Analysis, prediction and privacy-aware design of social media and online search

Luca Maria Aiello
Computer Science Department
University of Torino

A dissertation submitted to the University of Torino
for the degree of Doctor of Philosophy

Year 2011/2012

Abstract

The evolution of the Web into a collaborative and social platform where identities, resources, and services are sewn together by networks of people have revolutionized the way to communicate and search through the Internet. Information and ideas are increasingly originated by the "long tail" of Web users and are capillary spread through the online connections of the new-generation *social media*. Understanding and modeling the dynamics of information flows and of interpersonal interactions in online social media has become increasingly necessary to retrieve the relevant data from such information-overloaded environments and to offer effective search and personalization services. On the other hand, in a scenario where the quality of services increases with the amount of data provided by the users, serious privacy issues arise about the management, the use, and the diffusion of personal user data on the Web.

In this work we explore the orthogonal axes of services and privacy in social media and in search engines. Relying on data mining and complex systems analysis techniques we investigate the evolutionary dynamics of social and search systems with a focus on similarity patterns between users, correlations between activity indicators, homophily and social influence. Empirical findings from data-driven analysis allowed us to develop accurate *prediction* and *recommendation* techniques. In particular, we propose and discuss several flavors of link recommendation algorithms for social networks and we put the basis for the prediction of future user activity in conventional search engines. A new scheme of social search engine is also proposed to tackle the information overload in search from a collaborative perspective. Besides, we explore the dimension of privacy by surveying the hazards of personal data exploitation and leakage and proposing an architecture to support privacy-aware social networks and applications. We rely on a decentralized paradigm that effectively tackles privacy issues with strong network-level security devices and flexible access control mechanisms. Furthermore, we show that typical applications widely used in centralized social systems and search engines can be efficiently implemented and maintained on our decentralized platform.

Contents

Preamble

This work presents the original research results of a three-year PhD program in Computer Science. The exposed findings belong to multidisciplinary research areas including analysis of complex online systems, collaborative and centralized search, peer-to-peer overlays, network security, and privacy-aware design of online social networks. The work is structured in three Parts.

Part I is focused on the analysis of information overloaded online systems such as **social networks** and **search engines**. Using techniques from complex systems analysis and data mining we discover hidden patterns in the data describing the **dynamics of evