degree ratios.

According to Principle I in Definition 5.1, a basic way of scoring a node as a lurker is by means of its in/out-degree ratio. However, this way has clearly the disadvantage of assigning many nodes the same or very close ranks and, as we previously discussed, it ignores that the status of both the in-neighbors (Principle II) and out-neighbors (Principle III) contributes to the status of any given node. In the following we elaborate on each of those aspects separately.

In-neighbors-driven lurking. According to Principle II in Definition 5.1, an in-neighbors-driven lurking measure can be defined as:

$$r_i = \sum_{j \in B_i} \frac{out(j)}{in(j)} r_j$$

Hence, the score of node i increases with the number of its in-neighbors and with their likelihood of being non-lurkers, which is expressed by a relatively high out/in-degree. The above formula can be enhanced by including a factor that is inversely proportional to the i's out-degree. Formally, we define the *in-neighbors-driven lurking* score of node i as:

$$r_i = \frac{1}{out(i)} \sum_{j \in B_i} \frac{out(j)}{in(j)} r_j \tag{5.3}$$

Note that Eq. (5.3) accounts for both the contribution of a node's in-neighbors and its own in/out-degree property.

Out-neighbors-driven lurking. The exclusive contribution of out-neighbors for the calculation of a node's lurking score, according to Principle III of Definition 5.1, can be formalized as:

$$r_i = \sum_{j \in R_i} \frac{in(j)}{out(j)} r_j$$

However, this method would let the score of a node increase with the tendency of its out-neighbors of being lurkers, while ignoring the status of the node itself; as a consequence, not only reciprocal lurkers will be scored high but also every node from which lurkers receive information. A correction factor should hence be introduced as proportional to the in-degree of the target node. Formally, we define the *out-neighbors-driven lurking* score of node i as:

$$r_i = \frac{in(i)}{\sum_{j \in R_i} in(j)} \sum_{j \in R_i} \frac{in(j)}{out(j)} r_j \tag{5.4}$$

Note that in Eq. (5.4), the in-degree of node i is divided by the sum of in-degrees of its out-neighbors in order to score i higher if it receives more than what its out-neighbors receive.

In-Out-neighbors-driven lurking. The two previous definitions of lurking can in principle be combined to obtain an integrated representation of all three principles in Definition 5.1. To this aim, we define the *in-out-neighbors-driven lurking* score of node i as:

$$r_i = \left(\frac{1}{out(i)} \sum_{j \in B_i} \frac{out(j)}{in(j)} r_j \right) \left(1 + \left(\frac{in(i)}{\sum_{j \in R_i} in(j)} \sum_{j \in R_i} \frac{in(j)}{out(j)} r_j \right) \right) \tag{5.5}$$

Note that in Eq. (5.5) we have emphasized the aspect related to the strength of non-lurking behavior of in-neighbors, which is expected to have a better fit of the hypothetical likelihood function for a given node.

5.4.1 LurkerRank methods

We now define our lurker ranking methods, dubbed *LurkerRank* (for short LR), upon the previously defined lurking models. In order to provide a complete specification of our models, we resorted to the classic eigenvector-centrality schemes offered by *PageRank* [27] and *alpha-centrality* [23]. Note that while being widely applied to a variety of application domains with the purpose of scoring the influence or prestige in information networks, PageRank and alpha-centrality rely on different assumptions which make it worth the exploration of lurker ranking through both approaches.

Let us first recall the PageRank mathematics. The PageRank vector is the unique solution of the iterative equation $\mathbf{r} = \alpha\mathbf{S}\mathbf{r} + (1-\alpha)\mathbf{v}$. $\mathbf{S}$ denotes the column-stochastic transition probability matrix, which is defined as $(\mathbf{D}_{out}{}^{-1}\mathbf{A})^{\mathrm{T}} + \mathbf{e}\mathbf{a}^{\mathrm{T}}/|\mathcal{V}|$, where $\mathbf{A}$ is the adjacency matrix of the network graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, with $A_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, and $A_{ij} = 0$ otherwise; $\mathbf{D}_{out} = diag(\mathbf{A}\mathbf{e})$ is the out-degree diagonal matrix; $\mathbf{e}$ denotes a $|\mathcal{V}|$-dimensional column vector of ones; and $\mathbf{a}$ is defined such that $a_i = 1$ if node i has zero out-degree, and 0 otherwise. Vector $\mathbf{v}$ is typically defined as $(1/|\mathcal{V}|)\mathbf{e}$, but can be modeled to bias the PageRank to boost a specific subset of nodes in the graph. Term α is a real-valued coefficient ($\alpha \in [0, 1]$, commonly set to 0.85), which acts as a damping factor so that the random surfer is expected to discontinue the chain with probability $1-\alpha$, and hence to randomly select a page each with relevance $1/|\mathcal{V}|$ (teleportation).

We formulate three of our methods according to a PageRank-like scheme, i.e., at a high level, according to a combination of a random walk term with a random teleportation term. Our first LurkerRank method is named *in-neighbors-driven LurkerRank* (hereinafter denoted as LRin) since it is built upon Eq. (5.3):

$$r_i = \alpha \left(\frac{1}{out(i)} \sum_{j \in B_i} w(j,i) \frac{out(j)}{in(j)} r_j \right) + \frac{1-\alpha}{|\mathcal{V}|} \tag{5.6}$$

Note that with Eq. (5.6), we introduce edge weights to deal with weighted graphs as well, for the sake of generality; although, as in our experimental setting, they are set as unitary by default. Analogously, the *out-neighbors-driven LurkerRank* (hereinafter denoted as LRout) is defined as:

$$r_i = \alpha \left(\frac{in(i)}{\sum_{j \in R_i} in(j)} \sum_{j \in R_i} w(i,j) \frac{in(j)}{out(j)} r_j \right) + \frac{1-\alpha}{|\mathcal{V}|} \quad (5.7)$$

Finally, the *in-out-neighbors-driven LurkerRank* (hereinafter denoted as LRin-out) is defined as:

$$r_i = \alpha \left[\left(\frac{1}{out(i)} \sum_{j \in B_i} w(j,i) \frac{out(j)}{in(j)} r_j \right) \left(1 + \left(\frac{in(i)}{\sum_{j \in R_i} in(j)} \sum_{j \in R_i} w(i,j) \frac{in(j)}{out(j)} r_j \right) \right) \right] + \frac{1-\alpha}{|\mathcal{V}|} \quad (5.8)$$

Alpha-centrality [23] expresses the centrality of a node as the number of paths linking it to other nodes, exponentially attenuated by their length. Moreover, it takes into account the possibility that each node's status may also depend on information that comes from outside the network or that may regard solely the member. Alpha-centrality is defined as $\mathbf{r} = \alpha \mathbf{A}^{\mathrm{T}} \mathbf{r} + \mathbf{v}$, where $\mathbf{v}$ is the vector of exogenous source of information ($\mathbf{v} = \mathbf{e}$ as default), and α here reflects the relative importance of endogenous versus exogenous factors in the determination of centrality. High values of α (e.g., 0.85) make the close neighborhood contribute less to the centrality of a given node. The rank obtained using alpha-centrality can be considered as the steady state distribution of an information spread process on a network, with probability α to transmit a message or influence along a link.

We will denote our alpha-centrality based LurkerRank methods with prefix ac- to distinguish them from the PageRank-based counterparts. The alpha-centrality-based in-neighbors-driven LurkerRank (ac-LRin) is defined as:

$$r_i = \alpha \left(\frac{1}{out(i)} \sum_{j \in B_i} w(j,i) \frac{out(j)}{in(j)} r_j \right) + 1 \quad (5.9)$$

Analogously, other two methods, denoted as ac-LRout and ac-LRin-out, are defined according to the out-neighbors-driven and in-out-neighbors-driven lurking models, respectively.

Figure 5.3 compares the rankings obtained by our LRin and basic PageRank on the example network of Figure 5.1 (α set to the default 0.85). Using LRin, node 8 was ranked highest (0.146), followed by 3 and 7 (0.112), and then 11 (0.094), 10 (0.088): this sheds light on the ability of LRin to match our definition of lurking (cf. discussion about Fig. 5.1 in Section 5.3). By contrast, PageRank ranked first nodes 10 and 11 (both around 0.256), and then 3 and 7 with a significant gap in score from the first two (0.116), followed by

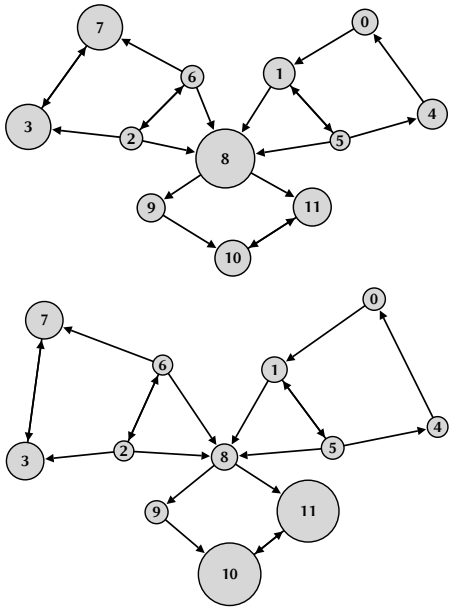


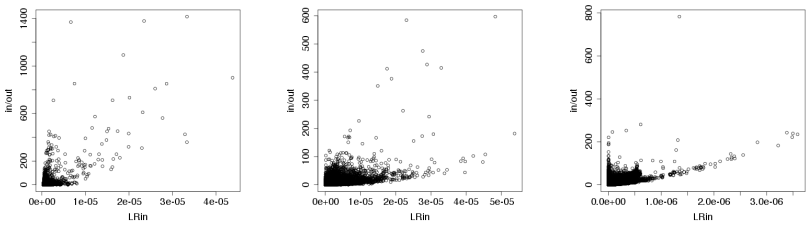
Fig. 5.3. Lurker ranking in the example OSN graph of Fig. 5.1: LRin (on top) versus PageRank (on bottom). Nodes are sized proportionally to their ranking scores.

8 (0.052), 1 (0.048); moreover, node 5 was ranked eighth, despite it is a major feeder of the lurker 8, while it was correctly ranked lowest by LRin. Similarly, alpha-centrality (results not shown) did not fare well as it ranked first nodes 11 (0.317) and 10 (0.308), before ranking node 8 (0.095), and nodes 3 and 7 in ninth and tenth position both with a score of 0.004.

5.4.2 Limit $\alpha \to 0$ of the LR functions

We investigate the behavior of LR functions to understand whether LR rank can be reduced to either the in/out-degree or the out/in-degree rank as α approaches 0. We take the LRin functional form as case in point, while analogous conclusions can be drawn for the other LR functions.

In the extreme case $\alpha = 0$, the LRin score of each vertex is equal to $1/|\mathcal{V}|$. If $\alpha \approx 0$, then $(1 - \alpha) \to 1$, therefore we write $1 - \alpha = 1 - \epsilon$, with $\epsilon \ll 1$, and $r_i \approx 1/|\mathcal{V}|$. Substituting these into Eq. (5.6), with unitary edge weights for the sake of simplicity, we have:



(a) *Flickr* (b) *FriendFeed* (c) *Twitter-Kwak*

Fig. 5.4. LRin rank versus in/out-degree rank. Damping factor α is set to 0.01. Sink and source vertices are discarded.

$$r_i = \epsilon \left(\frac{1}{out(i)} \sum_{j \in B_i} \frac{out(j)}{in(j)} r_j \right) + \frac{1-\epsilon}{|\mathcal{V}|} \approx \frac{1}{|\mathcal{V}|} \left[1 + \epsilon \left(\frac{1}{out(i)} \sum_{j \in B_i} \frac{out(j)}{in(j)} - 1 \right) \right] \quad (5.10)$$

A crucial part in Eq. (5.10) is the estimation of the sum. This term would be estimated as proportional to the in/out-degree of vertex i and to the average out/in-degree $\langle \frac{out}{in} \rangle$:

$$r_i \approx \frac{1}{|\mathcal{V}|} \left[1 + \epsilon \left(\frac{in(i)}{out(i)} \left\langle \frac{out}{in} \right\rangle - 1 \right) \right] \quad (5.11)$$

The above approximation is however admissible only if a relatively small dispersion can be assumed to hold for the out/in-degree distribution. Unfortunately, in all our evaluation network datasets (cf. Sect. 5.6.1), this does not seem the case since the out/in-degree distribution is always found to be less narrow than the corresponding in/out-degree distribution, as reported in Table 5.1. This would indicate that in principle LRin rank distribution is likely not to follow exactly the same trend as that of in/out-degree as $\alpha \approx 0$. In effect, although a moderate to strong positive correlation may still occur — 0.568 on *FriendFeed* (Fig. 5.4(b)), 0.674 on *Twitter-UDI*, 0.679 on *Flickr* (Fig. 5.4(a)), 0.686 on *Twitter-Kwak* (Fig. 5.4(c)), and 0.745 on *GooglePlus* — Fig. 5.4 shows that top-ranked vertices by LRin often do not correspond to top-ranked in/out.

5.5 Time-aware LurkerRank methods

In this section we describe our extensions to the LurkerRank algorithm, in order to account for the temporal dimension when determining the lurking

Table 5.1. Mean and standard deviation values of in/out-degree and out/in-degree.

	in/out *		in/out **		out/in *		out/in **	
	mean	*sd*	*mean*	*sd*	*mean*	*sd*	*mean*	*sd*
Flickr	1.096	3.377	2.731	10.557	1.263	4.583	5.554	20.085
FriendFeed	1.664	5.693	10.359	15.353	8.682	71.771	63.269	219.387
GooglePlus	3.947	11.200	24.350	27.665	3.739	46.235	27.984	144.051
Twitter-Kwak	2.647	3.863	11.662	9.442	1.263	46.078	6.910	145.665
Twitter-UDI	1.541	1.530	5.517	3.582	1.202	15.758	4.946	50.321

* Sink nodes and source nodes are discarded. ** Like *, but only 90th percentile is considered.

scores of users in the network. We follow two approaches based on different models of temporal graph:

- *Transient ranking*, i.e., a measure of a node's lurking score based on a time-static (snapshot) graph model;
- *Cumulative ranking*, i.e., a measure of a node's lurking score that encompasses a given time interval (sequence of snapshots), based on a time-evolving graph model.

The building blocks of our methods rely on the specification of the temporal aspects of interest, namely *freshness* and *activity trend*, both at user and user relation levels. Freshness takes into account the timestamps of the latest information produced (i.e., post) by a user, or the timestamps of the latest information consumed by a user (i.e., comment, like) in response to another user's post. Activity trend models how the activity of posting of a user, or the activity of responding by user to another one, varies over time. We will now elaborate on these concepts.

5.5.1 Freshness and activity trend functions

Users in the network are assumed to make actions and interact with each other over a time span $\mathcal{T} \subseteq \mathbb{T}$. The temporal domain $\mathbb{T}$ is conveniently assumed to be $\mathbb{N}$. Therefore, the time-varying graph of a SN is seen as a discrete time system, i.e., the time is discretized at a fixed granularity (e.g., day, week, month).

Freshness. Let $T \subseteq \mathcal{T}$ be a temporal subset of interest, being in interval notation of the form $T = [t_s, t_e]$, with $t_s \leq t_e$. For any time t, we define the *freshness function* $\varphi_T(t)$ as:

$$\varphi_T(t) = \begin{cases} 1/\log_2(2 + (t_e - t)), & \text{if } t \in T. \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

Function $\varphi_T(t)$ ranges within $[0, 1]$. Note that we opt for a function with logarithmic decay to ensure, as $(t_e - t)$ gets larger, a slower decrease w.r.t. other decreasing functions with values in $(0, 1]$—for instance, the graph of $\varphi_T(t)$ lies always above the graph of $2/(1 + \exp(t_e - t))$, or of $1/(1 + (t_e - t))$.

Given a user u, let T_u be the set of time units at which u performed actions in the network. The *freshness* of u at a given temporal subset of interest T is defined as:

$$f_T(u) = \max\{\varphi_T(t),\ t \in T_u \text{ s.t. } t_s \leq t \leq t_e\} \tag{5.13}$$

Note that $f_T(u)$ is always defined, non-negative, and increases as more recent is the latest activity of u w.r.t. T.

Activity trend. The second aspect we would like to model is the activity trend of a user. Let $S_u = [(x_1, t_1), \ldots, (x_n, t_n)]$ be the time series representing the activity of user u over T_u. For every pair $(x, t) \in S_u$, x denotes the number of u's actions at time t.

In [27], the *Derivative time series Segment Approximation* (DSA) model is proposed to represent time series into a concise form which is designed to capture the significant variations in the time series profile. For any given time series S of length n, DSA produces a new series τ of h values, with $h \ll n$. The main steps performed by DSA are summarized as follows:

- *Step 1 - Derivative estimation:* S is transformed into S', where each value $x \in S$ is replaced by its first derivative estimate.
- *Step 2 - Segmentation:* the derivative time series S' is partitioned into h variable-length segments. Each of the segments aggregates subsequent data values having very close derivatives, i.e., it represents a subsequence of values with a specific trend.
- *Step 3 - Segment approximation:* each of the segments in S' is mapped to an angular value α, which collapses information on the average slope within the segment.

The DSA series τ is of the form $\tau = [(\alpha_1, t_1), \ldots, (\alpha_h, t_h)$, such that $\alpha_j = \arctan(\mu(s_j))$ and $t_j = t_{j-1} + l_j$, with $j = [1..h]$, where s_j is the j-th segment, l_j its length, and $\mu(s_j)$ the mean of its points.

We apply DSA to each time series S_u in order to model the temporal evolution of u's activity. As a post-processing step, the values α_j of the DSA sequence τ_u are normalized within [0,1] as $\hat{\alpha_j} = \alpha_j/\pi + 1/2$, i.e., an increasing (resp. decreasing) trend of activity will correspond to an augmenting ($(0.5, 1]$), resp. penalty ($[0, 0.5)$), factor. Then, we define the *activity trend* of user u (over T_u) as the time sequence:

$$a(u) = [(\hat{\alpha}_1, t_1), \ldots, (\hat{\alpha}_h, t_h)] \tag{5.14}$$

Given a of interest T, the activity trend of u w.r.t. T corresponds to the subsequence $a_T(u)$ of $a(u)$ that best fits T.

Freshness and activity trend of interaction. The notions of freshness and activity trend for individual users are here extended to model the interaction of any two users u, v at a given time t, which corresponds to the directed edge (u, v) in the graph snapshot containing t. The rationale here is

that the more recent is an interaction (or the more increasing is its activity trend), the better should be weighted the edge. Recall that (u, v) means that v is consuming (i.e., commenting/"like"-ing) information at time t produced by u.

Let us denote with $P_u = \{p_1, \ldots, p_k\}$ the set of information-production actions (posts) of u, and with $T_u(P_u) = \{t_{p_1}, \ldots, t_{p_k}\}$ the associated timings. Moreover, let $C_{u \to v}$ be a set of triplets $(t_{p_i}, t_{c_j}, x_{c_j})$, such that $t_{p_i} \in T_u(P_u)$, and t_{c_j}, x_{c_j} denote the time and the frequency, respectively, at which v consumed (i.e., responded to) the u's post p_i.

According to the above formalism, we define the *freshness of interaction* $u \to v$ w.r.t. T as the maximum freshness over the sequence of pairs *(post-time, response-time)* in T:

$$f_T(u, v) = \max\{\varphi_{[t_p, t_c]}(t_p), \\ \text{with } t_c, t_p \; s.t. \exists (t_p, t_c, \_) \in C_{u \to v} \wedge t_s \leq t_p, t_c \leq t_e\} \quad (5.15)$$

Analogously, we model the activity trend of interaction, based on DSA model previously used to define the activity trend of user. To this end, given the interaction $u \to v$, we consider the time series $S_{u,v}$ representing pairs (x, t), where x denotes the number of actions at time t performed by v in response to a specific post by u. Then, we compute the *activity trend of interaction* $u \to v$, denoted with $a_T(u, v)$, as the result of the application of DSA to the time series $S_{u,v}$.

5.5.2 The Time-static LurkerRank algorithm

Our first formulation of time-aware LurkerRank is based on a time-static graph model, which contains one single snapshot of the network. Our key idea is to capitalize on the previously proposed functions of freshness and activity trend to define a time-aware weighting scheme that determines the strength of the interaction between any two users linked at a given time.

Given a temporal interval of interest T, we define the weighting function $w(\cdot, \cdot)$ in Eq. (5.8) as a convex combination of the functions of freshness and activity trend previously defined, plus one. Formally, for each $u, v \in \mathcal{V}$:

$$w_T(u, v) = 1 + (\omega_1 \; f_T(u) + \omega_2 \; a_T(u) + \\ + \omega_3 \; f_T(u, v) + \omega_4 \; a_T(u, v)) \quad (5.16)$$

where $\omega_i \geq 0$, and $\sum_i \omega_i = 1$, with $i \in [1..4]$. We will hereinafter denote as Ts-LR the *time-static LurkerRank* algorithm which is equipped with the above weighting function.

5.5.3 The Time-evolving LurkerRank algorithm

The time-static LurkerRank can work only on a subset of relational data that are restricted to a particular subinterval of the network timespan. Therefore,

information on the sequence of events concerning users' (re)actions is lost as relations are aggregated into a single snapshot. To overcome this issue, we define here an alternative weighting scheme for our LR algorithm that is able to model, for each target user v, the potential accumulated over a time-window of the contribution that each in-neighbor u had to the computation of the lurking score of v.

Cumulative freshness and activity functions

We begin with the definition of a *cumulative scoring function* which forms the basis for each of the subsequent functions that will apply to the previously defined freshness and activity trend of node/interaction. Intuitively, this cumulative scoring function ($\mathrm{g}_{\leq}$) should be defined at any time $t \in T$ to aggregate all values of a function g (defined in T) computed at times t' less than or equal to t, following an exponential-decay model:

$$\mathrm{g}_{\leq}(t) \propto \mathrm{g}(t) + \sum_{t'<t} (1 - 2^{t-t'})\mathrm{g}(t') \tag{5.17}$$

Let the timespan $\mathcal{T}$ of the network graph be partitioned in consecutive sub-intervals $T_1, T_2, \ldots, T_i, \ldots = [t_0, t_1], (t_1, t_2], \ldots, (t_{i-1}, t_i] \ldots$. The generic cumulative scoring function $\mathrm{g}_{\leq}(\cdot)$ has a straightforward translation in terms of user-freshness: if t_i corresponds to the end-time of the span of interest whose latest sub-interval is T_i, we define the *cumulative user-freshness* function applied to any user u to integrate (with exponential decay) all user-freshness values individually obtained at each sub-interval preceding t_i:

$$cf_{T_i}(u) = f_{T_i}(u) + \sum_{t_k<t_i} (1 - 2^{t_i-t_k}) f_{T_k}(u) \tag{5.18}$$

Our *cumulative user-activity trend* function, we denote with $ca_{T_i}(\cdot)$, has similar form:

$$ca_{T_i}(u) = \mu(a_{T_i}(u)) + \sum_{t_k<t_i} (1 - 2^{t_i-t_k}) \mu(a_{T_k}(u)) \tag{5.19}$$

where $\mu(a_T(u))$ denotes the average of the values contained in the activity trend series $a_T(u)$. The definition of cumulative freshness of interaction, $cf_{T_i}(u, v)$, and cumulative activity trend of interaction, $ca_{T_i}(u, v)$, at each T_i, follow intuitions analogous to Eq. (5.18) and Eq. (5.19), respectively.

The *time-evolving LurkerRank* algorithm, hereinafter denoted as Te-LR, will be equipped with a weighting function analytically similar to that defined in Eq. (5.16). Formally, for each $u, v \in \mathcal{V}$ and sub-interval T:

$$cw_T(u, v) = 1 + (\omega_1 \, \widehat{cf}_T(u) + \omega_2 \, \widehat{ca}_T(u) + \\ + \omega_3 \, \widehat{cf}_T(u, v) + \omega_4 \, \widehat{ca}_T(u, v)) \tag{5.20}$$

where $\omega_i \geq 0$, $\sum_i \omega_i = 1$, with $i \in [1..4]$, and the $\widehat{\ }$ symbol means that each of the four functions is normalized in [0,1].

Table 5.2. Main structural characteristics of the evaluation network datasets.

data	*# nodes*	*# links*	*avg in-degree*	*avg path length*	*clustering coefficient*	*assortativity*	*# sources* / *# sinks*	*LC* / *wLC*
Flickr	2,302,925	33,140,018	14.39	4.36*	0.107	0.015	360,416 57,424	0.573 0.248
FriendFeed	493,019	19,153,367	38.85	3.82	0.029	-0.128	41,953 292,003	0.955 0.354
GooglePlus	107,612	13,673,251	127.06	3.32	0.154	-0.074	35,341 22	0.869 0.096
Twitter-Kwak	16,009,364	132,290,000	8.26	5.91*	1.26E-4	-0.095	1,067,936 10,298,788	0.914 0.435
Twitter-UDI	24,984,590	284,884,500	11.40	5.45*	4.96E-3	-0.297	3,380,805 8,065,287	0.790 0.470

* Value estimated as $(\log(|\mathcal{V}|))/\log(2|\mathcal{E}|/|\mathcal{V}|)$.

5.6 Experimentation on static contexts

We present in this section our experimentation over static graphs. We first describe the data we used and the assessment criteria selected for the analyses. Then we begin presentation of our results with an analysis of reciprocity and attachment behaviors of lurkers. Section 5.6.4 is devoted to present quantitative results on the ranking performance obtained by the proposed and competing methods. In Section 5.6.5, we introduce a randomization-like model to study how to support "self-delurking" of a network, whereas in Section 5.6.6 we present a lurking-oriented percolation analysis. Finally, in Section 5.6.7, we provide a qualitative insight into the methods' ranking behavior.

Notations: Here we briefly recall main notations that will be used throughout this section. LR and ac-LR prefixed abbreviations refer to our proposed LurkerRank methods (cf. Section 5.4.1). The following notations are abbreviations for the competing methods (cf. Section 5.6.2): IO stands for in/out-degree ratio ranking; PR, PR, and FB stand for PageRank, alpha-centrality, and Fair-Bets model, respectively. Moreover, DD symbols refer to data-driven rankings.

5.6.1 Data

We used five OSN datasets for our static evaluation, namely *Twitter* (with two different dumps), *Flickr*, *FriendFeed*, and *GooglePlus*:

- From the Twitter dump studied in [93], which we will refer to as *Twitter-Kwak*, we extracted the follower-followee topology starting from a connected component of one hundred thousands of users and their complete neighborhoods. A partial copy of the tweet data used in [93] was exploited to define a Twitter-based data-driven ranking and also to perform a qualitative evaluation on *Twitter-Kwak*, as we shall describe in Section 5.6.2.
- The *Twitter-UDI* dataset [105] was originally collected in May 2011, hence it's more recent and also larger than *Twitter-Kwak*. Tweet data however could not be exploited for our analysis since they are available only for a

very small subset of users in *Twitter-UDI* (less than 0.6%) and they are also upper-bounded (limit of 500 tweets per user) [105].
- We used the entire *Flickr* data studied in [124], originally collected in 2006-2007. Information on the number of views and number of favorite markings every photo had, was exploited for our definition of Flickr-based data-driven ranking.
- We used the latest version of the *FriendFeed* dataset studied in [35]. Due to the recognized presence of spambots in this OSN dataset, we filtered out users with an excessive number of posts (above 20 posts per day) as suggested in [35].
- *GooglePlus* dataset was originally studied in [116], and consists of *circles* from GooglePlus. The dataset was collected from users who had manually shared their circles using the *share circle* feature, and the topology was built by combining the edges from each node's ego network.

Beyond the complexity of their technical and sociological aspects, the five networks have been selected since they naturally provide asymmetric relationships — recall that in our setting, a link from user i to user j means that j is a follower or subscriber of i — and also because they offer a variety of topological properties, as shown in Table 5.2. The table also reports each network's Lurking Coefficient (LC), in the upper row, and weighted LC (wLC), in the bottomer row (cf. Section 5.3.2). Notably, a high LC (ranging from about 0.8 to 0.95) was found for all networks except for *Flickr*: this may prompt us to suppose that lurkers would not characterize Flickr as much as other OSNs; in effect, differently from the other selected networks, users would subscribe and join the Flickr community when they are willing to upload and share their photos, thus showing a normal attitude to participate. Moreover, the lower value of weighted LC that characterizes *GooglePlus* could be explained due to a clustering coefficient, along with variation of in/out degree (Table 5.1), exhibited by this network, which are both relatively higher than in the other ones. Yet, note that the values of assortativity reported in Table 5.2 are always negative or close to zero, which would indicate no tendency of vertices with similar degree to connect to each other; interestingly, *Twitter-UDI* which has the most negative degree of assortativity, has also the largest value of weighted LC.

5.6.2 Assessment methodology

Competing methods and notations

We compared our proposed methods against PageRank (henceforth PR), alpha-centrality (henceforth AC), and Fair-Bets model [31] (henceforth FB). The latter method was included in the comparative evaluation as it also exploits the notion of in/out-degree ratio to rank users, which is seen as a fair-bets model of social capital accumulation and expenditure; originally con-

ceived to rank players in round-robin tournaments, the Fair-Bets model assumes that users are paying each other to accept invitations on an online community, then the fair bets score of a user is the amount she/he can afford to pay on average. Fair-Bets computes the score of any node i as

$$r_i = \frac{1}{out(i)} \sum_{j \in B_i} r_j$$

Finally, we included in the evaluation the in/out-degree distribution of the nodes in a network dataset, as a baseline method (henceforth IO).

Data-driven evaluation

Given the novelty of the problem at hand, we had to cope with an issue relating to the lack of ground-truth data for lurker ranking. In the attempt of simulating a ground-truth evaluation, we generated a *data-driven ranking* (henceforth DD) for a network dataset and used it to assess the proposed and competing methods.

On *Twitter-Kwak*, we calculated the score of a node as directly proportional to its in/out-degree (Laplace add-one smoothed, cf. Section 5.4) and inversely exponentially with a Twitter-specific measure of influence:

$$r_i^* = \frac{in(i)}{out(i)} \exp(-EI(i))$$

$EI(\cdot)$ denotes the *empirical measure of influence* [14] which is used to estimate the influence of a user based on the amount of information s/he posted (i.e., tweets) and that her/his followers have retweeted. For a user i,

$$EI(i) = \frac{1}{out(i)} \sum_{j \in R_i} nRetweets(j)$$

where $nRetweets(j)$ is the number of retweets by follower j. Note that, as found in [93], a ranking based on retweets differs from that based on the number of followers, and this prompted us to combine the two aspects in our data-driven ranking.

We defined an analytically similar function for the *FriendFeed* data-driven ranking, in which the *empirical measure of influence* has been redefined as:

$$EI(i) = \left(\frac{1}{out(i)} \sum_{j \in R_i} nCom(j,i) \right) \log_{10}(nPosts(i) + 10)$$

where $nCom(j,i)$ is the number of comments from user j to posts by user i, and $nPosts(i)$ is the total number of posts by user i. Note that this combination of indicators of user's activity with user's influence was needed since only a

Table 5.3. Reciprocity and lurking. *rle* is the number of reciprocal lurking edges (i.e., reciprocal edges in the lurking-induced network graph) divided by the total number of edges in the original graph.

		top-25% of the LRin-out solution			top-10% of the LRin-out solution			top-5% of the LRin-out solution		
	# recip. edges (full graph)	*# edges (induced graph)*	*# reciprocal lurking edges*	*% rle*	*# edges (induced graph)*	*# reciprocal lurking edges*	*% rle*	*# edges (induced graph)*	*# reciprocal lurking edges*	*% rle*
Flickr	20,603,483	23,352,367	16,440,872	49.61	12,349,595	8,704,922	26.27	5,030,759	3,192,712	9.63
FriendFeed	3,014,306	340,935	33,654	0.18	1,096	46	<0.01	2	0	0.00
GooglePlus	2,870,336	1,413,468	667,422	4.88	49,481	23,562	0.17	5,310	2,624	0.02
Twitter-Kwak	52,137,192	7,293	2,806	<0.01	216	52	<0.01	64	10	<0.01
Twitter-UDI	191,858,256	18,839,845	10,078,339	3.54	3,094,341	1,198,615	0.42	872,332	271,751	0.10

limited portion (below 10%) of users in *FriendFeed* had information on the number of received comments.

For *Flickr* we produced two data-driven rankings, dubbed DD-F and DD-V. While still related to the in/out degree as for the previously defined DD, we used the number of *favorites* (DD-F), or alternatively the number of *views* (DD-V), received by a user's photos to set the exponent (with negative sign) in the data-driven ranking function.

Unfortunately, for both *Twitter-UDI* and *GooglePlus* we were unable at the time of this writing to gather adequate information to produce a data-driven ranking, also due to the restrictive usage limits of both networks APIs. Note that the information used to generate DD for *Twitter-Kwak* was substantially incomplete and obsolete to be used for *Twitter-UDI*.

Assessment criteria

In order to comparatively evaluate our proposed methods' performance with respect to the competing methods, we resorted to well-known assessment criteria, namely *Kendall tau rank correlation coefficient* [1], *Fagin's intersection metric* [52] and *Bpref* [29] (cf. Chapter 2, Section 2.7).

For what concerns *Bpref* setting, we first determined the set of judged irrelevant candidates N as the set of nodes with data-driven ranking score below or equal to 1, and used it for comparisons with DD, when available; whereas, for comparisons among competing methods, N was defined as either the bottom of the corresponding method's ranking having the same size as N in the data-driven ranking, or (when DD is not available) as the bottom-25% of the method's ranking. The set of judged relevant candidates R was selected as the set of nodes having top-l% score from the complement of N.

Both F and $Bpref$ are within $[0, 1]$, whereby values closer to 1 correspond to better scores. For the experiments discussed in the following, we setup the size k of the top-ranked lists for Fagin evaluation to $k = 10^2, 10^3, 10^4$, and the l% of relevant candidates for Bpref evaluation to $l = 10, 25, 50$ (i.e., relevant candidates in the 90th percentile, the third quartile and the median). Moreover, unless otherwise specified, F scores will correspond to ranking lists without sink nodes, in order to avoid biasing (presumably overstating) our evaluation with trivial lurkers.

5.6.3 Lurker reciprocity and attachment

We aimed at understanding two different aspects of the lurking behaviors: (1) how lurkers relate to each other, in terms of *link reciprocity*, and (2) how lurker distribution grows with respect to active users, which can be explained in terms of *attachment* mechanisms.

Reciprocity

We examined the impact of the presence of lurkers on measures of reciprocity in the various network graphs, under three different settings that correspond to the top-25%, top-10% and top-5%, respectively, of a LR ranking solution. Specifically, we considered four measures of reciprocity, namely (i) the number of reciprocal lurking edges (i.e., reciprocal edges in the lurking-induced network graph), (ii) the percentage of reciprocal lurking edges to the total number of edges in the original graph (denoted as *rle*), (iii) the fraction of reciprocal edges in the original network graph that connect lurkers to each other, and (iv) the fraction of edges that connect lurkers to each other within a lurking-induced subgraph.

Table 5.3 reports results obtained by the LRin-out method. A first remark is that *rle* was very small or negligible regardless of the portion of LR ranking solution considered. An exception was represented by *Flickr*, whose *rle* varied from about 50% to 10%; this could be explained as an effect of the crawling mechanism used to build the *Flickr* network dataset, since unlike the other datasets, it was obtained starting from a single seed user and then performing a breadth-first search on the social network graph. Considering the fraction of reciprocal edges in the original network graph that connect lurkers to each other (results not shown), again with the exception of *Flickr* we observed a very small value even for the case of top-25% lurkers (around 23% for *Google-Plus*, 5% for *Twitter-UDI*, and below 1% for *FriendFeed* and *Twitter-Kwak*), while approaching zero when the top-ranked solution is narrowed to 10% or smaller.

Note that, while LRin behaved very similarly to LRin-out, results obtained by LRout showed that *rle* values were significantly higher than those observed in Table 5.3, with averages over the datasets equal to 35% (top-25%), 27% (top-10%), and 20% (top-5%). Even higher were the values of the fraction of reciprocal edges in the original network graph connecting lurkers, with peaks above 90% in the top-25% case, and averages of 85% (top-25%), 63% (top-10%), and 45% (top-5%). These findings were actually not surprising since LRout is designed to emphasize the lurking attitude of any node from which a target node receives information.

Figure 5.5, as complementary to Table 5.3, shows the fraction of edges that connect lurkers to each other within a lurking-induced subgraph. In the figure, we also included for comparison the case of "potential lurkers", regarding them as those nodes having in/out-degree ratio above 1. An evident remark is that the reciprocity between lurkers generally followed a decreasing trend varying

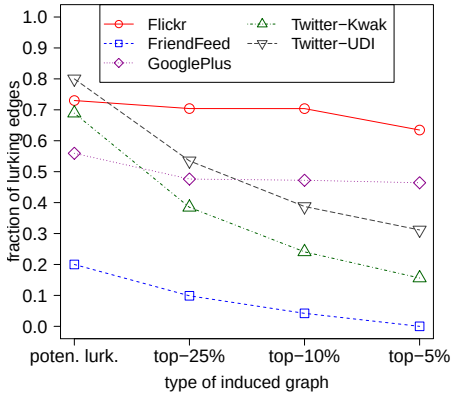


Fig. 5.5. Fraction of reciprocal edges in the lurking-induced subnetworks.

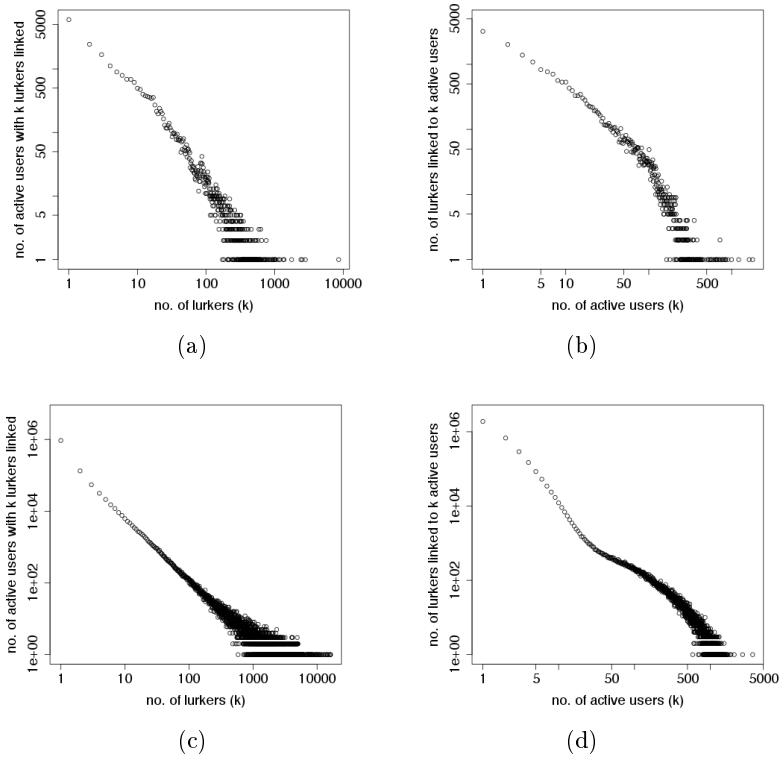


Fig. 5.6. Distribution of active users as a function of the lurkers-followers (a)-(c) and distribution of lurkers as a function of the active users-followees (b)-(d). (*GooglePlus*, two plots from the left, and *Twitter-UDI*, two plots from the right).

from the "potential lurkers" to the top-5% setting; this trend was quite slow or roughly stagnant on three out of five datasets (i.e., *Flickr*, *GooglePlus*, and *FriendFeed*) but much sharper in the two largest networks (i.e., the two Twitter datasets). Interestingly, when considering LRout instead of LRin-out or LRin, the fraction of reciprocal edges in the lurking-induced subgraph was in general not longer observed as a decreasing function by decreasing sizes of lurker sets; the trend was rather increasing for the Twitter datasets (upper values of 0.78 for *Twitter-UDI* and 0.60 for *Twitter-Kwak*) and for *FriendFeed* (upper value of 0.32).

Attachment

We focus now on the relation between lurkers and the "active" users they are linked to. Specifically, we analyzed the distribution of lurkers as function of the degree of attached active users, and dually for the distribution of active users. For this analysis, we selected the same fraction (25%) from the top and from the bottom of the LRin-out ranking solution in order to choose the set of lurkers and the set of active users, respectively, under examination.

Our goal was to understand whether the probability of observing active users with a certain degree of attached lurkers, and vice versa, can be predicted by a power law. Therefore, for each dataset, we learned the best fit of a power law distribution to the observed data, where the statistical significance of this fitting was assessed based on a Kolmogorov-Smirnov test. The resulting plots obtained on our datasets showed a power law behavior for both the distribution of lurkers (following k active users) and the distribution of active users (followed by k lurkers); Figure 5.6 shows the plots for *GooglePlus* and *Twitter-UDI*. The exponent of the fitted power law distributions varied from 1.67 (*GooglePlus* and *Twitter-Kwak*) to 2 (*FriendFeed*, *Twitter-UDI*), for the distribution of active users, and from 1.36 (*Flickr*, *Twitter-Kwak*) to 1.86 (*GooglePlus*), for the distribution of lurkers. Significant fitting was actually found in general for both distributions in each dataset, which would indicate that they may follow a preferential attachment mechanism: active users, who already are followed by a large number of lurkers, are likely to attract even more lurkers; analogously, lurkers, who already follow a large number of active users, are more likely to do so. Moreover, as smaller values of the Kolmogorov-Smirnov statistic denote better fit, we observed a slight tendency of better explaining the growing of the number of lurkers (rather than of active users) by preferential attachment on *GooglePlus* and *Twitter-UDI*, while an opposite situation was found on *Flickr*.

5.6.4 Ranking evaluation

Correlation analysis with data-driven rankings

Table 5.4 shows the Kendall tau rank correlation obtained by our LurkerRank methods and by the competing methods with respect to the data-driven ranking (DD) for all eligible datasets.

Table 5.4. Comparative performance of LurkerRank methods and competitors with respect to data-driven rankings: Kendall tau rank correlation values (with 95% confidence intervals in parentheses).

dataset	IO	PR	AC	FB
FriendFeed	.169 (± .003)	.128 (± .004)	.230 (± .005)	.373 (± .004)
Flickr vs DD-V	.046 (± .008)	.043 (± .005)	.043 (± .008)	.047 (± .002)
Flickr vs DD-F	.052 (± .007)	.049 (± .005)	.049 (± .008)	.053 (± .002)
Twitter-Kwak	.171 (± .006)	*.004* (± .011)	.215 (± .010)	.235 (± .012)

dataset	LRin	LRout	LRin-out	ac-LRin	ac-LRout	ac-LRin-out
FriendFeed	.661 (± .003)	-.169 (± .005)	.497 (± .003)	**.664** (± .003)	-.189 (± .005)	.470 (± .003)
Flickr vs DD-V	.247 (± .007)	*-.007* (± .013)	.239 (± .014)	.234 (± .014)	*.011* (± .014)	**.251** (± .013)
Flickr vs DD-F	.231 (± .006)	*.003* (± .012)	.260 (± .013)	.255 (± .013)	*.011* (± .014)	**.273** (± .012)
Twitter-Kwak	**.671** (± .007)	-.082 (± .004)	.559 (± .008)	.659 (± .008)	-.073 (± .004)	.560 (± .008)

Bold values refer to the highest correlation per dataset. All values except those in italic are statistically significant (under the null hypothesis of independence of two rankings).

The in-neighbors-driven and in-out-neighbors-driven LurkerRank methods generally obtained the highest correlation with DD (e.g., 0.67 by LRin on *Twitter-Kwak*, 0.66 by ac-LRin on *FriendFeed*). Results confirmed that LRin and LRin-out (and their ac- counterparts) significantly improved upon all competing methods, with maximum gains of 0.59 against IO, 0.66 against PR, 0.45 against AC and 0.43 against FB.

Note that LRout and ac-LRout obtained the lowest scores on all datasets: interestingly, this behavior confirms our intuition that determining the strength of lurking of a given node should not depend solely on the strength of the lurking behavior shown by the out-neighbors of that node (i.e., Principle III of Definition 5.1).

Concerning correlation of each of the competing methods with DD, we observed on *FriendFeed* and *Twitter-Kwak* some correlation for FB (up to 0.37) and AC (up to 0.23), while IO and PR showed poor correlation. However, on *Flickr*, all competing methods tended to be uncorrelated with the two DD, with an average correlation of 0.05 over all competitors. More interestingly, it is worth noting that IO generally showed poor correlation with DD, which not only would justify the use of in/out-degree ranking as a baseline competing method, but also gives evidence that in/out-degree cannot be considered as a basic approximation of LurkerRank.

Comparative evaluation with LurkerRank methods

Tables 5.5–5.9 compare our LurkerRank methods against PageRank, alpha-centrality, Fair-Bets (all at convergence) as well as against DD (where possible) and IO. Note that results are organized on 3-row groups, where each row in a group corresponds to a specific variation of the Fagin's or Bpref's parameters.

On *Twitter-Kwak* (Table 5.5), LRin and LRin-out along with their ac- counterparts showed a relatively much higher F intersection with DD (0.516 on average) and IO (0.473) than with FB (0.08), and a nearly empty F with

Table 5.5. Comparative performances on *Twitter-Kwak*.

	F $k = 10^2$ // 10^3 // 10^4					*Bpref* $l = 10$ // 25 // 50				
	DD	IO	PR	AC	FB	DD	IO	PR	AC	FB
LRin	.527	.404	0.0	0.0	.112	**.997**	.992	.121	.790	.441
	.289	.209	0.0	0.0	.127	.995	.989	.473	.914	.704
	.581	**.617**	.001	.001	.068	.985	.962	.521	.866	.606
LRout	.030	.032	.181	.010	.034	.045	0.0	.754	.311	.313
	.008	.008	.351	.024	.015	.055	.001	**.757**	.650	.600
	.003	.002	**.437**	.048	.005	.109	.074	.641	.678	.648
LRin-out	.475	.364	0.0	0.0	.064	.968	**.981**	.039	.826	.204
	.314	.277	0.0	0.0	.063	.979	.977	.387	.929	.524
	.666	**.688**	.001	.001	.032	.961	.925	.453	.878	.489
ac-LRin	.583	.459	0.0	0.0	.174	**.993**	.990	.072	.808	.339
	.573	.570	0.0	0.0	.122	.992	.988	.443	.921	.653
	.767	**.810**	.001	.001	.048	.982	.967	.501	.872	.575
ac-LRout	.038	.032	.244	.006	.036	.049	0.0	**.796**	.339	.307
	.009	.008	.319	.017	.011	.059	0.0	.775	.659	.598
	.003	.002	**.362**	.042	.004	.120	.081	.654	.687	.643
ac-LRin-out	.473	.363	0.0	0.0	.062	.957	**.981**	.039	.828	.203
	.278	.234	0.0	0.0	.062	.975	.976	.386	.930	.464
	.663	**.685**	.001	.001	.031	.957	.933	.453	.880	.454

Bold values refer to the highest scores per LurkerRank method and assessment criterion.
Underlined bold values refer to the highest scores per assessment criterion.

respect to PR and AC. By contrast, LRout and ac-LRout exhibited a larger F with PR, although below 0.316 on average, while scoring even lower with respect to the other methods. Bpref evaluation led to mostly similar remarks on the relative comparison between proposed and other methods: LRin, LRin-out and their ac- counterparts highly matched DD and IO (around 0.97 on average), but also a moderately high *Bpref* with respect to AC (0.87) and mid-low *Bpref* with respect to FB (0.47). Again, as already observed for both the Kendall evaluation and the Fagin evaluation, LRout and ac-LRout showed no significant matches in practice with DD (while scoring pretty high with respect to PR).

Results on *Twitter-UDI* (Table 5.6) corroborated the advantage of LRin and ac-LRin with respect to the other LR methods. LRin-out and ac-LRin-out achieved lower F than LRin and ac-LRin, respectively, with respect to IO and FB, especially for higher k. Compared to the *Twitter-Kwak* case, *Bpref* values were relatively higher (respectively, lower) with respect to PR (respectively, AC), except for LRout and ac-LRout which had higher *Bpref* with respect to AC than in *Twitter-Kwak*.

On *Flickr* (Table 5.7), once again the best performance against the data-driven ranking (DD-F and DD-V) was obtained by LRin and LRin-out along with their ac- counterparts, and also roughly similar F values were obtained with respect to IO and FB. Note that both data-driven ranking (the *favorites*-based one, DD-F, and the *views*-based one, DD-V) corresponded to nearly identical results, with a slightly better agreement of the LR algorithms with respect to DD-F. In terms of *Bpref*, LRin, LRin-out and their ac- counterparts

Table 5.6. Comparative performances on *Twitter-UDI*.

	F $k = 10^2$ // 10^3 // 10^4				*Bpref* $l = 10$ // 25 // 50			
	IO	PR	AC	FB	IO	PR	AC	FB
LRin	.337	0.0	0.0	.184	.809	.254	.230	.477
	.245	0.0	0.0	.136	.917	.645	.633	.546
	.455	0.0	0.0	.292	**.927**	.709	.715	.568
LRout	0.0	0.0	0.0	0.0	0.0	.867	.767	.164
	0.0	.004	0.0	.002	0.0	**.876**	.766	.339
	.001	**.021**	.006	.001	.275	.810	.732	.531
LRin-out	**.305**	0.0	0.0	.160	.762	.130	.123	.299
	.178	0.0	0.0	.078	.897	.546	.550	.405
	.172	0.0	0.0	.076	**.902**	.646	.656	.443
ac-LRin	.343	0.0	0.0	.186	.825	.216	.202	.454
	.267	0.0	0.0	.152	.924	.617	.617	.524
	.446	0.0	0.0	.324	**.932**	.690	.704	.550
ac-LRout	0.0	0.0	0.0	0.0	0.0	.861	.765	.159
	0.0	.004	0.0	.002	0.0	**.873**	.765	.338
	.001	**.021**	.006	.001	.272	.807	.730	.530
ac-LRin-out	**.306**	0.0	0.0	.161	.877	.113	.153	.140
	.176	0.0	0.0	.076	.947	.482	.607	.293
	.153	0.0	0.0	.060	**.949**	.598	.692	.399

Bold values refer to the highest scores per LurkerRank method and assessment criterion.
Underlined bold values refer to the highest scores per assessment criterion.

Table 5.7. Comparative performances on *Flickr*.

	F $k = 10^2$ // 10^3 // 10^4						*Bpref* $l = 10$ // 25 // 50					
	DD-F	DD-V	IO	PR	AC	FB	DD-F	DD-V	IO	PR	AC	FB
LRin	.576	.574	**.639**	0.0	0.0	.552	.361	.327	.921	.465	.769	.502
	.451	.433	.511	.003	.007	.463	.532	.496	.953	.522	.783	.488
	.297	.286	.383	.018	.008	.313	.650	.630	.931	.499	**.987**	.570
LRout	.102	.101	.123	.045	0.0	.037	.071	.060	.206	.620	.862	.138
	.124	.121	.107	.064	0.0	.008	.252	.218	.509	.503	.868	.229
	.015	.014	.126	**.237**	.007	.033	.460	.446	.645	.411	**.878**	.392
LRin-out	.561	.559	**.626**	0.0	0.0	.536	.353	.321	.878	.441	.761	.520
	.462	.444	.520	.004	.007	.462	.305	.292	**.883**	.474	.766	.509
	.311	.301	.398	.021	.008	.310	.430	.417	.667	.478	.748	.594
ac-LRin	.609	.607	**.676**	0.0	0.0	.587	.349	.316	.878	.458	.784	.498
	.535	.513	.604	.004	.007	.538	.523	.487	**.940**	.484	.792	.482
	.348	.336	.447	.018	.009	.352	.644	.625	.921	.481	.795	.573
ac-LRout	.102	.009	.123	.051	0.0	.037	.071	.060	.209	.622	.660	.138
	.105	.101	.107	.072	0.0	.008	.256	.220	.514	.510	.670	.232
	.115	.114	.127	**.229**	.007	.034	.477	.464	.645	.413	**.675**	.392
ac-LRin-out	.443	.440	**.510**	0.0	0.0	.432	.375	.345	.958	.604	.640	.520
	.305	.293	.337	.002	.004	.291	.569	.533	**.970**	.675	.677	.466
	.232	.224	.293	.013	.006	.215	.676	.655	.954	.569	.706	.494

Bold values refer to the highest scores per LurkerRank method and assessment criterion.
Underlined bold values refer to the highest scores per assessment criterion.

highly matched IO. *Bpref* values were also moderately high with respect to AC and mid-low with respect to PR and FB.

Looking at *FriendFeed* results (Table 5.8), LRin and LRin-out along with their ac- counterparts were again the best-performing methods against DD (0.42 *F* and 0.97 *Bpref*), and also showed mid *F* (0.34) and high *Bpref* (0.89) with respect to FB. Yet, LRout and ac-LRout were moderately in agreement with PR and AC in terms of *F*, whereas all LR generally achieved mid *Bpref* with both PR and AC.

Table 5.8. Comparative performances on *FriendFeed*.

	F $k = 10^2$ // 10^3 // 10^4					*Bpref* $l = 10$ // 25 // 50				
	DD	IO	PR	AC	FB	DD	IO	PR	AC	FB
LRin	.542	**.690**	.024	.010	.453	**1.0**	.980	.331	.606	.985
	.488	.586	.108	.118	.384	.998	.976	.570	.802	.977
	.576	.628	.126	.153	.493	.986	.953	.678	.843	.898
LRout	.015	.009	.479	.620	.011	.008	0.0	.691	.672	.031
	.138	.163	.550	**.725**	.167	.030	.038	**.764**	.746	.066
	.154	.156	.498	.704	.184	.062	.110	.739	.737	.258
LRin-out	.207	.297	.032	.042	.170	**.972**	.910	.252	.604	.879
	.278	.320	.061	.064	.166	.955	.910	.553	.794	.870
	.424	**.455**	.076	.099	.338	.914	.874	.642	.815	.813
ac-LRin	.575	**.735**	.025	.014	.467	**1.0**	.980	.300	.605	.980
	.520	.627	.118	.131	.403	.999	.977	.548	.803	.969
	.603	.660	.130	.161	.503	.988	.954	.661	.845	.882
ac-LRout	.015	.009	.479	.620	.011	.008	0.0	.691	.672	.031
	.138	.163	.550	**.725**	.167	.030	0.0	**.749**	.726	.066
	.154	.156	.498	.704	.184	.040	.080	.723	.718	.257
ac-LRin-out	.169	.243	0.0	0.0	.126	**.958**	.891	.237	.594	.852
	.240	.273	.001	.001	.122	.942	.892	.546	.785	.836
	.400	**.426**	.041	.064	.310	.898	.853	.634	.803	.782

Bold values refer to the highest scores per LurkerRank method and assessment criterion.
Underlined bold values refer to the highest scores per assessment criterion.

Table 5.9. Comparative performances on *GooglePlus*.

	F $k = 10^2$ // 10^3 // 10^4				*Bpref* $l = 10$ // 25 // 50			
	IO	PR	AC	FB	IO	PR	AC	FB
LRin	.742	0.0	0.0	.363	**1.0**	.434	.582	.976
	.850	.001	0.0	.480	.993	.584	.695	.962
	.881	.063	.144	.592	.987	.684	.722	.937
LRout	.011	.079	0.0	.015	**.972**	.796	.796	.686
	.015	.107	.012	.015	.971	.793	.790	.815
	.223	**.322**	.144	.213	.964	.782	.774	.807
LRin-out	.629	0.0	0.0	.318	**1.0**	.462	.587	.907
	.721	0.0	0.0	.419	.991	.572	.688	.910
	.799	.045	.130	.547	.989	.677	.731	.886
ac-LRin	.747	0.0	0.0	.361	**1.0**	.456	.578	.976
	.851	.001	0.0	.477	.992	.546	.702	.963
	.882	.063	.143	.591	.988	.699	.724	.937
ac-LRout	.011	.077	0.0	.015	**.972**	.796	.796	.687
	.015	.107	.012	.015	.971	.793	.790	.815
	.223	**.322**	.145	.212	.965	.782	.774	.807
ac-LRin-out	.647	0.0	0.0	.328	**1.0**	.489	.586	.896
	.729	0.0	0.0	.422	.994	.612	.675	.899
	.795	.042	.125	.543	.983	.702	.727	.875

Bold values refer to the highest scores per LurkerRank method and assessment criterion.
Underlined bold values refer to the highest scores per assessment criterion.

GooglePlus evaluation results (Table 5.9) led us to draw conclusions similar to the other network datasets in terms of F values: in- and in-out-based algorithms outperformed the out-based ones when comparing with IO and FB, while nearly empty intersection was found with respect to PR and AC. LRin, LRin-out and their ac- counterparts achieved very high *Bpref* with respect to IO, and also showed good agreement with FB.

Table 5.10. *Twitter-Kwak* t-test on the per-iteration performances.

	Fagin evaluation			*Bpref evaluation*		
	PR	AC	FB	PR	AC	FB
LRin	4.4E-65	4.4E-65	8.4E-11	5.2E-110	1.1E-25	2.1E-65
LRout	2.8E-41	2.7E-41	1.8E-04	3.2E-50	5.5E-79	9.2E-71
LRin-out	4.3E-277	4.4E-277	2.9E-12	1.5E-89	6.7E-21	7.6E-65
ac-LRin	5.6E-228	5.6E-228	4.8E-14	1.2E-91	2.1E-25	2.7E-65
ac-LRout	6.5E-34	6.2E-34	1.8E-04	4.1E-54	1.8E-71	2.3E-73
ac-LRin-out	3.8E-213	3.3E-265	3.4E-12	5.8E-85	2.1E-21	1.0E-64

Table 5.11. *FriendFeed* t-test on the per-iteration performances.

	Fagin evaluation			*Bpref evaluation*		
	PR	AC	FB	PR	AC	FB
LRin	1.3E-116	1.3E-103	2.6E-10	4.5E-195	5.9E-197	6.1E-10
LRout	8.5E-12	1.6E-101	1.5E-38	6.8E-252	1.3E-264	2.5E-271
LRin-out	6.0E-193	2.4E-166	2.1E-24	1.3E-298	2.1E-212	2.2E-116
ac-LRin	1.0E-195	1.0E-172	4.4E-13	5.0E-298	3.9E-189	7.8E-10
ac-LRout	2.6E-12	5.1E-88	1.3E-38	4.1E-99	5.9E-299	1.4E-282
ac-LRin-out	8.1E-63	1.3E-96	2.1E-25	8.3E-82	5.1E-226	1.5E-75

Statistical significance testing

We also determined the statistical significance of the better performance of LurkerRank methods with respect to the competing ones, through two stages of statistical testing analysis; in both cases, we fixed the Fagin parameter as $k = 10^4$ (which ensured a larger overlap between the ranking lists to be compared) and the Bpref parameter as $l = 25$ (for which $|R|$ was always smaller than $|N|$). Results refer here to *Twitter-Kwak* and *FriendFeed*, nevertheless similar conclusions were actually reached for the other evaluation networks.

Tables 5.10–5.11 show the p-values resulting from an unpaired two-tail t-test, in which the performance scores obtained for each iteration by a ranking method with respect to DD were regarded as the statistical samples, under the null hypothesis of no difference in performance with respect to DD between a LurkerRank method and a competing method. Note that in all cases, the number of iterations (samples) was adequate to perform a t-test (generally above 50). Looking at the two tables and both F and $Bpref$ evaluation, the p-values turned out to be extremely low in most cases, thus giving a strong evidence that the null hypothesis was always rejected, at 1% significance level. This finding was useful to confirm that a certain difference (actually, the improvement) in performance between the LR methods and the competing ones, also on *FriendFeed* for which relatively high $Bpref$ scores were observed in the previous analysis.

In the second stage of statistical testing, we analogously performed a paired two-tail t-test in which the samples corresponded to the F scores respectively obtained by two ranking methods with respect to DD over the same randomly generated subgraph. For each of the network datasets, we extracted 100 subgraphs, each time starting from a randomly picked seed node and roughly covering a fixed number of nodes (around 1/100 of the original network size). This test was hence intended to stress the ranking methods performing over

Table 5.12. Comparative performances on *FriendFeed* damping factor depending on the average path length.

	F $k = 10^2$ // 10^3 // 10^4					$Bpref$ $l = 10$ // 25 // 50				
	DD	IO	PR	AC	FB	DD	IO	PR	AC	FB
LRin	.450	**.686**	.012	.003	.445	.955	.978	.318	.601	**.985**
	.422	.582	.079	.078	.341	.914	.975	.567	.800	.974
	.529	.627	.111	.134	.472	.678	.952	.673	.839	.897
LRout	.015	.072	.510	.620	.015	.011	0.0	.689	.672	.027
	.138	.070	.571	**.725**	.184	.033	.041	**.762**	.747	.059
	.154	.208	.508	.704	.189	.155	.121	.738	.737	.199
LRin-out	.205	.294	.020	.031	.191	.759	.909	.250	.604	.871
	.274	.317	.053	.055	.182	.744	**.910**	.553	.792	.860
	.421	**.448**	.074	.096	.352	.602	.872	.642	.813	.804
ac-LRin	.485	**.727**	.016	.004	.479	.961	.978	.291	.600	**.981**
	.450	.623	.088	.090	.367	.916	.975	.545	.800	.967
	.553	.656	.115	.141	.488	.679	.951	.656	.841	.883
ac-LRout	.015	.072	.510	.620	.015	.011	0.0	.689	.672	.027
	.138	.070	.571	**.725**	.184	.033	.008	**.747**	.726	.059
	.154	.208	.508	.704	.189	.142	.102	.721	.718	.199
ac-LRin-out	.169	.239	0.0	0.0	.140	.745	.889	.237	.594	.850
	.240	.271	.001	.001	.136	.722	**.891**	.547	.785	.833
	.400	**.421**	.042	.064	.325	.592	.854	.636	.803	.780

Bold values refer to the highest scores per LurkerRank method and assessment criterion.
Underlined bold values refer to the highest scores per assessment criterion.

a pool of subnetworks having different characteristics from each other, and from the whole original network as well; for instance, on *Twitter-Kwak*, the subnetworks had average path length mean of 2.52 (0.86 stdev), and in/out-degree ratio mean of 0.07 (0.13 stdev) — this might be explained because of the adopted approach of breadth-first traversal of the network, which led to connect the majority of nodes with a few source nodes having very high out-degree. On *Twitter-Kwak*, we observed a close behavior between the LurkerRank methods (except LRout and ac-LRout) and AC (around 0.19 F on average), and between PR and FB, which however achieved a lower average F (0.029) — note that k was still set to 10^4, hence very high for such network sizes (i.e., around 200,000 nodes). In any case, i.e., for each pair of LurkerRank method vs. competing method, the null hypothesis of equal means was rejected even at 1% significance level, since the p-values were ranging from 1.4E-3 to 2.8E-19. Analogous final remarks were drawn for *FriendFeed*.

Relation between damping factor and average path length

In our proposed methods, the damping factor α is chosen to be 0.85, in analogy with the default setting of the parameter in the original PageRank algorithm. Recall this finds an explanation 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 $\alpha = 1-(1/6) \approx 0.85$. On the other hand, research on degrees-of-separation in directed network graphs has shown that for many OSNs the average path length is typically below 6 (e.g., [13, 125]). Here we leverage on

Table 5.13. Comparative performances on *GooglePlus* with damping factor depending on the average path length.

	F $k = 10^2$ // 10^3 // 10^4				*Bpref* $l = 10$ // 25 // 50			
	IO	PR	AC	FB	IO	PR	AC	FB
LRin	.729	0.0	0.0	.551	**1.0**	.438	.584	.985
	.829	.001	0.0	.631	.989	.585	.700	.963
	.864	.061	.140	.690	.983	.689	.725	.927
LRout	.011	.085	0.0	.022	.972	.994	**.996**	.671
	.015	.148	.012	.018	.971	.993	.990	.795
	.223	**.356**	.144	.232	.964	.981	.974	.783
LRin-out	.629	0.0	0.0	.474	**.997**	.467	.590	.940
	.720	0.0	0.0	.546	.989	.576	.689	.915
	.798	.047	.129	.642	.980	.679	.734	.876
ac-LRin	.732	0.0	0.0	.551	**1.0**	.459	.579	.986
	.830	.001	0.0	.629	.990	.550	.702	.963
	.864	.061	.139	.689	.986	.711	.726	.927
ac-LRout	.011	.083	0.0	.022	.972	.994	**.996**	.671
	.015	.148	.012	.018	.971	.993	.990	.796
	.223	**.356**	.145	.232	.965	.981	.974	.783
ac-LRin-out	.647	0.0	0.0	.488	**.998**	.492	.590	.935
	.729	0.0	0.0	.550	.991	.623	.678	.907
	.795	.044	.125	.638	.984	.709	.728	.866

Bold values refer to the highest scores per LurkerRank method and assessment criterion.
Underlined bold values refer to the highest scores per assessment criterion.

this result, confirmed in our network datasets as well, to understand how the ranking performance may change as the damping factor is varied in function of a network-specific structural characteristic like the average path length. Precisely, we set α as $\alpha = 1 - (1/apl)$, being apl the average path length of the particular network. For this evaluation stage, we focused on *FriendFeed* and *GooglePlus*, which exhibit the lowest average path lengths, i.e., 3.82 and 3.32, respectively (cf. Table 5.2).

Comparing the results in Table 5.13 that correspond to $\alpha = 0.7$ with the results obtained with default α (Table 5.9) on *GooglePlus*, F values were slightly lower (resp. unvaried) for the in- and in-out-based algorithms, (resp. for the out-based algorithms) with respect to IO, generally higher with respect to FB, and equal or higher with respect to PR and AC. Again comparing with the results in Table 5.9, *Bpref* slightly increased with respect to PR and AC and decreased with respect to IO. As for *FriendFeed*, comparing Table 5.8 with Table 5.12, we found that F values were generally lower when using $\alpha = 0.74$ for in- and in-out-based algorithms, and higher for out-based ones. A decrease in the performance of in- and in-out-based algorithms was observed for *Bpref* as well, especially with respect to DD.

Overall, it appears that the average path length cannot be regarded as a good estimator of damping factor in our methods, in the sense of a necessarily better alternative to the default 0.85. However, we would tend to take this sort of conclusion with a grain of salt, due to the heterogeneity of such networks and the lack of more example networks with average path length significantly below 6.

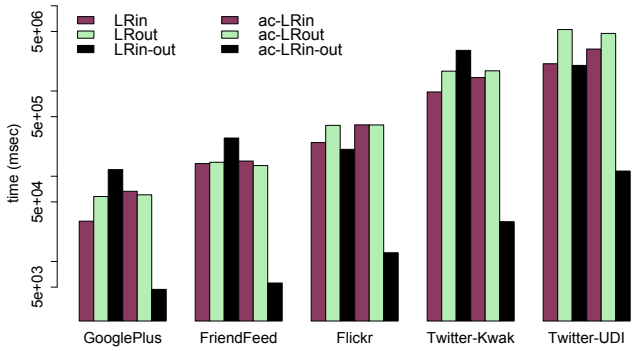


Fig. 5.7. Runtime performance of LurkerRank methods.

Efficiency results

Figure 5.7 shows the runtime performance of LurkerRank algorithms. The times do not include the graph building step.[1] Firstly, it was interesting to observe on all datasets that the LurkerRank methods consistently reached a ranking stability very quickly, in the range $35 \div 75$ iterations, with the exception of ac-LRin-out which always reached convergence with fewer iterations. The latter fact is however explained by a generally poor diversification of the ranking scores achieved by ac-LRin-out, which particularly affects the top of the ranking results: in fact, in most datasets, the scores at the maximum as well as the third quartile are of the same order of magnitude as the mean or even as the first quartile scores. LRin and LRout mostly required pretty similar running times, while LRin-out was slower than the other algorithms on 3 out of 5 networks — about twice the running time of LRin and LRout, which is clearly explained since LRin-out needs to iterate both on the in- and out-neighborhood of each node. As concerns the alpha-centrality based formulations, ac-LRin always required a higher number of iterations to reach ranking stability than LRin, while ac-LRout performed similarly and sometimes faster than LRout, considering that in most cases both algorithms needed the same number of iterations until ranking stability. As a side remark, it should be noted that our power-iteration-method implementation of the LR algorithms caused quite different performance for networks with a number of edges of the same order of magnitude, but a greater difference in the number of nodes (e.g., *FriendFeed* and *Flickr*).

5.6.5 Delurking-oriented randomization

As we discussed in the Introduction, the ultimate objective of lurker analysis is in principle to attract the lurkers to the community life, that is, to change

[1] Experiments were carried out on an Intel Core i7-3960X CPU @ 3.30GHz, 64GB RAM machine.

Algorithm 5 Delurking-oriented randomization

Input: The topology graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ of an OSN. The ranking L corresponding to a LR solution for $\mathcal{G}$. Cut-off percentage thresholds t_1, t_2 of ranking order in L. Probability p. Maximum fraction d of new edges to add to $\mathcal{G}$.

Output: A randomized graph $\mathcal{G}'$.

$\mathcal{E}' \leftarrow \emptyset$
Sort L by decreasing lurking score
Let L_{top} (resp. L_{bottom}) be the top-t_1 (resp. bottom-t_2) of the sorted L
$E_{al} \leftarrow \{e = (a, l) \in \mathcal{E} \mid a \in L_{\text{bottom}}, l \in L_{\text{top}}\}$
repeat
    Pick randomly with probability p an edge $(a_1, l_1) \in E_{al} \setminus \mathcal{E}'$
    Pick randomly with probability p an edge $(a_2, l_2) \in E_{al} \setminus \mathcal{E}'$, with $a_2 \neq a_1, l_2 \neq l_1$
    $\mathcal{E}' \leftarrow \mathcal{E}' \cup \{(l_1, a_2), (l_2, a_1)\}$ /* *add the new edges* */
until $(|\mathcal{E}'| \geq d|E_{al}|)$
$\mathcal{G}' \leftarrow \langle \mathcal{V}, \mathcal{E}' \cup \mathcal{E} \rangle$

their status to that of active players in the network. Although devising real delurking plans (which might rely on marketing aspects) goes beyond our study, we are still interested in conceiving a general topology-based model that can support "self-delurking" of a network.

For this purpose, we introduce a novel randomization-like model, named *delurking-oriented randomization*. Randomized models are commonly used to monitor how varying a certain topological feature may impact on the dynamics of the network. The most widely applied randomized model uses the concept of rewiring, so that the edges of the original (undirected) network are randomly rewired pairwise. The key idea behind our delurking-oriented randomization model is to simulate a mechanism of disclosure of the presence of lurkers, by letting more-likely-active users virtually hear from less-likely-active users.

Algorithm 5 shows our delurking-oriented randomization method, which substantially works by inserting new connections into the network each of which randomly links a vertex selected from the top of a predetermined LR ranking solution to a vertex selected from the bottom of that ranking. The algorithm hence requires cut-off thresholds to control the selection of the head and tail of the LR distribution, and a percentage threshold to control the degree of delurking-oriented randomization (i.e., the fraction of potentially new edges to add to the graph). At each step of insertion of a new pair of edges, it is to be ensured that both the new formed edges do not already exist in the graph — this restriction prevents the appearance of multiple edges connecting the same pair of vertices. It should be noted that Algorithm 5 does not provide a proper randomization model in its usual definition, since both the size of the network and the degree of vertices will change.

We applied Algorithm 5 to our networks, with the following setting: $p = 0.5$, $t_1 = t_2 = 25\%$, and d ranging from 0.2 to 1.0 (with increment by 0.2). Note that this setup of the algorithm was chosen to allow us to focus mainly

on the degree of delurking-oriented randomization (d); as for the partition of the lurker ranking list, we decided to leave the middle 50% out and hence select one quartile both for the top (t_1) and the bottom (t_2) of the ranking list.

For this stage of evaluation, we mainly focused on two features of the network: the LR distribution and the in/out-degree distribution (either with and without the inclusion of sink and source vertices), and analyzed the pairwise correlations between a LR (resp. in/out-degree) ranking on a particular network and the LR (resp. in/out-degree) rankings obtained on the corresponding delurking-randomized networks.

Considering the case where all vertices were included in the evaluation, we observed no clear trend both in the pairwise correlations between the LR ranking solutions at the different degrees of delurking-oriented randomization, which were either moderate (*Flickr*) or high, and in the correlations between an original LR and each of the LR solutions in the randomized networks, which were either absent (*Flickr* and *FriendFeed*) or moderate/high. However, when sink and source vertices were discarded from the analysis, trends become more evident: in one case (corresponding to the *Twitter* networks), the pairwise correlations between the LR ranking solutions at the different degrees of delurking-oriented randomization were moderate, while absent or moderate with respect to the original LR ranking; however, in the other case (corresponding to *GooglePlus*, *Flickr*, and *FriendFeed*), the LR ranking solutions at the different degrees of delurking-oriented randomization turned out to be not or scarcely correlated to each other as well as totally uncorrelated to the original LR ranking.

Interestingly, the above remarks indicate that upon a delurking-oriented randomization process, the top-ranked lurkers can significantly change, not only with respect to the original configuration of the network but also with respect to a configuration corresponding to a different degree of delurking-oriented randomization (shown in Fig. 5.8 for the LRin evaluation). Clearly, as expected, when considered as a global feature of the network, the delurking-oriented randomization impact can be lower for larger networks (e.g., *Twitter*), which have much lower (resp. higher) clustering coefficient (resp. average path length) than the other network datasets.

By contrast, the delurking-oriented randomization seems to negligibly affect the in/out-degree distribution: correlations turned out to be moderate to high (when sinks and sources were considered) both between the in/out ranking in the original network and each of the in/out rankings of the randomized networks, and between the randomized in/out rankings pairwise. This result would indicate that an apparently "invasive" alteration of the topology (through the insertion of new links) actually will not significantly change the topological features based on in- and out-degree distributions.

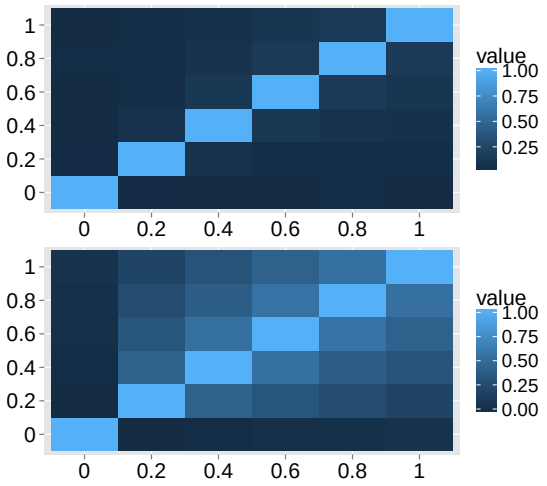


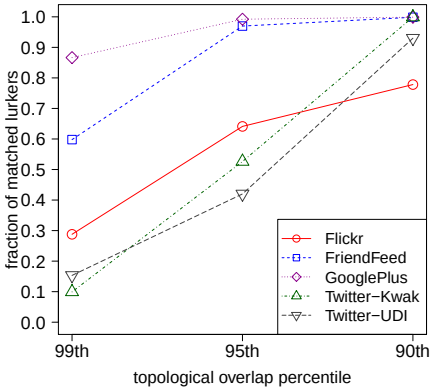
Fig. 5.8. Delurking-oriented randomization analysis: pairwise correlation between LRin solutions, as obtained on original network and randomized networks, for increasing degree of delurking-oriented randomization, on *Flickr* (top) and *FriendFeed* (bottom).

5.6.6 Percolation analysis

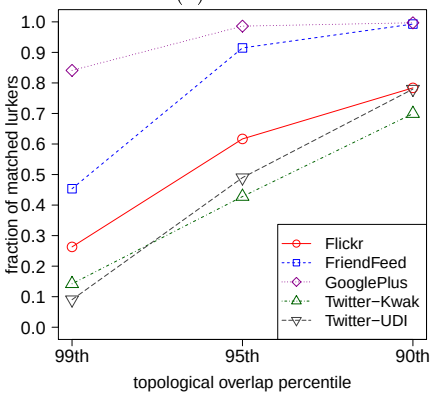
Percolation analysis corresponds to studying the effect of network disruption via edge removal strategies, generally with the purpose of assessing topological integrity properties of the network or its vulnerability to (random) failures/attacks. An edge removal strategy is typically based on local structural properties of edges, such as topological overlap. Topological overlap is a measure originally introduced in [132] for undirected networks, which evaluates the number of neighbors shared by two given vertices i and j. Edges between connected components are expected to have a low number of common neighbors, and hence low topological overlap.

Removing edges by increasing order of topological overlap has shown to effectively detect the edges that act as *bridges* between different communities [61, 137]. Upon this we build our intuition that if we would discover a certain correlation between the result of our lurker detection and the result of percolation based on topological overlap, then we could claim that *lurkers are likely to behave as bridges between communities.*

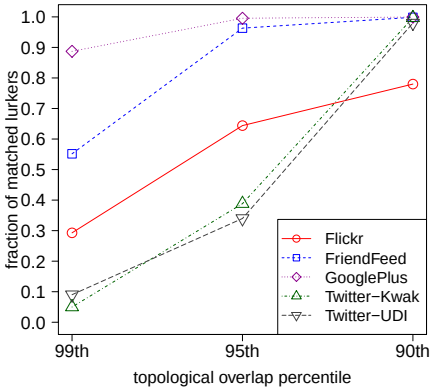
Our network model however implies that edges are directed from information-producer to information-consumer, therefore the notion of bridge as highly active user must be revised as less active user. Therefore, we needed first to adapt the basic topological overlap to our setting of directed networks, whereby the neighbor sets of any two selected vertices are partly considered according to the orientation of the edge drawn between the two vertices. Given edge (i, j), we define the *directed topological overlap* as:



(a) LRin



(b) LRout



(c) LRin-out

Fig. 5.9. Percolation analysis: fraction of lurkers matched as function of the vertices removed based on directed topological overlap.

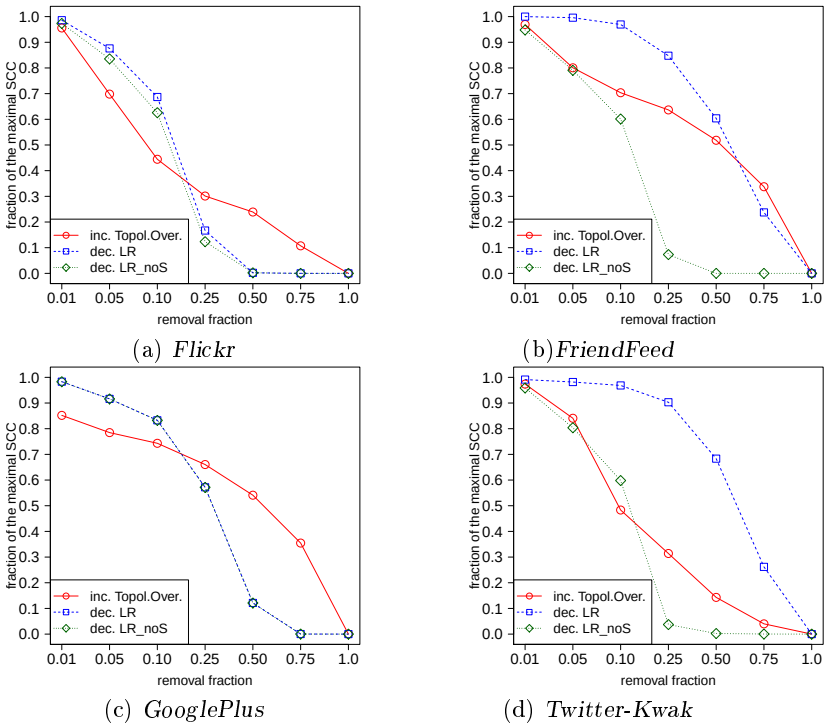
$$O(i,j) = \frac{|R_i \cap B_j|}{(|R_i| - 1) + (|B_j| - 1) - |R_i \cap B_j|} \tag{5.21}$$

We developed a stage of evaluation in which two sets of vertices are compared with each other: the one resulting from an edge removal strategy based on increasing order of our directed variant of topological overlap, and the other one corresponding to the highest-ranked lurkers detected by one of our LR algorithms.

Figure 5.9 plots the fraction of top-25% of lurkers that matched the sets of vertices respectively included in the 99th, 95th and 90th percentile of the edges with lowest directed topological overlap. LRin, LRout, and LRin-out were used to rank lurkers. The methods appear to behave very closely to each other for all data, with some relative differences on the two *Twitter* networks. At 90th percentile of the edges with lowest directed topological overlap, almost all top-lurkers were matched on *FriendFeed*, *GooglePlus*, and only by LRin and LRinout, on the two *Twitter* networks as well. Moreover, on *FriendFeed* and *GooglePlus*, most top-lurkers were matched already at 95th percentile.

Clearly, this relatively easy tendency of covering the set of top-lurkers needs to be interpreted in relation to the ratio of the number of vertices removed (by increasing directed topological overlap) with respect to the total number of vertices in the network. While on *FriendFeed* and *GooglePlus* the number of vertices removed corresponded to more than 90% of the total vertex set (which hence explains the high rate of coverage over the top-lurkers), on both the two *Twitter* networks, the above percentage was instead less than 27%. The latter, being observed on the two largest evaluation networks, should be taken as an important finding, which would confirm the relationship between the lurkers and the bridges between communities.

We also analyzed the resilience of the various networks when vertices are removed by decreasing lurking order. To better evaluate the impact of sinks on the network disruption, we distinguished two cases: either sinks were preliminarily filtered out or they were included when selecting the fraction of lurkers to remove from the network. As shown in Figure 5.10 for LRin, the removal strategy with the most disruptive effects was that based on decreasing LR rank (with pre-filtering of sink vertices, denoted as LR_noS in the figure), which led to mostly dismantle the maximal strongly connected component (i.e., 80 to 90% of its size) already for 25% of vertex removal in all networks except *GooglePlus* (for which 50% of vertex removal was needed). By contrast, the removal strategy based on increasing topological overlap produced disruptive effects smoother with respect to the two LR-based strategies, on all networks. Interestingly, by including sink vertices in the selection of lurkers to remove, the network resilience was the same as in the case of sink-pre-filtering on *GooglePlus* and *Flickr*, whereas on the two *Twitter* and *FriendFeed* the resilience was higher than for the other strategies, since a level of dismantling below 70-60% was reached only by a removal fraction of 50% or higher. Note



(a) *Flickr* (b) *FriendFeed*

(c) *GooglePlus* (d) *Twitter-Kwak*

Fig. 5.10. Percolation analysis: fraction of the maximal strongly CC as function of removed vertices.

that *Twitter* and *FriendFeed* are the networks with a strong presence of sink vertices, and with a sink/source ratio greater than 10.

5.6.7 Qualitative evaluation

We investigated the meaningfulness of the rankings produced by LurkerRank methods as well as produced by the competing methods. For this analysis, we retrieved the OSN pages of top-ranked users and examined the available information about their profile and neighborhoods. Our goal was to understand whether a user actually looks like a lurker, or conversely s/he takes another role in the network.

Tables 5.14–5.15 show the top-20 ranked users obtained on *Twitter-Kwak* and *FriendFeed* by PageRank, alpha-centrality, Fair-Bets, and LRin. Table 5.14 also reports the number of times a user was retweeted (*#rt*), whereas Table 5.15 reports the total number of posts by a user (*#posts*). Moreover, we left sink nodes out of consideration in order to avoid biasing our evaluation with trivial lurkers.

By comparing the top-ranked lists, it is evident that LRin behaved differently from the other algorithms, since it shared just two users with FB

(dark-grey shaded) and no users at all with PR and AC. Interestingly, the LRin top-ranked list contains only users who have never been retweeted; by retrieving the tweet post dates from Twitter, those users were all found as quite longer-time users, as in fact they joined Twitter much earlier (e.g., #8, #10 and #12 joined in 2007) than most users in the AC and PR top-ranked lists. Conversely, in the latter two lists most users have been significantly retweeted although they joined later (e.g., 2009).

PR and AC showed a certain association, with ten users in common (light-grey shaded). Most users in both AC and PR lists however were retweeted hundreds times, and hence they should not be considered as lurkers. Our hypothesis of non-lurking for those users was fully confirmed as we observed that those users' retweets were actually spread over a relatively short period of time (e.g., second half of 2009). Moreover, AC and PR ranked the same user on top, who is also the one having the highest number of retweets in the lists; indeed, that user is a very influential person, and in fact s/he has a followee/follower ratio much below 1: this would indicate that both AC and PR were not able to correctly handle this case (i.e., scoring it low enough), because their performance would be more affected by highly influential incoming links (i.e., followees) — which is a clear indication of tendency to absorb valuable knowledge — rather than by the number and type of followers. We also found other cases with characteristics similar to #1, e.g., #12 in the PR list, #10 and #14 in the AC list, and the common users "ZAP." (#3 in both lists) and "SCO." (#17 in PR, #12 in AC).

As concerns FB, it was surprising to find that 15 out of 20 top-ranked users refer to spammers (#4, a fashion/cosmetic marketing spammer, #9, in advertising, and #15, a porn spammer), or in general to suspended accounts (#2-3, #5, #8, #10-11, #13-14, #17-20). Only #6, #12 and #16 appear to be lurkers, which might be confirmed by their high in/out-degree ratio coupled with a zero retweet-count. By contrast, #1 is an art director and designer, and #7 refers to an account actively used for academic advising purposes; probably, the high number of followees (e.g., about 1800 for #7) has misled the method. Therefore, like PR and AC, FB might also fail to correctly recognize real lurkers.

In *FriendFeed* (Table 5.15), a large intersection was found among the top-20 users not only between PR and AC (like in *Twitter-Kwak*) but also between FB and LRin. Looking at the users' profiles and at the contents of their posts, we can state that most of the users shared by the top20 lists of FB and LRin are recognized either as content spammers (i.e., users that have produced spamming contents, regardless of the popularity and number of their posts), or as professionals who aim to improve their visibility while staying as observers in the community (e.g., #9 in LRin/#7 in FB is a marketing expert, #7 in LRin/#5 in FB is a graphic designer). Some distinct profiles are also found to be clones, as they are associated to the same spamming contents (e.g., #1 and #4 in LRin, which correspond to #1 and #17 in FB, both probably related to a Russian commercial site). A reason for this massive presence of spammers

Table 5.14. Top-20 *Twitter-Kwak* users by lurking score.

rank	PR		AC		FB		LRin	
	user	*#rt*	*user*	*#rt*	*user*	*#rt*	*user*	*#rt*
1	B.O.	17811	B.O.	17811	D.W.S.	0	R.F.	0
2	W.F.	1676	ZAI.	10902	n.a.	0	R.J.	0
3	ZAP.	8707	ZAP.	8707	APA.	0	R.M.K.	0
4	TH.	7169	AS.	1172	T.S.C.	1	B.B.P.	0
5	L.E.	683	M.M.	7	n.a.	0	TR.	0
6	J.B.	1248	W.F.	1676	CON.	0	MU.	0
7	M.S.	476	M.K.	48	K.T.	0	B.R.	0
8	AS.	1172	P.B.	328	n.a.	0	AZ.	0
9	OH.	1009	W.A.	2814	S.M.	0	O.L.	0
10	H.T.	43	C.B.	11943	n.a.	0	N.T.	0
11	E.T.	2435	EL.	902	n.a.	0	FR.	0
12	SCH.	3277	SCO.	6970	M.P.	0	D.W.S.	0
13	RE.	1467	WI.	811	n.a.	0	AW.	0
14	H.S.	1346	O.W.	1803	n.a.	0	O.B.	0
15	M.M.	7	T.B.B.	102	M.E.	0	N.C.	0
16	ZAI.	10902	T.S.	74	B.B.P.	0	D.P.	0
17	SCO.	6970	S.S.	789	n.a.	0	AU.	0
18	M.K.	48	M.W.	363	n.a.	0	EM.	0
19	WI.	811	H.R.	750	n.a.	0	DI.	0
20	W.A.	2814	A.K.	1572	n.a.	0	M.A.	0

For privacy reasons, users' names were replaced with their initials or abbreviations.

probably can be found in the nature of the *FriendFeed* social network: being a real-time cross-network feed aggregator makes it a desirable and user-friendly means for spammers to reach high visibility, producing a number of user profiles for spamming attempts having very similar characteristics to lurker ones (e.g., high in/out degree ratio, low interaction with other members). Looking at PR and AC top-20 users we found that, as in the *Twitter-Kwak* case, most of them are not recognizable as lurkers, but rather as active and authoritative users (e.g.,#7 in PR/#3 in AC is a finance blogger, #1 in both PR and AC is an industrial designer, #5 in PR/#2 in AC represents a philanthropic foundation). We also found a user shared by PR and FB top-20s: although the account does not exist anymore, its name would hint that the user was probably a spammer for a hosting solutions company.

Concerning *GooglePlus* (results not shown), the top-ranked list by LRin is mainly comprised of users that show poor public activity, and that added a lot of people to their circles although scarcely reciprocated. FB showed a behavior nearly similar to LRin, however its top-20 list contains less real lurkers than those detected by LRin. PR and AC ranked high users that are likely to be pretty influential, such as a classical guitarist with more than 60 thousand followers (ranked #1 by PR), a landscape photographer with more than 42 thousand followers (ranked #1 by AC), and even a social media director with nearly 400 thousand followers (ranked #6 by PR). In contrast to the other network datasets, there were no shared users among PR and AC top-20s, while FB shared 2 users with PR and 7 with LRin.

Table 5.15. Top-20 *FriendFeed* users by lurking score.

rank	PR		AC		FB		LRin	
	user	*#posts*	*user*	*#posts*	*user*	*#posts*	*user*	*#posts*
1	N.D.P.	350	N.D.P.	350	M.C.D.	11	M.C.D.	11
2	FRE.	3	C.T.	5	BOG.	367	BOG.	367
3	BR.	71	J.D.A.	282	L.H.	1	B.I.	61
4	A.C.	142	MBL.	37	DIM.	1	N.D.	13
5	C.T.	5	BR.	71	B.I.	61	G.A.	11
6	MBL.	37	U.R.	52	G.A.	11	L.H.	1
7	J.D.A.	282	TAV.	65	A.C.	2	R.W.	7
8	U.R.	52	D.H.	89	W.H.O.	10	ZAH.	3
9	S.M.	106	P.B.	13	ASR.	0	A.C.	2
10	W.H.O.	10	C.E.	447	H.P.B.	3	E.J.S.	24
11	RID.	886	RID.	886	MUA.	5	M.P.	2
12	D.G.	35	W.B.	5	E.J.S.	24	Y.P.	1
13	L.A.C.	4	R.T.	68	SVL.	1	S.E.	72
14	JSI.	49	K.K.	134	R.W.	7	J.N.	110
15	K.K.	134	D.S.	105	S.F.T.	4	H.P.B.	3
16	S.O.	12	L.A.C.	4	D.G.	5	P.C.	3
17	W.M.	108	JSI.	49	N.D.	13	MRT.	3
18	STR.	2	B.C.	14	I.P.G.	10	I.K.G.	2
19	C.F.	3	D.V.	85	ARG.	2	N.L.	1
20	R.T.	68	M.M.H.	34	E.E.M.	5	F.F.	764

For privacy reasons, users' names were replaced with their initials or abbreviations.

5.6.8 Some lessons learned

Our study so far allows us to draw some interesting conclusions, which are briefly summarized as follows.

Quantitative and qualitative results have demonstrated the ability of our approach in unveiling lurking cases that are intuitive yet non-trivial. The best-performing ranking methods are those based on in-neighbors-driven and in-out-neighbors-driven lurking, i.e., the models emphasizing the first two principles underlying our lurking definition. These methods have shown high correlation with the data-driven ranking, and outperform competing methods, i.e., PageRank, alpha-centrality, Fair-Bets model, and the baseline in/out-degree ranking. Moreover, results tend to be relatively consistent over the PageRank-based and the alpha-centrality-based formulations of the lurker ranking methods. (We expect however that a different setting in the damping factor along with the introduction of a term modeling personalization or exogenous information in the respective formulas would bring to a more evident differentiation of the two ranking approaches.) From a runtime efficiency viewpoint, LRin tends to perform faster than ac-LRin, while ac-LRin-out achieves the highest rate of convergence although at the cost of much less diversified ranking scores. Furthermore, our qualitative analysis of the OSN pages of the top-ranked users has provided clear evidence that: (i) our approach successfully detects lurkers in an OSN, and conversely (ii) the competing methods fail in doing this — PageRank and alpha-centrality still detect influential users, whereas Fair-Bets tends rather to identify spammers.

From a pure network-analysis perspective, lurkers are not very prone to reciprocate each other, whereas preferential attachment is likely to occur be-

tween lurkers and the active users they are linked to. Under a percolation analysis framework, lurkers tend to be matched by users that are involved in links with low (directed) topological overlap: this would hint at a relation existing between lurkers and users playing the role of bridges between communities, under the assumption of lurking-oriented topological graph of an OSN. Finally, our proposed delurking-oriented randomization strategy reveals that self-delurking can be useful to change the top-ranked lurkers in the network, while scarcely affecting the in/out degree distribution.

5.7 Experimentation on dynamic contexts

In this section, we will deal with temporal information to enhance the understanding and ranking of lurkers in an OSN, providing an in-depth analysis of lurking in dynamic contexts structured in two ideal phases: (i) we will frame and try to answer four research questions that build our analysis of lurkers along the time dimension in a SN environment and (ii) we will provide the effectiveness results obtained by our time-aware LurkerRank methods, for both the transient and cumulative cases. Network analysis and ranking evaluation performed on Flickr, FriendFeed and Instagram networks allowed us to draw interesting remarks on both the understanding of lurking dynamics and on transient and cumulative scenarios of time-aware ranking.

5.7.1 Motivations and research questions

Online social environments are highly dynamic systems, as individuals join, participate, attract, cooperate, and disappear across time. This clearly affects the shape of the network both in terms of its social (followship) and interaction graphs (e.g., [77, 92, 101, 30, 99, 176]). Research on temporal network analysis and mining strives to understand the driving forces behind the evolution of OSNs and what dynamical patterns are produced by an interplay of various user-related dimensions in OSNs. Dealing with the temporal dimension to mine lurkers appears to be even more challenging. It's also an emergent necessity, as users in an OSN naturally evolve playing different roles, thus showing a stronger or weaker tendency toward lurking at different times.

In this section we provide insights into the understanding of lurkers from different perspectives along the time dimension in an OSN environment. Using Flickr, FriendFeed and Instagram as evaluation networks, we address four research questions which are summarized as follows.

Q1: *Do lurkers match to zero-contributors?* Definitions of lurking are often related to nonposting behavior. We aim at gaining insights into the correspondence between passive users and lurkers over time. Passive users are here intended as *zero-contributors*, i.e., users who have never posted, or provided a comment or favorite-mark.

Q2: *How frequently do lurkers respond to the others' actions?* Lurkers can show a limited amount of activity in response to others' contributions to the community life. We are interested in measuring the distribution of time latency that occurs to observe repeated actions by a user in response to her/his followees (i.e., comments, or favorite-marks).

Q3: *Do lurkers create preferential relations with active users?* In the third research question, our goal is to unveil the dynamics of the binding between lurkers and active users, and how this relates to the popularity of the active users.

Q4: *How does lurking behavior evolve?* We explore how lurking trends evolve over time, how they can be grouped together, and whether characteristic patterns may arise to indicate different profiles of lurkers.

5.7.2 Data

We used Flickr, FriendFeed and Instagram network datasets to conduct our analysis. *Flickr* and *FriendFeed* refer to the same datasets used for the static analysis (see Section 5.6.1), while *Instagram* (about 55K nodes, 1M links) is our latest dump recently crawled in 2014,[2] also studied in [55].

The selected datasets contain significant information on the temporal activities of users. *Flickr* contains timestamps of 34.7M favorite markings assigned to the uploaded photos, and also contains (inferred) timings on the user subscriptions. In *Instagram*, every link between v (follower) and u (followee) is annotated with the number and timestamp of the v's comments to media posted by u (about 2M comments and 1.7M likes). Analogous to *Instagram* is the situation in *FriendFeed* but for information concerning likes ($\approx$230K) and comments (>687K) to posts.

5.7.3 Lurkers vs. inactive users

To answer our first question (**Q1**), we initially analyzed how much the set of zero-contributors overlaps with the set of users having an in/out-degree ratio higher than one, here dubbed "potential lurkers". When considering the static picture of a network dataset, one remark is that the set overlap between zero-contributors and potential lurkers may vary from 12% (favorite-based interaction network in *Flickr*) to 72% and 95% (comment-based interaction networks in *FriendFeed* and *Instagram*, respectively). Moreover, since the relative difference in size of the two sets can vary from one dataset to another, we also computed the overlap ratio w.r.t. the set of potential lurkers, which was found to be 57% on *Flickr*, 62% on *Instagram*, and 96% on *FriendFeed*. There are hence clues that the overlap (or overlap ratio) would be relatively smaller when favorite/like interactions are taken into account, that is, potential lurkers are more likely to behave similarly to passive users when activity is regarded in terms of commenting.

[2] Available at *http://uweb.dimes.unical.it/tagarelli/data/*.

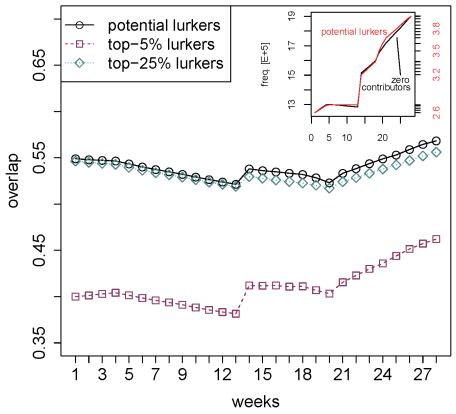


Fig. 5.11. Overlap ratio of zero-contributors against potential lurkers and top-ranked lurkers: distributions over weekly snapshots of the Flickr network. The inset shows the weekly distributions of zero-contributors and potential lurkers.

We further investigated how the relation between inactive and lurking users evolve over time. In this analysis, we also included the set of top-ranked users obtained by our LR algorithm. Figure 5.11 shows the temporal trends of overlap ratios w.r.t. potential lurkers, top-5% and top-25% ranked lurkers, on *Flickr*. Interestingly, the overlap ratios remain rather unaffected over time, despite the jump in frequency at the 14-th week (displayed in the inset). The distribution of top-5% ranked lurkers is always above the other two series (up to 0.15), which in turn roughly match. Note that in the inset, the distributions of potential lurkers and zero-contributors actually follow close trends, although they are scaled differently (on one order of magnitude).

5.7.4 Responsiveness

Concerning question **Q2**, we examined the distribution of time differences (in days) between any two consecutive responsive actions made by a user w.r.t. a post created by her/his followees. Figure 5.12 shows the empirical cumulative distribution functions over the first 90 days, for comments on *Instagram* and for favorites on *Flickr*; main chart areas refer to the top-ranked lurkers (in fractions of 5% and 25%), while the insets refer to all users. For both networks, the lurkers' responsiveness frequency is of the order of several days, or weeks. In fact, to observe 80% of responses, about 18 days would pass in *Flickr*, but nearly one month in *Instagram* for the top-25% lurkers (and even longer, i.e., more than 40 days, for the top-5% lurkers). In any case, the lurkers' responsiveness frequency appears to be twice slower than that exhibited by the users in general: looking at the insets in the two charts, it is clear that users tend to repeat comments/favorites to others' posts more rapidly, i.e., less than 20 (resp. 10) days on *Instagram* (resp. *Flickr*).

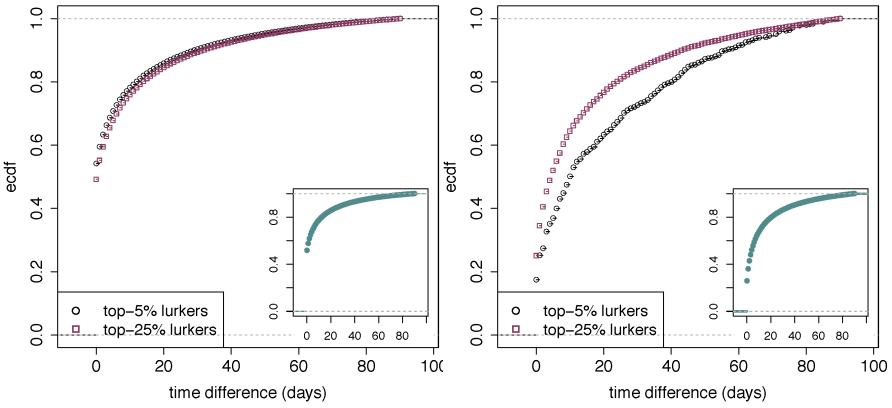


Fig. 5.12. Responsiveness frequency: cumulative distributions of lurker's reaction latency (in days), based on favorites (*Flickr*, on the left) and on comments (*Instagram*, on the right). Comparison w.r.t. generic user's reaction latency is shown in the insets.

5.7.5 Preferential attachment

We focus now on the relations between lurkers and the "active" users they are linked to. For this purpose, we selected the set of lurkers and the set of active users respectively from the top and the bottom of the LR ranking solution.

We first investigated whether the probability of observing active users with a certain degree of attached lurkers, and vice versa, can be predicted by a power-law. Figure 5.13 shows the distribution of lurkers as function of the degree of attached active users, and dually for the distribution of active users, obtained on the *Flickr* and *FriendFeed* followship graphs, using the top-25% and bottom-25% of the LR ranking solution. We computed the best fit of a power-law distribution to the observed data, and assessed the statistical significance of the fitting by a Kolmogorov-Smirnov test. From the figure it can be noted that the plots follow a power-law behavior. The exponents of the fitted power-law distributions are 1.725 ($x_{min} = 1$) and 1.363 ($x_{min} = 1$) for *Flickr* (Fig. 5.13(a) and (b), resp.), 2.015 ($x_{min} = 315$) and 1.679 ($x_{min} = 99$) for *FriendFeed* (Fig. 5.13(c) and (d), resp.). In all cases, the power-law fitting is statistically significant, with Kolmogorov-Smirnov test statistic (resp. p-value) of 0.0236 (resp. 0.8006) for Fig. 5.13(a), 0.0396 (resp. 0.7662) for Fig. 5.13(b), 0.0516 (resp. 0.9946) for Fig. 5.13(c), and 0.0546 (resp. 0.9161) for Fig. 5.13(d).

Our main goal to answer question **Q3** is to try explaining the relation between lurkers and active users in terms of preferential attachment, that is, we hypothesize that lurking connections are attached preferentially to active users that already have a large number of connected lurkers. This is also consistent with previous research in which preferential attachment has been studied as a growth model also for directed networks, such as Wikipedia [33] and Flickr [124]; however, in our context, such a type of analysis becomes more

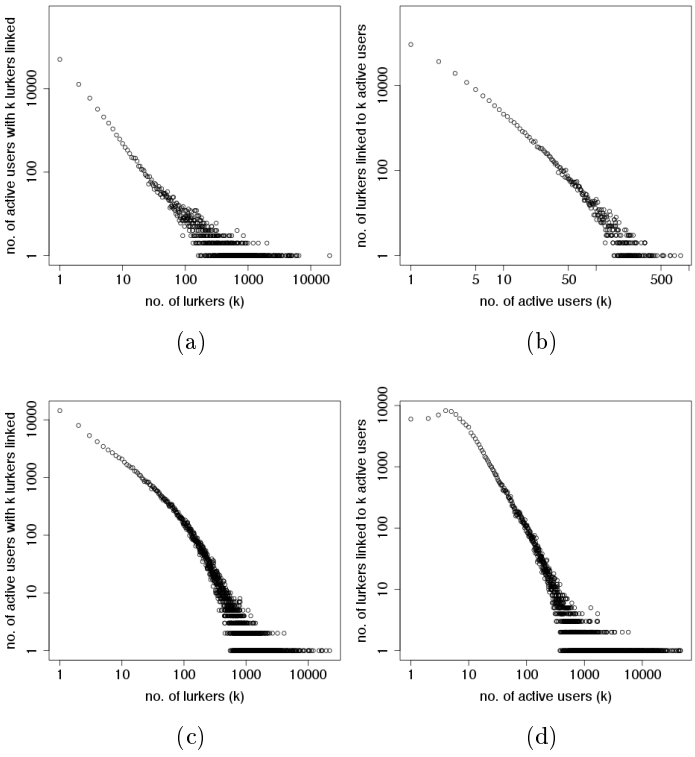


Fig. 5.13. Distribution of active users as a function of the lurkers-followers (a)-(c) and distribution of lurkers as a function of the active users-followees (b)-(d). (*Flickr*, two plots from the left, and *FriendFeed*, two plots from the right).

complicated since nodes (and their neighborhood) must be selected according to their lurking or activity properties.

Following the lead of [124], we studied two separate cases of "attachment": intuitively, these correspond to new connections received by active users for any k lurkers, and to new connections produced by lurkers for any k active users. Figure 5.14 shows results obtained on *Flickr*, averaged per user and per week, for each k. It can be noted that the number of lurkers shows a good linear correlation with the average number of new links received by active users (left-hand side of Fig. 5.14): the least-squared-error linear fit has a slope of 0.00836, which means that on average active users receive per week one new connection from lurkers for every 120 connections (lurkers) that they already have. By contrast, no preferential attachment behavior occurs for the opposite situation (right-hand side of Fig. 5.14), that is, lurkers that have a higher number of active users as followees are not more likely to create new connections to other active users.

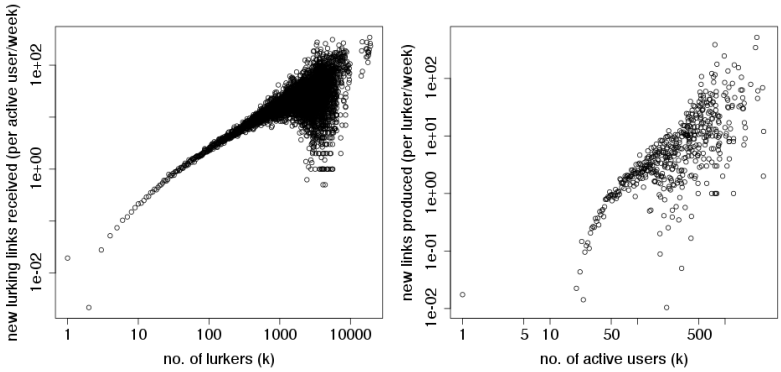


Fig. 5.14. Preferential attachment: lurkers vs. active users. New connections are detected for each weekly-aggregated network (on *Flickr*).

5.7.6 Temporal trends and clustering

To answer our fourth question (**Q4**), we aim at revealing structures hidden in the lurking trends that vary over time. We pursue this goal as a task of clustering of time series representing the users' lurking profiles. The basis for this clustering analysis lies in repeatedly applying our LR method to successive snapshots of a network dataset. Since the graph snapshots can vary in size, LR scores were first normalized to be comparable across different times. We then generated a time series of the normalized LR scores for every user in the dataset. The resulting set of time series was the input for our clustering task.

We adopted a *soft clustering* approach to group the time series of LR scores. This implies that a time series is allowed to obtain fuzzy memberships to all clusters. Our choice is motivated by the suspect that the natural clusters to be detected in this kind of time-course data could not be well-separated, rather they could be frequently overlap. A suitable method to detect clusters in this kind of data is based on *fuzzy c-means clustering*. We used a particularly efficient implementation, provided by the Mfuzz R-package tool,[3] based on minimization of the weighted square error function. Note that since the clustering is performed in Euclidean space, the time series were standardized to have a mean value of zero and a standard deviation of one. This preprocessing step ensures that series with similar variations are close in Euclidean space. Yet, the time series under study are expected not to have local time shifting, and they mostly are of the same length, which does not raise the need of a dynamic time warping approach to the similarity detection in our context. As concerns the setting of the fuzzifier and the number of clusters required by the clustering algorithm, we follow the methodology suggested in [143].

[3] *http://itb.biologie.hu-berlin.de/~futschik/software/R/Mfuzz/*.

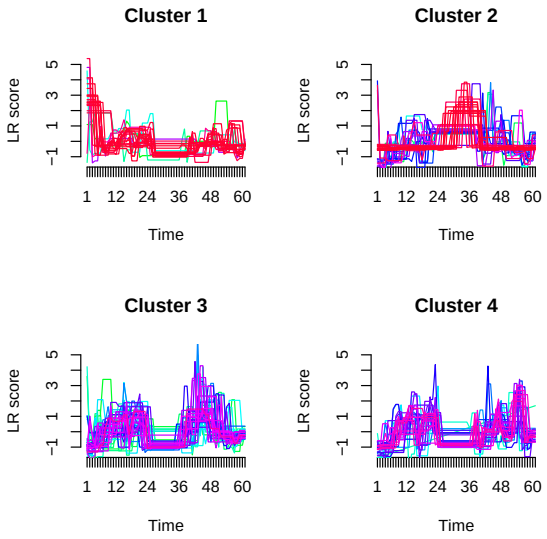
Figures 5.15 and 5.16 show some of the clustering results we obtained on the evaluation datasets. For this analysis, we initially selected the top-25% lurkers of the network snapshot at time zero, then we kept only those users appearing in at least 50% of the subsequent snapshots. Results correspond to different scenarios, both in terms of time-granularity (which impacts on the time series length) and type of relation (i.e., comment, favorite-mark, like+comment) underlying the graphs from which the time series were generated. Note that the membership values of time series are color-encoded in the plots, which facilitates the identification of temporal patterns in the clusters.

It can be noted from the figure that some cases are characterized by quite evident trends. For instance, on *Flickr*, cluster#2 groups lurkers whose behavior (lurking scores) evolves in the form of a series with an initial plateau followed by an increasing ramp and then a decreasing ramp, finally by a new stagnation trend. Similar is the situation depicted by cluster#3 on *Flickr*. On *FriendFeed*, clusters#1-#3-#4 present a more or less marked period of roughly constant lurking behavior between the 24th and 36th weeks, along with various peaks in the heads or tails of the series, which would hint at particularly critical (passive) periods of lurking. In general, more time-consuming actions (i.e., comments on *Instagram*, like+comments on *FriendFeed*) tend to correspond to trends that present sharper upward/downward shifts, and to clusters with more noisy data. Finally, note that except for cluster#1 on *Flickr*, lurking series do not tend to group into decreasing trends, which would suggest that lurkers are not likely to spontaneously "de-lurk" themselves, i.e., to turn their behavior into a more active participation to the community life.

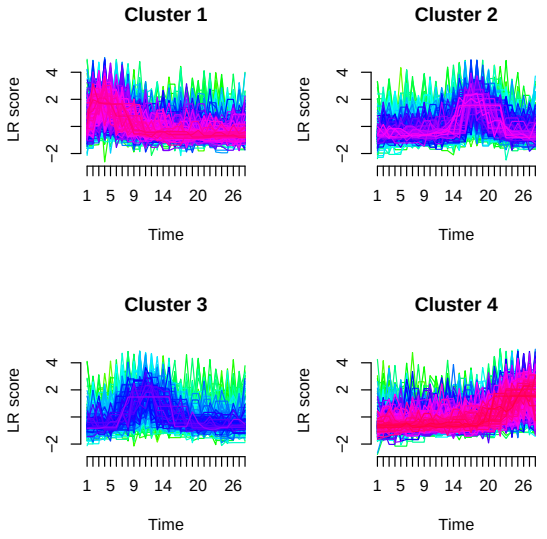
5.7.7 Evaluation of Time-aware LurkerRank algorithms

Data-driven evaluation

According to Section 5.6.2, we generated a *data-driven ranking* (henceforth DD) for every network graph and used it to assess the proposed and competing methods. However, in our context, a network graph corresponds to the interaction graph (rather than the topology graph) induced at a given time configuration. On *Flickr*, we calculated the score of a node as directly proportional to its in/out-degree (Laplace add-one smoothed, cf. Section 4.3) and inversely exponentially with a dataset-specific measure of influence: $r_v^* = \frac{in(v)}{out(v)} \exp(-EI(v))$. $EI(\cdot)$ denotes the empirical measure of influence (inspired by the one proposed in [14] for Twitter) which is used to estimate the influence of a user based on the number of favorite-markings her/his posts received. For a user u, $EI(u) = \frac{1}{out(u)} \sum_{v \in R(u)} nFavorites(v)$, where $nFavorites(v)$ is the number of favorites made by v. We defined analytically similar functions for *FriendFeed* and *Instagram*, in which the exponent in the empirical measure of influence has been redefined in terms of number of comments. Note again a different DD was generated per dataset and per time-window, in both transient and cumulative cases.



(a)



(b)

Fig. 5.15. Clustering of the time series representing LR scores in time-evolving graph networks: (a) *FriendFeed* on daily snapshots built on like+comment relations, (b) *Flickr* on weekly snapshots built on favorite relations. Warmer colors correspond to series with higher cluster-membership.

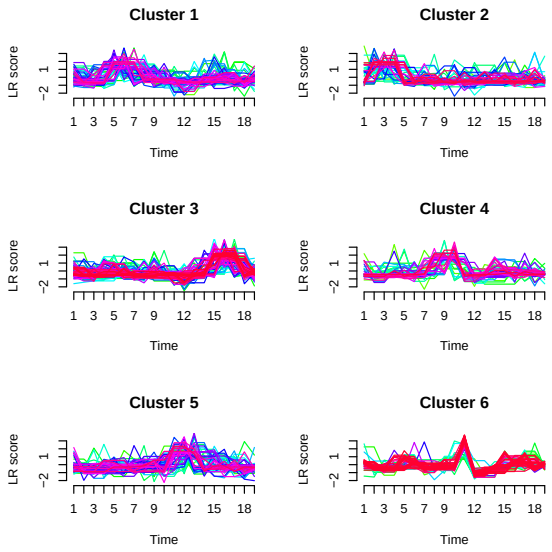


Fig. 5.16. Clustering of the time series representing LR scores in the time-evolving graph network of *Instagram* monthly snapshots built on comment relations. Warmer colors correspond to series with higher cluster-membership.

Competing methods

We compared our proposed methods Ts-LR and Te-LR against the (time-unaware) LurkerRank (LR) [162]. We also compared Ts-LR and Te-LR against T-Rank*, a modified version of the T-Rank algorithm [19] in which, to avoid bias w.r.t. our algorithms, the personalization vector is kept as uniform over all nodes. We used the setting of the parameters in T-Rank as suggested in [19].

Results

Table 5.16 reports on results for both the time-static case (upper subtable) and the time-evolving case (bottomer subtable). For any given dataset and setting of the weighting function, the reported value is the average of the differences between the Kendall-tau correlation achieved by our method vs. the reference DD,[4] and the Kendall-tau correlation achieved by a competitor vs. DD; each difference computation corresponded to a week. Note the first four rows in each subtable correspond to singleton time-static (resp. time-evolving) weighting, i.e., $f_T(\cdot)$ (resp. $\widehat{cf}_T(\cdot)$), $a_T(\cdot)$ (resp. $\widehat{ca}_T(\cdot)$), and their user-to-user function counterparts.

[4] Correlation values were in the range 0.45-0.63 on *Flickr*, and 0.78-0.95 on the other datasets. Recall that Kendall-tau correlation ranges within [-1,1].

Table 5.16 Average difference (per week) of Kendall-tau correlation with DD: comparison between the proposed Ts-LR and Te-LR against LR and a modified version of T-Rank [19]:

setting of the weighting function Eq. (5.16)	*Flickr*		*FriendFeed*		*Instagram*	
	Ts-LR vs. LR	Ts-LR vs. T-Rank*	Ts-LR vs. LR	Ts-LR vs. T-Rank*	Ts-LR vs. LR	Te-LR vs. T-Rank*
$\omega_1 = 1$	.107	.209	.221	.204	.201	.214
$\omega_2 = 1$	.221	.231	.224	.316	.212	.296
$\omega_3 = 1$	.046	.122	**.244**	**.340**	.274	.384
$\omega_4 = 1$	.012	.104	.021	.179	.031	.108
$\omega_1 = \omega_2 = 0.5$	**.223**	**.232**	.221	.255	.208	.302
$\omega_1 = \omega_3 = 0.5$	.102	.146	.242	.307	**.276**	**.385**
$\omega_2 = \omega_4 = 0.5$	.079	.142	.088	.201	.108	.166
$\omega_i = 0.25$ (i=1..4)	.094	.186	.103	.226	.128	.249

setting of the weighting function Eq. (5.20)	*Flickr*		*FriendFeed*		*Instagram*	
	Ts-LR vs. LR	Ts-LR vs. T-Rank*	Ts-LR vs. LR	Ts-LR vs. T-Rank*	Ts-LR vs. LR	Te-LR vs. T-Rank*
$\omega_1 = 1$	.240	.454	**.322**	.432	.326	.452
$\omega_2 = 1$	.261	**.521**	.227	.389	.222	.409
$\omega_3 = 1$	.180	.403	.318	.428	.336	.454
$\omega_4 = 1$	.091	.211	.128	.203	.103	.218
$\omega_1 = \omega_2 = 0.5$	**.276**	.412	.215	.371	.204	.393
$\omega_1 = \omega_3 = 0.5$	.207	.488	.317	**.454**	**.341**	**.471**
$\omega_2 = \omega_4 = 0.5$	.198	.376	.084	.201	.114	.203
$\omega_i = 0.25$ (i=1..4)	.206	.423	.168	.362	.195	.371

Bold values correspond to the best scores per method comparison, per dataset.

A first important remark concerns the comparison of the proposed Ts-LR and Te-LR against LR: the former consistently outperformed the (time-unaware) LurkerRank (with average gains up to 0.21 on *Flickr*, 0.22 on *FriendFeed*, and 0.23 on *Instagram*), confirming our initial hypothesis of increased performance when temporal aspects are integrated into the ranking algorithm.

Focusing on the combinations of the freshness and activity functions that determine the best-performing form of the LR weighting function, we observed stronger improvement (upon LR and the competitor T-Rank*) due to the user activity trend function (on *Flickr*), and both freshness functions, used as singleton or combined. Accounting for variations in activity trend appears to be more beneficial on a favorite-based network like *Flickr*, whereas information concerning freshness would have a better potential on comment-based networks.

The Te-LR method generally achieved higher performance gains (w.r.t. the reference DD) than the transient ranking method (Ts-LR) against both LR and T-Rank*. Both Te-LR and Ts-LR consistently outperformed T-Rank*, regardless of the particular setting of the weighting function. In detail, on average: 0.17 on *Flickr*, 0.25 on *FriendFeed* and 0.26 on *Instagram* in the time-static case, 0.41 on *Flickr*, 0.36 on *FriendFeed* and 0.37 on *Instagram* in the time-evolving case. Moreover, higher gains were obtained by using the relation-freshness (i.e., $\omega_3 > 0$) and the node-activity trend functions (i.e.,

$\omega_2 > 0$); note in particular that T-Rank makes use of a simple notion of activity amount rather than activity trend [19].

Finally note that the gains obtained by Ts-LR and Te-LR on *Flickr* were generally lower than those achieved on *FriendFeed* or *Instagram*. This could indicate that the "comment" type of interaction would act as a better discriminant than "favorite/like" to capture the lurker dynamics via a time-varying graph model.

5.8 Applications to other domains

We believe that the problem of identifying and ranking lurkers finds application in a variety of information networks other than the online social network scenario described in previous sections. Here we discuss two different instances of the lurker ranking problem concerning the scenarios of social trust and research collaboration networks. A common principle that characterizes lurkers in any information network is that for those nodes there is often an unbalance between the information consumed with respect to the information produced.

5.8.1 Lurking and social trust

Measuring trust behaviors has long been an important topic in psychology and social science [2, 117]. Trust is a complex relationship: deciding the trustworthiness of a user relies on a host of factors, such as personal relationship and past experiences of that user with her/his friends, and opinions about actions that the user has made in the past. In social network analysis from a computer science perspective, most existing studies have mainly focused on behavioral aspects discriminating between users who play "good" roles (e.g., reliable or influential users) and users who play "bad" roles (e.g., spammers) [34, 60, 8, 71, 81]. In any case, regardless of the specific task being addressed (e.g., trust prediction [68, 107, 103, 182], trust/distrust ranking [170, 129, 168, 134, 66]), research on trust computing has normally depended on the variety of *active* behaviors shown by the users in an online community. These behaviors are generally expressed at different levels and intentions of information production, that is, trustworthy/influential users typically produce good or useful contents, whereas untrustworthy/spamming users produce malicious or undesired contents and links.

An unexplored path in trust computing concerns *lurking* users. To a certain extent, this is quite surprising since, as we already stated in previous sections, lurkers represent the large majority of members in any online community. Lurking is often explained by a subjective reticence (rather than malicious motivations) to contribute to the community wisdom, and active users tend to avoid wasting their time with people who are very likely to not reply or show slow responsiveness, or who have few/bad feedbacks. This is a common scenario not only in question-answering systems (where timeliness is crucial)

but also in any social media network with microblogging services. Therefore, lurkers could in principle be perceived as *untrustworthy* users.

However, because of the lack of user-generated content and of the limited activity in the community life that characterize lurkers, determining trust or distrust relationships that involve lurkers appears to be a challenging task. In effect, to the best of our knowledge, no study so far has investigated possible relations between trustworthy/untrustworthy users and lurkers.

The subsequent study aims at pushing forward research on lurking analysis and mining by investigating how and to what extent lurkers are related to trustworthy and untrustworthy users. A new perspective in social trust analysis is introduced, which is built on the awareness that most people are lurkers in social networks. Moreover, we propose to use, in social networks without any explicit trust indicators, an entropy-based user function that infers the likelihood of a user to be trustworthy.

5.8.1.1 Comparative Evaluation

Methods

In this study we comparatively evaluate lurker ranking methods (cf. Section 5.4) against classic approaches to trust/distrust ranking, namely TrustRank and Anti-TrustRank (cf. Chapter 2, Section 2.6.1). For TrustRank we devised an alternative to step 1) of the algorithms which uses a ranking based on the add-one smoothed *in-degree/out-degree* ratio of the nodes instead of inverse-PageRank to perform the preliminary ordering of the nodes for the seed set selection. Similarly, we devised an alternative step 1) of Anti-TrustRank in which the *out-degree/in-degree* is used instead of PageRank. To distinguish between the two alternatives we will indicate with TrustRank_InvPR (resp. Anti-TrustRank_PR) the original algorithm, and with TrustRank_IO (resp. Anti-TrustRank_OI) the alternative one.

Discussion

The target of this evaluation, TrustRank and Anti-TrustRank, belong to the global trust ranking category (cf. Chapter 2, Section 2.6), and as such their ranking results are compared with those produced by LurkerRank methods. We would like to point out that our selection of such competitors is explained by the following motivations:

- Both methods involve a biased PageRank algorithm in their computation core, where the bias depends either on trust information or distrust information.
- TrustRank is a de-facto standard in global trust ranking, and Anti-TrustRank normally represents the counterpart of TrustRank for distrust ranking.

Table 5.17. Main structural characteristics of the evaluation network datasets.

data	*# nodes*	*# links*	*avg in-degree*	*avg path length*	*cluster. coef.*	*assortativ.*
Advogato	7,422	56,508	7.61	3.79	0.093	-0.069
Epinions	131,828	841,372	6.38	4.53	0.081	-0.064
Flickr	2,302,925	33,140,018	14.39	4.36	0.107	0.015
FriendFeed	94,779	3,269,303	34.49	3.30	0.019	-0.144

- Besides the damping factor required for the PageRank computation, the only parametric aspect of both methods is related to the seed set selection, whose outcome is much more easily understandable than other parameters (such as maximum length of path in the trust graph or minimum trust threshold).
- They allowed us to focus the evaluation on trust and distrust analysis separately; by contrast, methods such as PageTrust and PolarityTrust jointly computes trust and distrust scores but with increased storage and indexing costs.
- They are scalable, since their computation is linear in the size of the trust graph; this in general represents an advantage with respect to local trust ranking, which by definition requires a cost at least quadratic in the size of the trust graph.

Data

Our experimental evaluation was performed on four datasets, two of which are *who-trusts-whom* networks and the other two are *followship* networks.

We used the trust networks of Advogato.org and Epinions.com, which are de-facto benchmarks for trust analysis tasks. We built our *Advogato* network dataset by aggregating the daily-snapshot graph files available at the www.trustlet.org site, which cover the period Jan 1, 2008 - Apr 2, 2014. Edges in the *Advogato* network graph are labeled according to three different levels of certifications (trust links), namely *master*, *journeyer*, *apprentice*; a user without any trust certificate is called an *observer*. For each link from user u to user v, in the final aggregated graph we kept the last certification given by u to v. *Epinions* is the trust/distrust network studied in [115]. This network consists of about 132K users who issued above 841K statements (links).

We also used the followship graphs of two social media networks, Flickr and FriendFeed. We used the entire *Flickr* data studied in [124], originally collected in 2006-2007. For evaluation purposes, we also retrieved information originally stored in the dataset about the number of favorite markings every user's photos had. Our *FriendFeed* dataset refers to [35]. In order to fairly exploit information on the "likes" every user received in the network, we used the maximal strongly connected component of the subgraph containing all users that received a "like" and their neighborhoods.

Graph models. The different characteristics of TrustRank and LurkerRank algorithms require the use of two different graph models, which have

Table 5.18. Different edge orientation in TrustRank and LurkerRank graph models.

description	$u \to v$	$u \leftarrow v$
u's certificate to v	TrustRank	LurkerRank
u likes v's post	TrustRank	LurkerRank

opposite edge orientation, as summarized in Table 5.18. Note that the Anti-TrustRank algorithm runs on the same graph as TrustRank, since it starts upon the transposition of the adjacency matrix.

Setup of trust /distrust ranking methods

In TrustRank the notion of human-checking for a page to be spam is formalized by a binary function called *oracle*. However, a human-based oracle may not always be available, and hence relying on it could limit the applicability and scalability of TrustRank and similar algorithms (like Anti-TrustRank). To deal with this issue, in this work we follow different approaches to the definition of oracle function and of "goodness/badness" of a user, depending on whether the data provide trust indicators that are *explicit* or *implicit*. We elaborate on the two cases next.

Explicit trust indicators *Advogato* and *Epinions* provide annotations on the trustworthiness/untrustworthiness of links between users, as previously discussed. We exploit such annotations to define the following oracle functions:

- *majority voting* (henceforth denoted as MV) over the set of trust/distrust statements that each user receives.
- *advogato-trust-metric* (henceforth denoted as AT), which exploits information on the user certifications available from the Advogato site.[5] This of course applies to *Advogato* only.
- *controversial scoring* (henceforth denoted as CS), which applies to *Epinions* only. Similarly to [114], CS is calculated for each user u as

$$CS(u) = \frac{trust(u) - distrust(u)}{trust(u) + distrust(u)}$$

  where $trust(u)$ (resp. $distrust(u)$) is the number of $+1$s (resp. -1s) received by u from her/his neighbors. A user with CS equal to 1 (resp. -1) is trusted (resp. distrusted) by all her/his neighbors.

To decide if a user is to be regarded as "good" or "bad", we again distinguish between *Advogato* and *Epinions*. For each of the oracle functions in *Advogato*, we defined two variants of *goodness*: (i) users that are certified as *master* are considered good (henceforth denoted as M), or (ii) users that are certified as *master* or as *journeyer* are considered good (henceforth $M|J$). Dually, we defined two variants of *badness*: (i) users that are certified as *observer*

[5] www.advogato.org/trust-metric.html.

are considered bad (henceforth denoted as O), or (ii) users that are certified as *observer* or as *apprentice* are considered bad (henceforth $O|A$). For *Epinions*, we defined goodness/badness for the CS function based on numerical thresholds at 0.5, 0.75; here, we aimed to resemble a mapping to ordinal scale of the Advogato certification levels. Notions of goodness/badness are straightforward for the MV function (henceforth $MV+$ to denote majority of trust certificates, whereas $MV-$ stands for the opposite).

Implicit trust indicators Unlike trust network data, online social networks (OSNs) do not contain explicit trust assessments among users. Nevertheless, behavioral trust information can be inferred from some forms of user interaction that would provide an intuitive way of indicating trust in another user. Adali et al. [3] have in fact demonstrated that retweet data are a valid mechanism to infer trust in OSNs like Twitter. Accordingly, we leverage information on the number of *favorite markings* received by a user's photographs in *Flickr*, and on the number of *likes* received by a user's posts in *FriendFeed*, as empirical indicators of trust.

In order to define an oracle function based on the above indicators of trust, we postulate that the higher the number of users that indicate trust in a user u (by means of implicit trust statements), the more likely is the trustworthiness of user u. We formalize this intuition as an *entropy-based oracle* function H, in such a way that for any user u

$$H(u) = -\frac{1}{\log N(u)} \sum_{v \in N(u)} p_v \log p_v \tag{5.22}$$

with $p_v = ET(v,u)/(\sum_{z \in N(u)} ET(z,u))$, where $N(u)$ is the set of neighbors of node u, and $ET(v,u)$ is the empirical trust function measuring the number of implicit trust statements (i.e., likes or favorites) assigned by node v to node u. Analogously to the trust networks case, we defined goodness/badness for the H function based on numerical thresholds equal to the median (Q_2) and third quartile (Q_3) of the distribution of H values over all users.

Assessment criteria

We assessed the ranking methods in terms of *Kendall tau rank correlation* coefficient and *Bpref* measure (cf. Chapter 2, Section 2.7). For what concerns the setting of *Bpref*, we considered as relevant the *good*-certificated (resp. *bad*-certificated) nodes, as computed by the TrustRank (resp. Anti-TrustRank) oracle function, and as irrelevant all the remaining nodes.

Results

We organize the presentation of our results as follows. We begin with an evaluation of each of the ranking methods to assess their ability of ranking trustworthy and untrustworthy users. Then we provide a comparative evaluation of the ranking methods in terms of correlation of their ranking results.

Table 5.19. Trustworthiness evaluation of LurkerRank and TrustRank performance (Bpref) on trust networks.

dataset	*Advogato*		*Epinions*		
*relevant set**	M	$M\|J$	$CS \geq 0.5$	$CS \geq 0.75$	$MV+$
LRin	.135	.598	.290	.240	.336
LRin-out	.142	.604	.299	.247	.345
LRout	*.486*	*.673*	.789	.731	.825
acLRin	.124	.596	.304	.253	.351
acLRin-out	.173	.609	.288	.235	.336
acLRout	*.486*	*.673*	**.837**	**.780**	**.872**
TrustRank_InvPR	.659	.718	.791	.737	.826
TrustRank_IO	**.733**	**.767**	*.809*	*.765*	*.838*

* The *relevant* set corresponds to the set of *good*-certificated nodes, as computed by the oracle function of each method.

Table 5.20. Untrustworthiness evaluation of LurkerRank and Anti-TrustRank performance (Bpref) on trust networks.

dataset	*Advogato*		*Epinions*		
*relevant set**	O	$O\|A$	$CS < 0.5$	$CS < 0.75$	$MV-$
LRin	.228	.243	.640	.650	**.637**
LRin-out	.300	.294	.631	.645	.627
LRout	.050	.086	.188	.220	.167
acLRin	.279	.282	.627	.640	.622
acLRin-out	*.364*	*.347*	**.641**	**.654**	.636
acLRout	.050	.086	.144	.175	.123
Anti-TrustRank_PR	.287	.303	.422	.453	.401
Anti-TrustRank_OI	**.375**	**.369**	*.470*	*.496*	*.456*

* The *relevant* set corresponds to the set of *bad*-certificated nodes, as computed by the oracle function of each method.

We recall main notations that will be used throughout this section. MV stands for majority voting criterion (further, $MV+$ and $MV-$ are used to denote majority of trusts and distrusts, in *Epinions*. M, $M|J$, O, $O|A$ refer to the four goodness/badness notions used for *Advogato* (i.e., master, journeyer, etc.) and AT stands for Advogato trust metric. CS refers to the controversial scoring function used for *Epinions*. H refers to the entropy-based oracle function used for the online social networks (henceforth, OSNs).

In the result tables reported in the following (Tables 5.19–5.26), we will show in bold the best-performing values per method, and in underlined bold the absolute best-performing values for the specific dataset.

Note that we tested TrustRank and AntiTrustRank with different sizes of the seed set, varying from 5% to 25% of the total number of nodes. We observed no significant variations in the ranking, therefore for the sake of brevity of presentation we will present results obtained with seed set size equal to 10%.

Trust and distrust evaluation

Table 5.19 and Table 5.20 report Bpref results for the evaluation of trustworthiness and untrustworthiness, respectively. We focus here on *Advogato* and

Epinions, since the explicit trust indicators such networks provide fit more closely to a ground-truth-like evaluation.

Looking at Table 5.19, TrustRank methods had better Bpref than LurkerRank methods in detecting trustworthy users, under both settings of oracle's goodness (i.e., M and $M|J$), on *Advogato*. However, on *Epinions*, TrustRank was able to achieve higher Bpref scores for the detection of trustworthy users only against the in-neighbors-driven and in-out-neighbors-driven variants of LurkerRank; by contrast, LRout performance was very close to that of TrustRank_InvPR, and even acLRout was the absolute best-performing method.

Concerning the evaluation of untrustworthy users (Table 5.20), on *Advogato*, Anti-TrustRank behaved better than LurkerRank, although the best-performing LurkerRank method (i.e, acLRin-out) was quite close to the best-performing Anti-TrustRank variant. Even more surprisingly, on *Epinions*, the in-neighbors-driven and in-out-neighbors-driven variants of LurkerRank achieved higher Bpref than Anti-TrustRank methods.

Ranking correlation analysis

LurkerRank vs. TrustRank. Tables 5.21–5.23 report the Kendall rank correlation obtained by comparing TrustRank and LurkerRank on the various datasets. Both in trust networks and OSNs, LRout and acLRout showed higher correlation with TrustRank than the other LurkerRank methods.

On the trust networks, the highest correlation corresponded to similar scores (i.e., 0.74 for *Advogato* and 0.72 for *Epinions*, both comparing LRout with TrustRank_InvPR). The various LurkerRank methods performed quite differently from each other: the gap of both LRin/acLRin and LRin-out/acLRin-out w.r.t. LRout/acLRout was smaller on *Advogato* (0.4 on average), while on *Epinions* the difference in correlation among the same methods was about 0.8. More specifically, while on *Advogato* LRin/acLRin and LRin-out/acLRin-out showed some correlation with TrustRank (in the range of 0.14-0.38), on *Epinions* the correlation was always negative (in the order of -0.2 w.r.t. TrustRank_IO and -0.09 w.r.t. TrustRank_InvPR).

As concerns evaluation on OSNs, LRout and acLRout again obtained higher correlation scores w.r.t. the other LurkerRank methods, but lower than the scores observed for trust networks (up to 0.62 on *Flickr* and 0.34 on *FriendFeed*). LRin/acLRin and LRin-out/acLRin-out showed some significant correlation with TrustRank (0.38-0.44) on *Flickr*, but no correlation on *FriendFeed*.

LurkerRank vs. Anti-TrustRank. Ranking correlation results obtained by comparing LurkerRank with Anti-TrustRank are reported in Tables 5.24–5.26.

A first remark is that the highest correlation scores were lower than those obtained when comparing LurkerRank with TrustRank in both trust networks. Moreover, again in contrast to the previous analysis vs. TrustRank, on *Advogato* the relative differences in performance among the LurkerRank methods were much less larger. On both trust networks, LRin/acLRin and LRin-out/acLRin-out generally showed higher correlation with Anti-TrustRank than LRout

Table 5.21. Kendall correlation between LurkerRank and TrustRank methods on *Advogato*.

	oracle, *goodness*	LRin	LRin-out	LRout	ac-LRin	ac-LRin-out	ac-LRout
TrustRank_InvPR	AT, M	.367	.276	**.741**	.305	.205	**.710**
TrustRank_InvPR	$AT, M\|J$	**.382**	**.293**	.738	**.323**	**.220**	.707
TrustRank_IO	AT, M	.356	.262	.723	.290	.193	.691
TrustRank_IO	$AT, M\|J$	.316	.217	.692	.244	.149	.661
TrustRank_InvPR	MV, M	.310	.238	.642	.264	.182	.613
TrustRank_InvPR	$MV, M\|J$	.284	.228	.577	.254	.184	.551
TrustRank_IO	MV, M	.352	.257	.713	.284	.189	.682
TrustRank_IO	$MV, M\|J$	.314	.212	.683	.238	.145	.652

Table 5.22. Kendall correlation between LurkerRank and TrustRank methods on *Epinions*.

	oracle, *goodness*	LRin	LRin-out	LRout	ac-LRin	ac-LRin-out	ac-LRout
TrustRank_InvPR	$CS \geq 0.5$	-.091	-.044	**.717**	-.039	-.049	**.683**
TrustRank_IO	$CS \geq 0.5$	-.211	-.201	.676	-.197	-.206	.651
TrustRank_InvPR	$CS \geq 0.75$	-.093	-.047	.716	-.041	-.052	.682
TrustRank_IO	$CS \geq 0.75$	-.210	-.204	.675	-.200	-.208	.650
TrustRank_InvPR	$MV+$	**-.089**	**-.043**	**.717**	**-.037**	**-.047**	**.683**
TrustRank_IO	$MV+$	-.212	-.201	.676	-.197	-.204	.651

Table 5.23. Kendall correlation between LurkerRank and TrustRank methods on *Flickr* (upper subtable) and *FriendFeed* (bottomer subtable).

	oracle, *goodness*	LRin	LRin-out	LRout	ac-LRin	ac-LRin-out	ac-LRout
TrustRank_InvPR	$H \geq Q_2$	.282	.440	.553	.443	**.410**	.552
TrustRank_IO	$H \geq Q_2$	.292	.434	.615	.440	.385	.610
TrustRank_InvPR	$H \geq Q_3$	.293	**.441**	.562	.445	.402	.561
TrustRank_IO	$H \geq Q_3$	**.298**	.439	**.618**	**.446**	.386	**.614**
TrustRank_InvPR	$H \geq Q_2$	-.035	-.045	.334	-.035	-.018	.334
TrustRank_IO	$H \geq Q_2$	-.031	-.041	.335	-.031	-.017	.335
TrustRank_InvPR	$H \geq Q_3$	**.024**	**.004**	**.340**	**.024**	**.010**	**.340**
TrustRank_IO	$H \geq Q_3$	.021	.002	.339	.021	.008	.339

and acLRout. By contrast, the correlation between LRin and Anti-TrustRank was very weak in both OSNs, where the highest scores were obtained by LRout/acLRout and acLRin-out (up to 0.48).

Remarks on seed-set selection and oracle functions. The methods used to select the seed set impacted differently on the four datasets. On *Epinions* TrustRank_IO showed higher correlation with LRin/acLRin and LRin-out/acLRin-out than TrustRank_InvPR, while the latter showed higher correlation with LRout and acLRout than the former. The seed set selection methods did not lead to large variation in performance on *Advogato* when comparing TrustRank with LurkerRank, while Anti-TrustRank_PR always showed higher correlation with LurkerRank than Anti-TrustRank_OI. An opposite situation was observed on OSNs, where Anti-TrustRank_OI showed higher correlation

Table 5.24. Kendall correlation between LurkerRank and Anti-TrustRank methods on *Advogato*.

	oracle, *badness*	LRin	LRin-out	LRout	ac-LRin	ac-LRin-out	ac-LRout
Anti-TrustRank_PR	AT, O	.477	.428	.369	.472	.354	.349
Anti-TrustRank_PR	$AT, O\|A$	.454	.419	.349	.460	.351	.332
Anti-TrustRank_OI	AT, O	.300	.307	.234	.332	.270	.227
Anti-TrustRank_OI	$AT, O\|A$	.313	.316	.244	.343	.276	.235
Anti-TrustRank_PR	MV, O	**.550**	**.485**	**.439**	**.540**	**.394**	**.413**
Anti-TrustRank_PR	$MV, O\|A$	.465	.409	.347	.452	.337	.328
Anti-TrustRank_OI	MV, O	.469	.423	.430	.467	.346	.406
Anti-TrustRank_OI	$MV, O\|A$	.345	.337	.285	.366	.288	.273

Table 5.25. Kendall correlation between LurkerRank and Anti-TrustRank methods on *Epinions*.

	oracle, *badness*	LRin	LRin-out	LRout	ac-LRin	ac-LRin-out	ac-LRout
Anti-TrustRank_PR	$CS < 0.5$	.310	.336	.183	.344	.301	.157
Anti-TrustRank_OI	$CS < 0.5$	.411	.435	.125	.444	.376	.086
Anti-TrustRank_PR	$CS < 0.75$	.314	.342	.178	.350	.303	.152
Anti-TrustRank_OI	$CS < 0.75$	**.418**	**.443**	.122	**.452**	**.383**	.082
Anti-TrustRank_PR	$MV-$	.303	.328	**.194**	.335	.293	**.166**
Anti-TrustRank_OI	$MV-$	.407	.431	.126	.439	.372	.087

Table 5.26. Kendall correlation between LurkerRank and Anti-TrustRank methods on *Flickr* (upper subtable) and *FriendFeed* (bottomer subtable).

	oracle, *badness*	LRin	LRin-out	LRout	ac-LRin	ac-LRin-out	ac-LRout
Anti-TrustRank_PR	$H < Q_2$	.127	.368	.236	.356	.422	.244
Anti-TrustRank_OI	$H < Q_2$	.182	.442	.296	.433	.468	.299
Anti-TrustRank_PR	$H < Q_3$	.120	.371	.238	.359	.431	.247
Anti-TrustRank_OI	$H < Q_3$	**.205**	**.459**	**.306**	**.450**	**.477**	**.308**
Anti-TrustRank_PR	$H < Q_2$	.185	.138	.399	.185	.060	.399
Anti-TrustRank_OI	$H < Q_2$	**.296**	**.242**	.356	**.296**	**.136**	.356
Anti-TrustRank_PR	$H < Q_3$	.186	.139	**.402**	.186	.062	**.402**
Anti-TrustRank_OI	$H < Q_3$	**.296**	**.242**	.356	**.296**	**.136**	.356

with LurkerRank than Anti-TrustRank_PR, while no clear trend can be identified when comparing Trust- Rank_IO and TrustRank_InvPr with LurkerRank.

As concerns the oracle functions, on *Advogato* LurkerRank generally showed higher correlation with TrustRank (resp. Anti-TrustRank) when using AT (resp. MV). By contrast, on *Epinions*, no significantly different effect was observed when using CS or MV.

Summary of findings and discussion

"To trust or not to trust lurkers?" is the question we raised in this study. In the attempt to give a first answer to it, we summarize main findings of our study.

The LurkerRank methods based on the out-neighbors-driven model (i) behaved as good as or even better than TrustRank methods in terms of Bpref, and (ii) they showed high Kendall correlation with TrustRank methods. These findings should be interpreted at the light of a major conclusion we had in our experimentation in Section 5.6 that LRout and acLRout are less effective in scoring lurkers than the other LurkerRank methods. This implies that LRout and acLRout are able to produce high ranking scores also for (relatively active) users that are likely to be trustworthy. In short:

Trustworthy users can be found among lurkers.

Concerning untrustworthiness, the in-neighbors-driven and in-out-neighbors-driven variants of LurkerRank (iii) achieved higher Bpref than Anti-TrustRank methods, and (iv) they showed moderate Kendall correlation with Anti-TrustRank methods in trust networks, but also poor correlation in social media networks. Note that point (iii) refers only to trust networks, wherein we assumed that those rates of information-production to information-consumption that are peculiar of lurking behaviors may be hidden in explicit trust/distrust links. In effect, when considering social media networks, no LurkerRank methods showed significant correlation with a method like Anti-TrustRank specifically designed to detect untrustworthy/spam users. Therefore, we would tend to state that:

Lurkers are not necessarily untrustworthy users.

5.8.1.2 TrustRank-biased LurkerRank

We provide here a preliminary insight on how to integrate the ability of detecting trustworthy users (featured by TrustRank) into our LurkerRank in order to improve the *trustworthiness* of the lurkers to be detected. The result is a new set of methods, we call *TrustRank-biased LurkerRank* methods, in which the uniform personalization vector of a LurkerRank method is replaced by the ranking vector produced by TrustRank over the same network.

Table 5.27 summarizes Kendall correlation values obtained on *Flickr* by a pairwise comparison between our LurkerRank methods, their TrustRank-biased versions (denoted as trust-LR), and the original TrustRank. Several observations stand out. First, looking at the first-column group of results, all LurkerRank methods showed positive correlation with TrustRank. This is interesting as it would indicate that the trustworthiness of users is likely to be considered when ranking lurkers; note that the LurkerRank behavior against untrustworthy users or spammers was already observed in our qualitative evaluation (cf. Section 5.6.7). By personalizing a LurkerRank method with TrustRank, the correlation with TrustRank itself generally increased (up to 0.72), as we expected. More interestingly, trust-LR methods still showed a strong correlation with their respective original LurkerRank methods. This suggests that introducing a trust-oriented bias in LurkerRank methods would

Table 5.27. Comparative performance (Kendall tau rank correlation) of TrustRank-biased LurkerRank methods against original TrustRank and LurkerRank methods, on *Flickr*.

	LR vs. TrustRank	trust-LR vs. TrustRank	trust-LR vs. LR
LRin	.393	.436	.639
LRout	**.562**	.556	**.980**
LRin-out	.441	.640	.688
ac-LRin	.445	.434	.728
ac-LRout	.561	.559	.945
ac-LRin-out	.402	**.724**	.498

Bold values refer to the highest scores per method.

not significantly decrease their lurker ranking effectiveness while also accounting for the user trustworthiness.

5.8.2 Lurking in collaboration networks

Research collaboration networks (RCNs) are being formed as prototypes of social networks on top of digital libraries. As for other major types of social networks, data management and mining research in RCNs has focused mainly on expert finding, community discovery, and relation (link) prediction (e.g., [45, 63, 151, 167, 171, 145]). All these tasks have to be aware of the interactions underlying research collaboration, and in this respect, the discovery of hidden *expert-apprentice* or *advisor-advisee* relationships among researchers is particularly important. Mining such relationships can be useful for several reasons such as understanding how a research community is formed in a particular institutional context, how research themes evolve over time, predicting whether a researcher will likely influence a research community, and aiding an expert finding application to foster several experts on specific topics.

In this context, the current trend is to push the search for expert-apprentice relationships towards an expert-oriented investigation of co-authorships. By contrast, it seems that a study of roles as "non-expert" has been neglected in RCN mining research, despite the fact that a significant part of members in a RCN is more likely to be apprentice: after all, an apprenticeship or training status not only clearly holds for the initial stage of a researcher lifetime, but also naturally holds for any researcher w.r.t. any topic that at a particular time does not represent her/his main research interests.

A particularly challenging type of relationship to discover from the apprentice perspective concerns *vicarious learning*. In social learning theory, vicarious learning refers to the notion that people can learn through being given access to the learning experiences of others, and in general it is seen as a function of observing, modeling and replicating the learning behavior of others [16]. However, while vicarious learning can be seen as legitimate in any training stage, a question becomes whether in a publication context it can still be

identified and measured in collaborations in which one might marginally contribute to the research activity (possibly materialized in a publication). The question is clearly tricky, since in reality it could be difficult even for human beings to judge who is vicariously learning by the track records related to a research community. Nevertheless, investigating the case of vicarious learners can offer us a chance to gain an insight into a research community that goes beyond the simple co-authorships. In particular, scoring RCN members based on their degree of vicarious learning should be seen as essential to determine the contingencies in the network under which different apprentice' behaviors occur; ultimately, it should aid devising both generic and ad-hoc strategies of eliciting more proactive participation in research activities, commitment to work in teams more collaboratively and trying to take influential actions, and building up strong collaborative subnetworks. Moreover, by analyzing the RCN over time, the changes in each apprentice's profile can be detected and several actions can be taken, for instance, to promote emerging researchers (e.g., with calls for tenure track positions).

In this study we address the novel problem of identifying and ranking vicarious learners in RCNs. We would like to point out that our usage of the term "vicarious learner" is intended to actually entail the meaning related to the more generic "advisee" and "apprentice"; however, as it will be clarified later, the term vicarious learner is chosen as preferred since it better fits the notion of knowledge gained in research collaboration. We introduce a topology-driven vicarious learning definition and present an adaptation of our Lurker Rank method for ranking vicarious learners. Results obtained on DBLP networks support the significance and uniqueness of the proposed approach.

5.8.2.1 The proposed model

Modeling vicarious-learning-oriented RCNs. A common assumption in RCNs is that two researchers are regarded as connected to each other if they have co-authored a paper. Co-authorships in a RCN are naturally treated as symmetric relations, however a directed graph model would be more convenient when the target of investigation is on scoring influential researchers, or as in our setting, vicarious learners. Authors' interactions in a RCN are typically modeled as influence-oriented relationships, and in fact traditional ranking approaches, including PageRank based methods for RCN analysis (e.g., [181, 48, 111, 110]), assume an influence graph model which implies that the more (or more relevant) incoming links a node has the more important it is; for instance, the more advisees a researcher supports the more authoritative or expert s/he might be. Clearly, as researchers might influence each other, both the incoming and outgoing links of a node should be taken into account. Moreover, vicarious learner behaviors build on the amount of information or benefit a node receives, therefore we believe that a convenient graph model is such that edges are drawn from experts to apprentices (except for cases of presumed equal relationships).

Our vicarious-learner-oriented RCN graph model assumes the availability only of basic information stored in a co-authorship network, at a given temporal window. Within this view, a reasonable solution is to look at each author's amount of publications to discriminate among the varying degrees of expertise/apprenticeship. Let us denote with $\mathcal{G}_t = \langle \mathcal{V}_t, \mathcal{E}_t, w_t \rangle$ a weighted directed graph representing a RCN at a discrete time interval t, with set of nodes (authors) $\mathcal{V}_t$, set of edges $\mathcal{E}_t$, and edge weighting function w_t. The semantics of any edge (i, j) is that j is likely most to benefit from the collaboration with i at time t, and the edge weight, $w_t(i, j)$, expresses the strength of the benefit received by j from i.

Drawing edges

A link $(i, j) \in \mathcal{E}_t$ is drawn from author i to author j if they are co-authors in some publication at time interval t, with author i having a total number of publications (as calculated at time t or earlier) greater than that of author j; in case of a tie, reciprocal edges are inserted between i and j.

Clearly, author's academic achievements, citations and productivity indexes, such as h-index, might also be taken into account to uncover hidden expert-apprentice relationships; however, we do not follow that line since the above information is either not always available or easily derivable from a RCN (for some author or for any given time) or it may be too coarse. To corroborate our choice of simply using the amount of publication to estimate the expert/apprentice relationships, we indeed evaluated the impact of exploiting a criterion based on h-index. For this purpose, we used the co-authorship network derived from the whole DBLP dataset (cf. Table 5.28) as case in point, and we also retrieved the authors' h-indexes from an external source, ArnetMiner.[6] At first, we changed our defined edge drawing model by replacing an author's total number of publications with her/his h-index, but this actually led to a significant increase in the number of reciprocal edges (from 10% to 65% of the total number of edges). Then, we set an edge (i, j) according to the comparison between the authors' h-indexes, and in case of a tie according to their number of publications: as a result, we surprisingly observed an overlap of 94% between this edge set and that based solely on number of publications; analogously, by inverting the two conditions for edge drawing (i.e., by amount of publication first, then by h-index), the overlap was about 99%.

Weighting edges

The number of co-authored publications between any two nodes can be regarded as an essential criterion for expressing the strength of collaboration; moreover, this simple criterion has the advantage of being readily available at any time interval. However, it should also be noted that an advisor has normally to divide her/his *attention* over all incoming stimuli that come

[6] http://arnetminer.org/ranks/author

from her/his advisees; consequently, the benefit each advisee might gain from her/his advisor could decrease as the number of advisees the advisor must support increases. Therefore, we weight the strength of (i, j) not only as directly proportional to the evidence of collaboration in terms of publications but also as inversely proportional to the number of publications co-authored by the advisor i with advisees other than j. Formally, at a time interval t:

$$w_t(i,j) = coPubs(i,j,t)\left(1 - \frac{\sum_{k \in advisees(i,t)\setminus\{j\}} coPubs(i,k,t)}{\sum_{k \in advisees(i,t)} coPubs(i,k,t)}\right)$$

where $coPubs(i, j, t)$ is the number of publications co-authored by i and j at time t, and $advisees(i, t)$ denotes the set of i's advisees (i.e., i's out-neighbors) at time t.

Ranking vicarious learners. In order to capture the intuition that researchers influence each other in a RCN, both the incoming and outgoing links of a node will be taken into account in our notion of vicarious learning and, consequently, in our proposed ranking method. Within this view, the simplest non-trivial form of a measure of vicarious learner ranking would be given by the ratio of the in-degree to the out-degree of a node. However, this has clearly the disadvantage of giving many nodes the same or very close ranking scores and it totally ignores that both in-neighbors and out-neighbors might contribute to the status of a given node. Learning from our study on silent actors in online social networks 5.4, our key idea is hence to determine a node's status of vicarious learner according to the following criteria:

- to be inversely proportional to the node's out-degree, and
- to be proportional to the number of its in-neighbors and to their likelihood of being non-vicarious-learners (i.e., advisors or expert researchers), which is expressed by a relatively high out/in-degree.

The above set of criteria states that determining a vicarious learning status for a node not only relies on its in/out-degree ratio but also on the extent to which its in-neighbors are rather influential nodes (i.e., advisors or expert researchers) as well as its out-neighbors may in turn show a vicarious learning behavior.

To provide a complete specification of the above intuition, we resort to the renowned PageRank [27], which has been widely applied to various types of RCNs [181, 48, 111, 110].

For any node $i \in \mathcal{V}_t$ in the RCN graph $\mathcal{G}_t = \langle \mathcal{V}_t, \mathcal{E}_t, w_t \rangle$, let $B_t(i)$ and $R_t(i)$ the set of i's in-neighbors and the set of i's out-neighbors at time interval t, respectively, and hence the sizes of $B_t(i)$ and $R_t(i)$ are respectively the in-degree and the out-degree of i. Upon the PageRank formula, our vicarious learner ranking method, named VLRank, is defined by the set of equations:

$$r_i = \alpha\left(\frac{1}{|R_t(i)|}\sum_{j \in B_t(i)} w_t(j,i)\frac{|R_t(j)|}{|B_t(j)|}r_j\right) + \frac{1-\alpha}{|\mathcal{V}_t|}, \qquad \forall i \in \mathcal{V}_t$$

Table 5.28. Main characteristics of the DBLP datasets: comparison between the global network (updated at March 2013) and the last three terms of three years.

time interval	*# nodes*	*# links*	*avg in-degree*	*avg path length*	*# source nodes # sink nodes*	*avg in/out-degree* *	*clustering coefficient*
-2013	1,191,619	4,712,489	3.95	7.50	54,647; 533,101	1.75	0.18
2004-2006	341,282	957,922	2.81	7.61	32,511; 139,016	1.33	0.44
2007-2009	469,345	1,412,556	3.01	7.16	40,021; 188,166	1.41	0.32
2010-2013	582,206	1,926,184	3.31	6.82	45,916; 227,990	1.50	0.28

* Sink nodes and source nodes are excluded.

Note that to deal with sink nodes and avoid infinite in/out-degree ratios, we add-one smoothed both the in-degree and out-degree of a node. Moreover, like for the basic PageRank, we implemented a power-iteration method for an efficient computation of the ranking scores.

5.8.2.2 Experimental Evaluation

Data

We used the DBLP XML data repository (dump updated at March 19, 2013[7]), from which we retrieved all types of publications and selected authors as well as editors—hereinafter we will simply refer to both as authors. We extracted information about the number of joint publications for each pair of co-authors, and the total number of publications for every author, on a yearly basis. We devised two stages of evaluation: in the first one we used all the DBLP data in order to test the methods on a RCN as large as possible, while in the second stage we selected three subsets corresponding to the last three terms of approximately three years in order to assess the behavior of the methods in capturing the temporal evolution of the identified vicarious learners. Table 5.28 summarizes characteristics of the various datasets. It should be noted that the number of authors intuitively increases during the years, up to about 70% of increment from 2004-06 to 2010-13; moreover, the simultaneous increase in the average in-degree and decrease in the average path length confirms the growth in cooperation among researchers (both at intra- and inter-institutional level) that has occurred in recent years. Conversely, the decrease of the clustering coefficient would indicate that co-authors of one author are less likely to publish together, however this should be ascribed to an increasing number of peripheral nodes (i.e., newcomers, which obviously may have very few co-authorships) that both 2010-13 and 2007-09 have w.r.t. the corresponding earlier three-year term.

Competitors and ground-truth evaluation

We compared the proposed VLRank method w.r.t. the following competitors: the *in/out-degree* (henceforth InOut) distribution of the nodes in the network

[7] http://dblp.uni-trier.de/xml/

Table 5.29. Comparative performance results: Kendall rank coefficient.

	-2013	*2004-06*	*2007-09*	*2010-13*
VLRank vs. InOut	.153	.249	.259	.256
VLRank vs. DDRank	.284	.283	.295	.298
PageRank vs. InOut	-.097	.182	.177	.177
PageRank vs. DDRank	.133	.246	.246	.255
VLRank vs. AMRank	.115	–	–	.148
PageRank vs. AMRank	.043	–	–	.083
VLRank vs. PageRank	.422	.424	.410	.407

dataset, as a baseline, and a weighted version of PageRank, as a related ranking method. However, given the novelty of the problem addressed, we had to face the total lack of a gold-standard for vicarious learning. For this purpose, we followed two alternative approaches to the ground-truth evaluation of the ranking methods. In the first approach, for each of the selected DBLP datasets, we generated a *data-driven ranking* (henceforth DDRank) of a node by exploiting the different semantics of the strength of the relationships with in-neighbors, which likely model a vicarious learning modality, and the relationships with out-neighbors, which should instead express a higher likelihood of scholar advising modality. Formally, for each node $i \in \mathcal{V}_t$:

$$r_i^* = \frac{1 + \sum_{j \in B_t(i)} w_t(j,i)}{1 + \sum_{j \in R_t(i)} w_t(i,j)} \exp(-nPubs(i,t))$$

where weights on incoming (resp. outgoing) edges count the number of co-authored publications with an in-neighbor (resp. out-neighbor), and $nPubs(i,t)$ is the number of single-authored publications by author i (at time t). In the second approach, we again exploited the academic statistics provided by ArnetMiner, and in particular the expert's *activity score* which is used to rank the researchers based on the cumulated weighted impact factor of one's papers published in the last years.[8] We hence defined an ArnetMiner based ranking (henceforth AMRank) as follows:

$$r_i^* = \frac{1 + \sum_{j \in H^+(i)} AS(j)}{1 + \sum_{j \in H^-(i)} AS(j)}$$

where $H^+(i)$ (resp. $H^-(i)$) denotes the set of i's co-authors who have h-index greater than (resp. lower than or equal to) the h-index of i, and $AS(j)$ denotes the ArnetMiner activity score for author j. Note that AMRank was not calculated for the subsets 2004-06 and 2007-09 since the activity scores are only available for the latest update of ArnetMiner.

Assessment criteria

In order to evaluate the performance of the ranking methods, we resorted to two well-known assessment criteria: *Kendall Rank correlation coefficient* and *Bpref* (cf. Chapter 2, Section 2.7).

[8] http://arnetminer.org/AcademicStatistics

Table 5.30. Comparative performance results: Bpref.

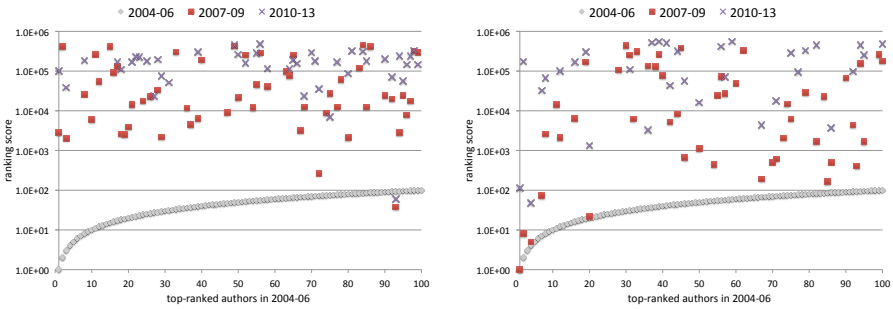
	-2013			*2004-06*			*2007-09*			*2010-13*		
	$p = 10$	$p = 25$	$p = 50$	$p = 10$	$p = 25$	$p = 50$	$p = 10$	$p = 25$	$p = 50$	$p = 10$	$p = 25$	$p = 50$
VLRank vs. InOut	.336	.584	.664	.362	.561	.702	.409	.583	.713	.415	.580	.714
VLRank vs. DDRank	.687	.784	.744	.605	.701	.706	.644	.726	.717	.667	.730	.720
PageRank vs. InOut	.099	.328	.467	.204	.449	.666	.211	.460	.670	.219	.469	.676
PageRank vs. DDRank	.481	.626	.592	.528	.639	.648	.544	.658	.654	.580	.671	.668
VLRank vs. AMRank	.191	.448	.645	–	–	–	–	–	–	.264	.508	.663
PageRank vs. AMRank	.131	.338	.573	–	–	–	–	–	–	.166	.385	.603
VLRank vs. PageRank	.804	.853	.857	.620	.726	.815	.656	.754	.834	.650	.735	.817

For what concerns the setting of *Bpref*, we distinguished two cases: when comparing w.r.t. the data-driven ranking, N was defined as the set of nodes with data-driven ranking score below or equal to 1 (which should be regarded as an intuitive indicator of non-vicarious-learning), otherwise, i.e., comparison w.r.t. competing methods, as the bottom of the corresponding method's ranking having the same size as in the case of data-driven ranking. R was selected as the set of nodes having top-p% score from the complement of N.

Results

Tables 5.29–5.30 provide performance results corresponding to the various DBLP networks. VLRank always obtained positive Kendall scores (Table 5.29) w.r.t. InOut, DDRank and AMRank, which tended to increase from 2004-06 to 2010-13, and were in general higher than in the largest dataset (-2013). More importantly, VLRank always achieved higher correlation with InOut, DDRank and AMRank than PageRank, with gains up to 21.7% for InOut, 11.8% for DDRank, and 6.5% for AMRank. Similar conclusions were drawn from Table 5.30, with VLRank always outperforming PageRank also in terms of Bpref, where the p% of relevant candidates for Bpref evaluation was set as $p = 10, 25, 50$. Bpref scores generally increased with p when comparing VLRank and PageRank with InOut and AMRank, which indicates that in both cases the similarity between the rankings was lower (resp. higher) when comparing the head (resp. tail) of the lists. Interestingly, in contrast to the above results that would put in evidence the different behaviors of VLRank and PageRank (in favor of the former), the relative similarity between the corresponding rankings by VLRank and PageRank, for both Kendall and Bpref evaluations (last rows of the respective tables), prompted us to investigate this more in detail.

We therefore aimed to confirm the presumed higher reliability of the rankings produced by VLRank in capturing vicarious learners. For this analysis, we compared the top-100 ranked lists produced by VLRank and PageRank on the whole DBLP network (-2013). VLRank detected different situations in terms of total amount of publications produced in career, total number of co-authors, and average number of co-authors per publication. For instance, a few authors have produced little research in their short career and always within a research team, whose consistency of composition can vary from case to case. Other authors have shown a relatively long career and published



(a) VLRank (b) PageRank

Fig. 5.17. Understanding the temporal evolution of vicarious learners.

several journal and conference papers together with many co-authors; for example, the 9th ranked author published 25 conference papers in the period 1999-2004 with an average of more than 10 co-authors per paper and a total of 17 co-authors. Overall, we can state that our VLRank detected and assigned highest scores to authors whose status can be tagged as vicarious learner with a certain objectivity. Conversely, PageRank did not behave as good as VLRank. The top-1 rank corresponded to an author with only one publication co-authored with other seven persons. Throughout the PageRank top-ranked list, we found authors with a potential status as vicarious learner, however we also found a few other top-ranked authors exhibiting characteristics that do not look typical of a vicarious learner. For instance, the 2nd ranked author published more than 50 journal and conference papers, jointly with other colleagues but with different research teams in the career period (2004-12); an analogous situation was found for the 3rd ranked author who is involved in a team. In both cases, we easily found by looking at the authors' CVs in their homepages, that the authors should be instead considered as team leaders or at least active contributors. The case corresponding to the 6th ranked author is even more clearly close to a research team leader: 60 publications among journal and conference papers in nine years, usually co-authored with few other colleagues in very different groups.

We further investigated the behaviors of VLRank and PageRank by comparing, for each of the methods, the distribution of the top-ranked authors in the 2004-06 snapshot w.r.t. the subsequent terms of three-years. This was useful to gain an insight into the temporal evolution of the authors recognized as vicarious learners at a certain time interval (2004-06) by VLRank and PageRank, respectively. Looking at Figure 5.17(a), a large number of top-100 authors by VLRank in 2004-06 were also present in the subsequent periods (60% in 2007-09, 44% in 2010-13), but with much lower ranks (in the range of 10^2-10^5): in effect, we then observed that such authors were effectively vicarious learners that later took on more expert roles; conversely, the authors present at 2004-06 but ranked higher later were effectively recognized

as new cases of vicarious learners due to a drastic decrease in their individual productivity. PageRank (Figure 5.17(b)) seemed to be less able to effectively capture the temporal evolution of vicarious learners, since many of the top-100 authors in 2004-06 were still ranked high in 2007-09 (most of these cases were concentrated at the very top of the ranking). Upon investigation of such situations, we found in particular five authors within the top-20 in 2004-06 that were still ranked within the top-100 authors in 2007-2009, even though no case of vicarious learning could be clearly recognized among them, but rather a prolific collaboration among peers belonging to the same research team.

5.9 Related work

5.9.1 Lurking and social network analysis

The topic of lurking has been long studied in social science and recently has gained renewed interest in the computer-human interaction community. [148] investigates relations between lurking and cultural capital, i.e., a member's level of community-oriented knowledge. Cultural capital is found positively correlated with both the degree of active participation and, except for longer-time lurkers, with de-lurking. [41] leverages the significance of conceptualizing the lurking roles in relation to their boundary spanning and knowledge brokering activities across multiple community engagement spaces. The study proposed in [38] raises the opportunity of rethinking of the nature of lurking from a group learning perspective, whereby the engagement of intentional lurkers is considered within the collective knowledge construction activity. The interactive/interpassive connotation of social media users' behavior is studied in [83], under a qualitative and grounded-theory-based approach. In the context of multiple online communities in an enterprise community service, lurking is found as only partially driven by the member's engagement but significantly affected by the member's disposition toward a topic, work task or social group [126]. Exploring epistemological motivations behind lurking dynamics is the main focus of the study in [141], which indeed reviews major relevant literature on epistemic curiosity in the context of online communities and provides a set of propositions on the propensity to lurk and de-lurk. However, as with [41], the paper only offers insights that might be useful to guide an empirical evaluation of lurkers' emotional traits. The study in [70] examines peripheral participation in Wikipedia, and designs a system to elicit lightweight editing contributions from Wikipedia readers.

To the best of our knowledge, there has been no study other than ours that provides a formal computational methodology for lurker ranking. The study in [54], which aims to develop classification methods for the various OSN actors, actually treats the lurking problem marginally, and in fact lurking cases are left out of experimental evaluation. Similarly, [94] analyzes various factors that influence lifetime of OSN users, also distinguishing between active

and passive lifetime; however, analyzing passive lifetime is made possible only when the user's last login date is known, which is a rarely available information.

We finally mention some research studies that have focused on latent relationships or side-effect benefits in an OSN. For instance, [9] defines a Stackelberg game to maximize the benefit each user gains extending help to other users, hence to determine the advantages of being altruistic. Some interesting remarks relate the altruism of users to their level of capabilities, and indicate that the benefit derived from being altruistic is larger than that reaped by selfish users or free riders. [113] also builds upon game theory to study the property of users' departure dynamics, i.e., the tendency of individuals to leave the community. [180] studies the problem of identifying the off-line real-life social community of a given user, by analyzing the topological structure in an on-line social network like Twitter. To the purpose, user interactions are modeled in the form followee-to-follower (like in our setting), and a PageRank-like algorithm is applied over a probability transition matrix that embeds three key principles underlying the notion of off-line community, namely mutual reachability, friendship retainability, and community affinity. It should be noted that mutual reachability is not a peculiar characteristic of lurkers, i.e., it can hold for active users as well. Moreover, as for the community affinity principle, lurkers are usually not grouped into communities such that each community members are (indirectly) connected to each other; rather, as we have discussed in this chapter, lurkers may lay on the boundary of a component and bridge over other components.

Relations with existing definitions of lurking

Our definition of lurking is substantially consistent with the various existing perspectives on lurking, previously mentioned in the Introduction. It can in general recognize and measure behaviors that rely on phenomena of lack of information production (i.e., inactivity or occasional activity) as well as on phenomena of information hoarding or overconsumption, like free-riding and leeching.

It is worth emphasizing that taking into account the authoritativeness of the information received as well as the non-authoritativeness of the information produced by lurkers is essential to the correct scoring of lurkers. Therefore, our definition of lurking can also explain more complex perspectives, such as legitimate peripheral participation. In this case, a lurker is regarded as a novice, for which it's legitimate to learn from experts as a form of cognitive apprenticeship. Indeed, by applying our LurkerRank methods, in [161] we have addressed an exemplary form of legitimate peripheral participation, known as vicariously learning, in the context of research collaboration networks.

Finally, note that other interpretations of lurking, such as microlearning and knowledge sharing barriers, actually aim to understand the various reasons for lurking, and to what extent they might be perceived as fruitful, rather

than neutral or harmful, for the knowledge sharing in the online community. Therefore, they mostly involve sociological and psychological aspects whose study is beyond the objective of our work.

5.9.2 Research collaboration networks

As mentioned in Section 5.8.2, major attention in co-authorship network analysis has been paid to the discovery of author impact (e.g., [63, 181, 48]), and more recently focusing on expert search (e.g., [45, 167]) and link prediction (e.g., [151]). Focusing on contexts closer to our study, [171] proposes a time-constrained probabilistic factor graph model to discover advisor-advisee relationships. Like in our case, the study exploits only information available on the network, with neither text annotations nor supervised information such as labeled relations (i.e., who is advisor of whom); however, the goal in [171] is to rank advisors for every author. [106] builds a linear regression model on PageRank scores to predict an improvement in productivity of the researchers. The method exploits mutual influence between each pair of co-authors, which significantly impacts on the ranking graph size, while it makes a straightforward adaptation of PageRank that only utilizes out-going links to score an author-node. Moreover, each author is initially assigned a weight that estimates the quality of her/his publication set based on an external, predefined rating scheme of the publication venues, which is clearly controversial by nature. In any case, the peculiarities of the respective models in both [171] and [106] make it improbable an adaptation to the problem of vicariously learner ranking.

5.10 Chapter review

We addressed the previously unexplored problem of ranking lurkers in an OSN. We introduced a topology-driven lurking definition that rely on three basic principles to model lurking in a network, namely overconsumption, authoritativeness of the information received, and non-authoritativeness of the information produced. We proposed various lurker ranking models, for which we provided a complete specification in terms of the well-known PageRank and alpha-centrality. We then expanded this definitions to include the time dimension. We first focused on the development of measures related to freshness and activity trend, both for individual users and for interactions between users. Such measures were used as key elements in a time-aware weighting scheme on which our proposed time-static and time-evolving lurker ranking methods rely. We conducted a rigorous analysis focusing on a number of temporal aspects related to lurking behavior. These include a comparison between lurkers and passive/inactive users, a study on preferential attachment between lurkers and active users, an analysis of the lurkers' responsiveness to others' actions, and a cluster analysis of the lurking trends over time. We have been

positively impressed by results achieved on a number of real-world networks by some of our lurker ranking methods, especially in terms of significance and higher meaningfulness with respect to other competing methods. We investigated the possibility to apply the proposed lurker ranking model to domains different than the classic OSNs, taking into account scenarios of social trust and research collaboration networks.

We believe there are still several open issues in the understanding of dynamics of lurkers in SNs, which at least include further enhancing the time-aware lurker ranking methods. Understanding latent interactions among users would offer a great potential [80] for mining lurkers as well. Another challenge would be exploring the relations between lurkers and other actors in a SN, including newcomers and spammers.

Given the inherent complexity of lurking, we believe there are still several open issues in the understanding of dynamics of lurkers in SNs. Some of the most challenging issues for research in this context are discussed next.

Context-biased lurking

Starting as visitors and newcomers, members of a community naturally evolve over time playing different roles (cf. Section 5.7), thus showing a stronger or weaker tendency toward lurking on different times. Moreover, the user's engagement level in the community clearly depends also on the number and type of contexts in which the user is involved. A topic-modeling-based network analysis could advance research on how participating in different contexts influences the lurking level of a user.

Boundary-spanning and cross-network lurking

Some of the members that lay on the boundary of a component may bridge over other components. In Section 5.6.6, we have found out that indeed relations may exist between lurkers and users that act as bridges over different components of an OSN graph. To a larger extent, and given the increased interest towards cross-network services (see the latest examples of YouTube and GooglePlus), members who lurk inside an OSN may not lurk, or even take on the role of experts, in other OSNs. An analysis of the lurker ranking problem across different OSNs would represent a great potential to get a more complete picture of their users.

6

Conclusion and Future Research

Concluding remarks. This thesis has focused on novel ranking problems in information networks. We dealt with information networks that belong to different application domains, specifically: semantic networks, package recommendation networks, online social networks, research collaboration networks and social trust networks. For each domain, we defined novel approaches and methods in order to face ranking and analysis problems of different nature. A common denominator to all the proposed approaches is the focus on the development of random walk, eigenvector-based methods, which in part resembles the renowned Google's PageRank algorithm or related algorithms (e.g., alpha-centrality).

We defined a multi-relational PageRank model (MSSPR) for structural sense ranking, also focusing on the modeling of a heterogeneous information network consisting of a semantic multidigraph as ranking context graph (cf. Chapter 3). We dealt with package recommendation networks (cf. Chapter 4), for which we developed an application-independent framework for the recommendation of packages of items, which is fully unsupervised and uses a PageRank-like algorithm to produce a personalized ranking of the packages in the network. Concerning the domain of online social networks, we defined a new family of methods, namely LurkerRank (cf. Chapter 5), which showed its uniqueness in effectively identifying and ranking lurkers in an online social network. Moreover, an adaptation of the LurkerRank methods was studied in the context of research collaboration networks, in order to address the vicarious learning problem, mining the advisor-advisee relations from an apprentice point of view. A comparative evaluation was conducted between LurkerRank and global trust algorithms such as TrustRank and Anti-TrustRank, making also a first step towards the integration of the two techniques, with the purpose of modeling the understanding of lurkers from a perspective of social trust computing.

Our choice to focus on the definition of eigenvector-centrality based algorithms turned out to be successful, since the theoretical robustness and flexibility of this class of algorithms allowed us to define innovative ranking

methods in each domain, which improved upon baselines and competitors, producing highly quality ranking solutions when dealing with different tasks. We also carried out in-depth studies on the nature of the different information networks we had to cope with, presenting structural and qualitative analysis which helped us to better understand the nature of interactions among informative nodes in different contexts.

Future research. Concerning future research, there are plenty of open questions and challenges regarding the study of ranking problems in information networks. Some opportunities and necessities derive from the advent of *big data*: due to the increasing technological capabilities in terms of information transferring and storage, the great majority of (also personal) information regarding people all over the world is digitalized, giving birth to information networks of huge size and highly interconnected databases. Dealing with information networks of size in the order of (at least) billions of nodes with classic link-analysis algorithms can lead to efficiency and computational complexity issues. Developing effective approximated versions of the proposed ranking algorithms is hence challenging, since the aim is to define algorithms which can produce meaningfully approximated rankings of the nodes in this big-sized networks (or at least of a subset of interest) in reasonable time and with the use of a reasonable amount of resources.

Another interesting research topic is to bring the expressive power of heterogeneous information network models, which we successfully exploited in the field of semantic networks in Chapter 3, to other domains such as online social networks and recommendation networks. A complex nework model that has not been widely studied may be built around nodes that are not only users or endogeneous, within-network pieces of information (e.g., tags, microblogs, travel packages), but also exogeneous, *side information* available from the (off-line) users' real life (e.g., job, habits, visited places, real-life interactions) or from the membership of users to other online networks. This would enable to unveil how the understanding of manifold relations underlying the entities of interest in such complex networks can support and improve mining tasks in social networks and other information network domains.

Statistical topic modeling is an essential analysis tool that can obviously be helpful in many ways in our research lines as well: for instance, topic models might be integrated in the proposed frameworks in order to discover hidden communities among users in a social network, or latent relations among labels in a tree-data or to better understand different trends in the descriptions of travel destinations. Besides content analysis, the study of evolving networks is a necessary frontier to be reached, due to the high dynamism of many kinds of information networks. In literature, the time dimension is often taken into account using discrete models, e.g., snapshots of the networks related to a certain time range. This approach can certainly be improved, devising a network model which can effectively represent the evolving nature of real dynamic complex systems.

Finally, particular attention should be given to the role of trust/distrust algorithms in the study of information networks. Due to the large amount of content and users that characterizes the Web (and information networks in general), being able to recognize to what extent a certain source/user is trustworthy is a crucial problem. Particular effort should be made in the definition of a model that is able to integrate global and local trust ranking methods, which would allow for overcoming the expressiveness limits of the two approaches in representing real complex interactions among the nodes in a network (e.g., interactions among nodes which are considered trustworthy in a global context, but have low local trust scores when considering direct relations among them).

References

1. Abdi, H.: The Kendall Rank Correlation Coefficient. In: Encyclopedia of Measurement and Statistics (2007)
2. Adali, S.: Modeling Trust Context in Networks. Springer Briefs in Computer Science. Springer (2013)
3. Adali, S., Escriva, R., Goldberg, M.K., Hayvanovych, M., Magdon-Ismail, M., Szymanski, B.K., Wallace, W.A., Williams, G.T.: Measuring behavioral trust in social networks. In: Proc. IEEE Int. Conf. on Intelligence and Security Informatics, pp. 150–152 (2010)
4. Agirre, E., de Lacalle, O.L., Fellbaum, C., Hsieh, S., Tesconi, M., Monachini, M., Vossen, P., Segers, R.: SemEval-2010 Task 17: All-words Word Sense Disambiguation on a Specific Domain. In: Proc. of the 5th Int. Workshop on Semantic Evaluation, SemEval '10, pp. 75–80 (2010)
5. Agirre, E., Soroa, A.: Using the Multilingual Central Repository for Graph-Based Word Sense Disambiguation. In: Proc. Int. Conf. on Language Resources and Evaluation (LREC), pp. 1388–1392 (2008)
6. Agirre, E., Soroa, A.: Personalizing PageRank for Word Sense Disambiguation. In: Proc. 12th Conf. of the European Chapter of the Association for Computational Linguistics (EACL), pp. 33–41 (2009)
7. Ahn, Y.Y., Han, S., Kwak, H., Moon, S.B., Jeong, H.: Analysis of topological characteristics of huge online social networking services. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 835–844 (2007)
8. Al-Oufi, S., Kim, H., El-Saddik, A.: A group trust metric for identifying people of trust in online social networks. Expert Systems with Applications **39**(18), 13,173–13,181 (2012)
9. Anand, S., Chandramouli, R., Subbalakshmi, K.P., Venkataraman, M.: Altruism in social networks: good guys do finish first. Social Netw. Analys. Mining **3**(2), 167–177 (2013)
10. Angel, A., Chaudhuri, S., Das, G., Koudas, N.: Ranking objects based on relationships and fixed associations. In: Proc. Int. Conf. on Extending Database Technology (EDBT), pp. 910–921 (2009)
11. Ardichvili, A.: Learning and knowledge sharing in virtual communities of practice: motivators, barriers, and enablers. Advances in Developing Human Resources **10**, 541–554 (2008)

12. Avrachenkov, K., Litvak, N., Nemirovsky, D., Osipova, N.: Monte Carlo Methods in PageRank Computation: When One Iteration is Sufficient. SIAM Journal on Numerical Analysis **45**(2), 890–904 (2007)
13. Bakhshandeh, R., Samadi, M., Azimifar, Z., Schaeffer, J.: Degrees of Separation in Social Networks. In: Proc. Symp. on Combinatorial Search (SOCS), pp. 18–23 (2011)
14. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on Twitter. In: Proc. ACM Conf. on Web Search and Web Data Mining (WSDM), pp. 65–74 (2011)
15. Balmin, A., Hristidis, V., Papakonstantinou, Y.: ObjectRank: Authority-Based Keyword Search in Databases. In: Proc. Int. Conf. on Very Large Data Bases (VLDB), pp. 564–575 (2004)
16. Bandura, A.: Social foundations of thought and action: A social cognitive theory. Englewood Cliffs, NJ: Prentice Hall (1986)
17. Baraglia, R., Frattari, C., Muntean, C.I., Nardini, F.M., Silvestri, F.: A Trajectory-Based Recommender System for Tourism. In: Proc. Int. Conf on Active Media Technology (AMT), pp. 196–205 (2012)
18. Benevenuto, F., Rodrigues, T., Cha, M., Almeida, V.A.F.: Characterizing user behavior in online social networks. In: Proc. ACM SIGCOMM Conf. on Internet Measurement (IMC), pp. 49–62 (2009)
19. Berberich, K., Vazirgiannis, M., Weikum, G.: Time-Aware Authority Ranking. Internet Mathematics **2**(3), 301–332 (2005)
20. Bharat, K., Henzinger, M.R.: Improved algorithms for topic distillation in hyperlinked environments. In: Proc. of the ACM Conf. on Research and Development in Information Retrieval (SIGIR), pp. 104–111 (1998)
21. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: Structure and dynamics . Physics Reports **424**(4âĂŞ5), 175 – 308 (2006)
22. Bonacich, P.: Factoring and weighing approaches to status scores and clique identification. Journal of Mathematical Sociology **2**, 113–120 (1972)
23. Bonacich, P., Lloyd, P.: Eigenvector-like measures of centrality for asymmetric relations. Social Networks **23**, 191–201 (2001)
24. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Finding authorities and hubs from link structures on the World Wide Web. In: Proc. of the ACM Conf. on World Wide Web (WWW), pp. 415–429 (2001)
25. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology **25**(2), 163–177 (2001)
26. Brandes, U.: On variants of shortest-path betweenness centrality and their generic computation. Social Networks **30**(2), 136–145 (2008)
27. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems **30**(1-7), 107–117 (1998)
28. Brodsky, A., Henshaw, S.M., Whittle, J.: CARD: a decision-guidance framework and application for recommending composite alternatives. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp. 171–178 (2008)
29. Buckley, C., Voorhees, E.M.: Retrieval evaluation with incomplete information. In: Proc. ACM Conf. on Research and Development in Information Retrieval (SIGIR), pp. 25–32 (2004)
30. Budak, C., Agrawal, D., El Abbadi, A.: Structural trend analysis for online social networks. Proceedings of the VLDB Endowment **4**(10), 646–656 (2011)

31. Budalakoti, S., Bekkerman, R.: Bimodal invitation-navigation fair bets model for authority identification in a social network. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 709–718 (2012)
32. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Comput. Ling. **32**(1), 13–47 (2006)
33. Capocci, A., Servedio, V.D.P., Colaiori, F., Buriol, L.S., Donato, D., Leonardi, S., Caldarelli, G.: Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia. Phys. Rev. E **74** (2006)
34. Castillo, C., Mendoza, M., Poblete, B.: Information credibility on twitter. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 675–684 (2011)
35. Celli, F., Lascio, F.M.L.D., Magnani, M., Pacelli, B., Rossi, L.: Social Network Data and Practices: The Case of Friendfeed. In: Proc. Int. Conf. on Social Computing, Behavioral Modeling, and Prediction (SBP), pp. 346–353 (2010)
36. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring User Influence in Twitter: The Million Follower Fallacy. In: Proc. AAAI Conf. on Weblogs and Social Media (ICWSM) (2010)
37. Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., Kleinberg, J.M.: Automatic resource compilation by analyzing hyperlink structure and associated text. In: Proc. of the ACM Conf. on World Wide Web (WWW), pp. 65–74 (1998)
38. Chen, F.C., Chang, H.M.: Do lurking learners contribute less?: a knowledge co-construction perspective. In: Proc. Conf. on Communities and Technologies (C&T), pp. 169–178 (2011)
39. Choudhury, M.D., Feldman, M., Amer-Yahia, S., Golbandi, N., Lempel, R., Yu, C.: Automatic construction of travel itineraries using social breadcrumbs. In: Proc. ACM Conf. on Hypertext and Hypermedia (HT), pp. 35–44 (2010)
40. Chun, H., Kwak, H., Eom, Y.H., Ahn, Y.Y., Moon, S.B., Jeong, H.: Comparison of online social relations in volume vs interaction: a case study of Cyworld. In: Proc. ACM SIGCOMM Conf. on Internet Measurement (IMC), pp. 57–70 (2008)
41. Cranefield, J., Yoong, P., Huff, S.L.: Beyond Lurking: The Invisible Follower-Feeder In An Online Community Ecosystem. In: Proc. Pacific Asia Conf. on Information Systems (PACIS), p. 50 (2011)
42. Davis, D.A., Lichtenwalter, R., Chawla, N.V.: Multi-relational link prediction in heterogeneous information networks. In: Proc. Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM), pp. 281–288 (2011)
43. De Cao, D., Basili, R., Luciani, M., Mesiano, F., Rossi, R.: Robust and Efficient PageRank for Word Sense Disambiguation. In: Proc. Workshop on Graph-based Methods for Natural Language Processing (2010)
44. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by Latent Semantic Analysis. JASIS **41**(6), 391–407 (1990)
45. Deng, H., Han, J., Lyu, M.R., King, I.: Modeling and exploiting heterogeneous bibliographic networks for expertise ranking. In: Proc. Int. Joint Conf. on Digital Libraries (JCDL), pp. 71–80 (2012)
46. Deng, T., Fan, W., Geerts, F.: On the complexity of package recommendation problems. In: Proc. ACM Symposium on Principles of Database Systems (PODS), pp. 261–272 (2012)
47. Dhungel, P., Wu, D., Schonhorst, B., Ross, K.W.: A measurement study of attacks on BitTorrent leechers. In: Proc. Conf. on Peer-to-peer systems (IPTPS), p. 7 (2008)

48. Ding, Y., Yan, E., Frazho, A.R., Caverlee, J.: Pagerank for ranking authors in co-citation networks. Journal of the American Society for Information Science and Technology **60**(11), 2229–2243 (2009)
49. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press, New York, NY, USA (2010)
50. Edelmann, N.: Reviewing the definitions of "lurkers" and some implications for online research. Cyberpsychology, Behavior, and Social Networking **16**(9), 645–649 (2013)
51. Erkan, G., Radev, D.R.: LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. J. Artif. Intell. Res. (JAIR) **22**, 457–479 (2004)
52. Fagin, R., Kumar, R., Sivakumar, D.: Comparing Top k Lists. SIAM Journal on Discrete Mathematics **17**(1), 134–160 (2003)
53. Farzan, R., DiMicco, J.M., Brownholtz, B.: Mobilizing Lurkers with a Targeted Task. In: Proc. AAAI Conf. on Weblogs and Social Media (ICWSM), pp. 235–238 (2010)
54. Fazeen, M., Dantu, R., Guturu, P.: Identification of leaders, lurkers, associates and spammers in a social network: context-dependent and context-independent approaches. Social Netw. Analys. Mining **1**(3), 241–254 (2011)
55. Ferrara, E., Interdonato, R., Tagarelli, A.: Online Popularity and Topical Interests through the lens of Instagram. In: Submitted to ACM Hypertext and Social Media Conf. (2014)
56. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**, 35–41 (1977)
57. Freeman, L.C.: Centrality in social networks conceptual clarification. Social Networks **1**(3), 215–239 (1979)
58. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In: Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 1606–1611 (2007)
59. Gayo-Avello, D.: Nepotistic relationships in Twitter and their impact on rank prestige algorithms. Inf. Process. Manage. **49**(6), 1250–1280 (2013)
60. Ghosh, S., Viswanath, B., Kooti, F., Sharma, N.K., Korlam, G., Benevenuto, F., Ganguly, N., Gummadi, P.K.: Understanding and combating link farming in the Twitter social network. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 61–70 (2012)
61. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. (PNAS) **99**(12), 7821–7826 (2002)
62. Golbeck, J.: Computing and Applying Trust in Web-based Social Networks. Ph.D. thesis, College Park, MD, USA (2005)
63. Gollapalli, S.D., Mitra, P., Giles, C.L.: Ranking authors in digital libraries. In: Proc. Int. Joint Conf. on Digital Libraries (JCDL), pp. 251–254 (2011)
64. Gori, M., Pucci, A.: ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In: Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 2766–2771 (2007)
65. Gracia, J., Mena, E.: Web-Based Measure of Semantic Relatedness. In: Proc. Int. Conf. on Web Information Systems Engineering (WISE), pp. 136–150 (2008)
66. Graham, F.C., Tsiatas, A., Xu, W.: Dirichlet PageRank and Ranking Algorithms Based on Trust and Distrust. Internet Mathematics **9**(1), 113–134 (2013)

67. Granovetter, M.: The Strength of Weak Ties. American Journal of Sociology **78**(6), 1360–1380 (1973)
68. Guha, R.V., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 403–412 (2004)
69. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.O.: Combating Web Spam with TrustRank. In: Proc. Int. Conf. on Very Large Data Bases (VLDB), pp. 576–587 (2004)
70. Halfaker, A., Keyes, O., Taraborelli, D.: Making peripheral participation legitimate: reader engagement experiments in Wikipedia. In: Proc. ACM Conf. on Computer Supported Cooperative Work (CSCW), pp. 849–860 (2013)
71. Hamdi, S., Bouzeghoub, A., Gançarski, A.L., Yahia, S.B.: Trust Inference Computation for Online Social Networks. In: Proc. Int. Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 210–217 (2013)
72. Haveliwala, T.H.: Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. IEEE Trans. Knowl. Data Eng. **15**(4), 784–796 (2003)
73. Helou, S.E., Salzmann, C., Gillet, D.: The 3A Personalized, Contextual and Relation-based Recommender System. J. UCS **16**(16), 2179–2195 (2010)
74. Helou, S.E., Salzmann, C., Sire, S., Gillet, D.: The 3A contextual ranking system: simultaneously recommending actors, assets, and group activities. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp. 373–376 (2009)
75. Interdonato, R., Tagarelli, A.: Multi-relational PageRank for Tree Structure Sense Ranking. In: Proc. Int. Conf. on Web Information Systems Engineering (WISE) - Part I, pp. 306–319 (2013)
76. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Information Systems **20**(4), 422–446 (2002)
77. Jeong, H., Néda, Z., Barabási, A.L.: Measuring preferential attachment in evolving networks. EPL (Europhysics Letters) **61**(4), 567 (2003)
78. Ji, M., Han, J., Danilevsky, M.: Ranking-based classification of heterogeneous information networks. In: Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 1298–1306 (2011)
79. Ji, M., Sun, Y., Danilevsky, M., Han, J., Gao, J.: Graph Regularized Transductive Classification on Heterogeneous Information Networks. In: Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD), pp. 570–586 (2010)
80. Jiang, J., Wilson, C., Wang, X., Sha, W., Huang, P., Dai, Y., Zhao, B.Y.: Understanding latent interactions in online social networks. ACM Trans. on the Web **7**(4), 18:1–18:39 (2013)
81. Jiang, W., Wang, G., Wu, J.: Generating trusted graphs for trust evaluation in online social networks. Future Generation Comp. Syst. **31**, 48–58 (2014)
82. Kahnwald, N., KÃűhler, T.: Microlearning in virtual communities of practice? an explorative analysis of changing information behaviour. In: Proc. Microlearning Conf., pp. 157–172 (2006)
83. Kappler, K.E., de Querol, R.R.: Is there anybody out there? – social media as a new social fetish. In: Proc. ACM Web Science Conf. (WebSci) (2011)
84. Kashyap, A., Amini, R., Hristidis, V.: SonetRank: leveraging social networks to personalize search. In: Proc. ACM Conf. on Information and Knowledge Management (CIKM), pp. 2045–2049 (2012)
85. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (1953)

86. de Kerchove, C., Dooren, P.V.: The PageTrust Algorithm: How to rank web pages when negative links are allowed? In: Proc. SIAM Int. Conf. on Data Mining (SDM), pp. 346–352 (2008)
87. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 668–677 (1998)
88. Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment. J. ACM **46**(5), 604–632 (1999)
89. Kollock, P., Smith, M.: Managing the virtual commons: Cooperation and conflict in computer communities. Computer-Mediated Communication: Linguistic, Social, and Cross-Cultural Perspectives pp. 109–128 (1996)
90. Koutrika, G., Bercovitz, B., Garcia-Molina, H.: FlexRecs: expressing and combining flexible recommendations. In: Proc. ACM Int. Conf. on Management of Data (SIGMOD), pp. 745–758 (2009)
91. Krishnan, V., Raj, R.: Web Spam Detection with Anti-Trust Rank. In: Proc. Int. Workshop on Adversarial Information Retrieval on the Web (AIRWeb), pp. 37–40 (2006)
92. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 611–617 (2006)
93. Kwak, H., Lee, C., Park, H., Moon, S.B.: What is Twitter, a social network or a news media? In: Proc. ACM Conf. on World Wide Web (WWW), pp. 591–600 (2010)
94. Lang, J., Wu, S.F.: Social network user lifetime. Social Network Analysis and Mining **April** (2012)
95. Langville, A.N., Meyer, C.D.: Deeper inside PageRank. Internet Mathematics **1**(3), 335–400 (2005)
96. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Machine Learning **81**(1), 53–67 (2010)
97. Lave, J., Wenger, E.: Situated Learning: Legitimate Peripheral Participation. Cambridge University Press (1991)
98. Lee, S., Song, S., Kahng, M., Lee, D., Lee, S.: Random walk based entity ranking on graph for multidimensional recommendation. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp. 93–100 (2011)
99. Lehmann, J., Gonçalves, B., Ramasco, J.J., Cattuto, C.: Dynamical classes of collective attention in Twitter. In: Proc. 21th International Conference on World Wide Web, pp. 251–260 (2012)
100. Lempel, R., Moran, S.: The stochastic approach for link-structure analysis (SALSA) and the TKC effect. Computer Networks **33**(1-6), 387–401 (2000)
101. Leskovec, J., Backstrom, L., Kleinberg, J.: Meme-tracking and the dynamics of the news cycle. In: Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 497–506. ACM (2009)
102. Leskovec, J., Horvitz, E.: Planetary-scale views on a large instant-messaging network. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 915–924 (2008)
103. Leskovec, J., Huttenlocher, D.P., Kleinberg, J.M.: Predicting positive and negative links in online social networks. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 641–650 (2010)
104. Li, L., Xu, G., Zhang, Y., Kitsuregawa, M.: Random walk based rank aggregation to improving web search. Knowl.-Based Syst. **24**(7), 943–951 (2011)

105. Li, R., Wang, S., Deng, H., Wang, R., Chang, K.C.C.: Towards social user profiling: unified and discriminative influence model for inferring home locations. In: Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 1023–1031 (2012)
106. Li, X., Foo, C.S., Tew, K.L., Ng, S.K.: Searching for Rising Stars in Bibliography Networks. In: Proc. Int. Conf. on Database Systems for Advanced Applications (DASFAA), pp. 288–292 (2009)
107. Liu, H., Lim, E., Lauw, H.W., Le, M., Sun, A., Srivastava, J., Kim, Y.A.: Predicting trusts among users of online communities: an epinions case study. In: Proc. ACM Conference on Electronic Commerce (EC), pp. 310–319 (2008)
108. Liu, N.N., Yang, Q.: EigenRank: a ranking-oriented approach to collaborative filtering. In: Proc. ACM Conf. on Research and Development in Information Retrieval (SIGIR), pp. 83–90 (2008)
109. Liu, Q., Chen, E., Xiong, H., Ge, Y., Li, Z., Wu, X.: A Cocktail Approach for Travel Package Recommendation. IEEE Trans. Knowl. Data Eng. (PrePrints, doi.ieeecomputersociety.org/10.1109/TKDE.2012.233) (2012)
110. Liu, X., Bollen, J., Nelson, M.L., de Sompel, H.V.: Co-authorship networks in the digital library research community. Information Processing and Management **41**, 1462–1480 (2005)
111. Ma, N., Guan, J., Zhao, Y.: Bringing PageRank to the citation analysis. Information Processing and Management **44**(2), 800–810 (2008)
112. Makris, C., Plegas, Y., Stamou, S.: Web Query Disambiguation Using PageRank. J. American Society for Information Science and Technology (JAIST) **63**(8), 1581–1592 (2012)
113. Malliaros, F.D., Vazirgiannis, M.: To stay or not to stay: modeling engagement dynamics in social graphs. In: Proc. ACM Conf. on Information and Knowledge Management (CIKM), pp. 469–478 (2013)
114. Massa, P., Avesani, P.: Controversial Users Demand Local Trust Metrics: An Experimental Study on Epinions.com Community. In: Proc. AAAI Conf. on Artificial Intelligence (AAAI), pp. 121–126 (2005)
115. Massa, P., Avesani, P.: Trust-aware bootstrapping of recommender systems. In: Proc. ECAI Workshop on Recommender Systems, pp. 29–33 (2006)
116. McAuley, J.J., Leskovec, J.: Learning to Discover Social Circles in Ego Networks. In: Neural Information Processing Systems (NIPS), pp. 548–556 (2012)
117. McKnight, D.H., Chervany, N.L.: Trust and Distrust Definitions: One Bite at a Time. In: Trust in Cyber-societies, LNAI 2246, pp. 27–54 (2001)
118. Meo, P.D., Ferrara, E., Abel, F., Aroyo, L., Houben, G.J.: Analyzing user behavior across social sharing environments. ACM Transactions on Intelligent Systems and Technology **5**(1) (2013)
119. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM (2000)
120. Mihalcea, R., Tarau, P.: TextRank: Bringing Order into Text. In: Proc. European Conf. on Empirical Methods in Natural Language Processing (EMNLP), pp. 404–411 (2004)
121. Mihalcea, R., Tarau, P., Figa, E.: PageRank on Semantic Networks, with Application to Word Sense Disambiguation. In: Proc. 20th Int. Conf. on Computational Linguistics (COLING) (2004)
122. Minkov, E., Cohen, W.W.: Learning to rank typed graph walks: local and global approaches. In: Proc. of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web mining and social network analysis, pp. 1–8 (2007)

123. Minkov, E., Cohen, W.W.: Learning Graph Walk Based Similarity Measures for Parsed Text. In: Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP), pp. 907–916 (2008)
124. Mislove, A., Koppula, H.S., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Growth of the Flickr Social Network. In: Proc. ACM SIGCOMM Workshop on Social Networks (WOSN) (2008)
125. Mislove, A., Marcon, M., Gummadi, P.K., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proc. ACM SIGCOMM Conf. on Internet Measurement (IMC), pp. 29–42 (2007)
126. Muller, M.: Lurking as personal trait or situational disposition: lurking and contributing in enterprise social media. In: Proc. ACM Conf. on Computer Supported Cooperative Work (CSCW), pp. 253–256 (2012)
127. Nastase, V.: Unsupervised All-words Word Sense Disambiguation with Grammatical Dependencies. In: IJCNLP, pp. 757–762 (2008)
128. Newman, M.E.J.: Mixing patterns in networks. Physical Review E **67**(2), 026,126 (2003)
129. Nguyen, V., Lim, E.P., Jiang, J., Sun, A.: To Trust or Not to Trust? Predicting Online Trusts Using Trust Antecedent Framework. In: Proc. IEEE Int. Conf. on Data Mining (ICDM), pp. 896–901 (2009)
130. Nie, Z., Zhang, Y., Wen, J.R., Ma, W.Y.: Object-level ranking: bringing order to Web objects. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 567–574 (2005)
131. Nonnecke, B., Preece, J.J.: Lurker demographics: counting the silent. In: Proc. ACM Conf. on Human Factors in Computing Systems (CHI), pp. 73–80 (2000)
132. Onnela, J.P., Saramaki, J., Hyvonen, J., Szabó, G., de Menezes, M.A., Kaski, K., Barabási, A.L., Kertész, J.: Analysis of a large-scale weighted network of one-to-one human communication. New J. Phys. **9**(179) (2007)
133. O'Reilly, T.: What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Communications and Strategy **65**(1) (2007)
134. Ortega, F.J., Troyano, J.A., Cruz, F.L., Vallejo, C.G., Enríquez, F.: Propagation of trust and distrust for the detection of trolls in a social network. Computer Networks **56**(12), 2884–2895 (2012)
135. Preece, J.J., Nonnecke, B., Andrews, D.: The top five reasons for lurking: improving community experiences for everyone. Computers in Human Behavior **20**(2), 201–223 (2004)
136. Pretto, L.: A theoretical analysis of PageRank. In: Proc. of the Int. Symposium on String Processing and Information Retrieval, pp. 131–144 (2002)
137. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proc. Natl. Acad. Sci. (PNAS) **101**(9), 2658–2663 (2004)
138. Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press (2011)
139. Ramage, D., Rafferty, A.N., Manning, C.D.: Random Walks for Text Semantic Similarity. In: Proc. ACL Workshop on Graph-based Methods for Natural Language Processing, pp. 23–31 (2009)
140. Sabidussi, G.: The centrality index of a graph. Psychometrika **31**, 581–603 (1966)
141. Schneider, A., von Krogh, G., Jager, P.: "What's coming next?" Epistemic curiosity and lurking behavior in online communities. Computers in Human Behavior **29**, 293–303 (2013)

142. Schneider, F., Feldmann, A., Krishnamurthy, B., Willinger, W.: Understanding online social network usage from a network perspective. In: Proc. ACM SIGCOMM Conf. on Internet Measurement (IMC), pp. 35–48 (2009)
143. Schwämmle, V., Jensen, O.N.: A simple and fast method to determine the parameters for fuzzy c-means cluster analysis. Bioinformatics **26**(22), 2841–2848 (2010)
144. Seeley, J.R.: The net of reciprocal influence: a problem in treating sociometric data. Canadian Journal of Psychology **3**, 234–240 (1949)
145. Sharma, M., Urs, S.R.: Network of Scholarship: Uncovering the Structure of Digital Library Author Community. In: Proc. Int. Conf. on Asian Digital Libraries (ICADL), pp. 363–366 (2008)
146. Sinha, R., Mihalcea, R.: Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity. In: Proc. IEEE Int. Conf. on Semantic Computing (ICSC), pp. 363–369 (2007)
147. Sinha, R., Mihalcea, R.: Using Centrality Algorithms on Directed Graphs for Synonym Expansion. In: Proc. Int. Florida Artificial Intelligence Research Society Conf. (FLAIRS), pp. 311–316 (2011)
148. Soroka, V., Rafaeli, S.: Invisible participants: how cultural capital relates to lurking behavior. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 163–172 (2006)
149. Stephenson, K., Zelen, M.: Rethinking centrality: Methods and applications. Social Networks **11**, 1–37 (1989)
150. Sun, Y., Aggarwal, C.C., Han, J.: Relation Strength-Aware Clustering of Heterogeneous Information Networks with Incomplete Attributes. Proc. of the VLDB Endowment (PVLDB) **5**(5), 394–405 (2012)
151. Sun, Y., Barber, R., Gupta, M., Aggarwal, C.C., Han, J.: Co-author Relationship Prediction in Heterogeneous Bibliographic Networks. In: Proc. IEEE/ACM Conf. on Advances in Social Networks Analysis and Mining (ASONAM), pp. 121–128 (2011)
152. Sun, Y., Han, J.: Mining Heterogeneous Information Networks: Principles and Methodologies. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers (2012)
153. Sun, Y., Han, J., Aggarwal, C.C., Chawla, N.V.: When will it happen?: relationship prediction in heterogeneous information networks. In: Proc. ACM Conf. on Web Search and Web Data Mining (WSDM), pp. 663–672 (2012)
154. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. Proceedings of the VLDB Endowment (PVLDB) **4**(11), 992–1003 (2011)
155. Sun, Y., Han, J., Zhao, P., Yin, Z., Cheng, H., Wu, T.: RankClus: integrating clustering with ranking for heterogeneous information network analysis. In: Proc. Int. Conf. on Extending Database Technology (EDBT), pp. 565–576 (2009)
156. Sun, Y., Norick, B., Han, J., Yan, X., Yu, P.S., Yu, X.: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In: Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 1348–1356 (2012)
157. Sun, Y., Tang, J., Han, J., Gupta, M., Zhao, B.: Community Evolution Detection in Dynamic Heterogeneous Information Networks. In: Proc. of the Eighth Workshop on Mining and Learning with Graphs, MLG '10, pp. 137–146 (2010)

158. Sun, Y., Yu, Y., Han, J.: Ranking-based clustering of heterogeneous information networks with star network schema. In: Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 797–806 (2009)
159. Tagarelli, A.: Exploring Dictionary-based Semantic Relatedness in Labeled Tree Data. Information Sciences **220**, 244–268 (2013)
160. Tagarelli, A., Gullo, F.: Evaluating PageRank methods for structural sense ranking in labeled tree data. In: Proc. 2nd Int. Conf. on Web Intelligence, Mining and Semantics (WIMS) (36, 2012)
161. Tagarelli, A., Interdonato, R.: Ranking Vicarious Learners in Research Collaboration Networks. In: S. Urs, J.C. Na, G. Buchanan (eds.) Digital Libraries: Social Media and Community Networks, *LNCS*, vol. 8279, pp. 93–102. Springer International Publishing (2013)
162. Tagarelli, A., Interdonato, R.: "Who's out there?": identifying and ranking lurkers in social networks. In: Proc. IEEE/ACM Conf. Advances in Social Networks Analysis and Mining (ASONAM), pp. 215–222 (2013)
163. Taherian, M., Amini, M., Jalili, R.: Trust Inference in Web-Based Social Networks Using Resistive Networks. In: Proc. Int. Conf. on Internet and Web Applications and Services (ICIW), pp. 233–238 (2008)
164. Tran, A., Bowes, C., Brown, D., Chen, P., Choly, M., Ding, W.: TreeMatch: A Fully Unsupervised WSD System Using Dependency Knowledge on a Specific Domain. In: Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval '10, pp. 396–401 (2010)
165. Tran, Q.T., Chan, C.Y., Wang, G.: Evaluation of set-based queries with aggregation constraints. In: Proc. ACM Conf. on Information and Knowledge Management (CIKM), pp. 1495–1504 (2011)
166. Tsatsaronis, G., Varlamis, I., Nørvrag, K.: SemanticRank: Ranking Keywords and Sentences Using Semantic Graphs. In: Proc. Int. Conf. on Computational Linguistics (COLING), pp. 1074–1082 (2010)
167. Tsatsaronis, G., Varlamis, I., Torge, S., Reimann, M., Nørvrag, K., Schroeder, M., Zschunke, M.: How to Become a Group Leader? or Modeling Author Types Based on Graph Mining. In: Proc. Int. Conf. on Theory and Practice of Digital Libraries (TPDL), pp. 15–26 (2011)
168. Varlamis, I., Eirinaki, M., Louta, M.D.: A Study on Social Network Metrics and Their Application in Trust Networks. In: Proc. IEEE/ACM Conf. on Advances in Social Networks Analysis and Mining (ASONAM), pp. 168–175 (2010)
169. Viswanath, B., Mislove, A., Cha, M., Gummadi, P.K.: On the evolution of user interaction in Facebook. In: Proc. ACM SIGCOMM Workshop on Social Networks (WOSN), pp. 37–42 (2009)
170. Walter, F.E., Battiston, S., Schweitzer, F.: Personalised and dynamic trust in social networks. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp. 197–204 (2009)
171. Wang, C., Han, J., Jia, Y., Tang, J., Zhang, D., Yu, Y., Guo, J.: Mining advisor-advisee relationships from research publication networks. In: Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 203–212 (2010)
172. Wang, Y., Lin, X., Zhang, Q.: Towards metric fusion on multi-view data: a cross-view based graph random walk approach. In: Proc. ACM Conf. on Information and Knowledge Management (CIKM), pp. 805–810 (2013)
173. Wasserman, S., Faust, K.: Social Networks Analysis: Methods and Applications. Cambridge University Press (1994)

174. Watts, D., Strogatz, S.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998)
175. Webber, W., Moffat, A., Zobel, J.: A similarity measure for indefinite rankings. ACM Trans. Information Systems **28**(4), 20 (2010)
176. Wilson, C., Sala, A., Puttaswamy, K.P.N., Zhao, B.Y.: Beyond Social Graphs: User Interactions in Online Social Networks and their Implications. ACM Trans. on the Web **6**(4), 17 (2012)
177. Wu, B., Goel, V., Davison, B.D.: Propagating Trust and Distrust to Demote Web Spam. In: Proceedings of the WWW06 Workshop on Models of Trust for the Web (MTW06) (2006)
178. Wu, J., Aberer, K.: Using a layered markov model for distributed web ranking computation. In: Proc. Int. Conf. on Distributed Computing Systems, pp. 533–542 (2005)
179. Xie, M., Lakshmanan, L.V.S., Wood, P.T.: Composite recommendations: from items to packages. Frontiers of Computer Science **6**(3), 264–277 (2012)
180. Xie, W., Li, C., Zhu, F., Lim, E.P., Gong, X.: When a friend in Twitter is a friend in life. In: Proc. ACM Web Science Conf. (WebSci), pp. 344–347 (2012)
181. Yan, E., Ding, Y.: Discovering author impact: A PageRank perspective. Inf. Process. Manage. **47**(1), 125–134 (2011)
182. Ye, J., Cheng, H., Zhu, Z., Chen, M.: Predicting positive and negative links in signed social networks by transfer learning. In: Proc. ACM Conf. on World Wide Web (WWW), pp. 1477–1488 (2013)
183. Zhang, M., Feng, S., Tang, J., Ojokoh, B.A., Liu, G.: Co-Ranking Multiple Entities in a Heterogeneous Network: Integrating Temporal Factor and Users' Bookmarks. In: Proc. Int. Conf. on Asian Digital Libraries (ICADL), pp. 202–211 (2011)
184. Zhang, X., Liang, W., Zhu, S., Han, B.: Automatic seed set expansion for trust propagation based anti-spam algorithms. Inf. Sci. **232**, 167–187 (2013)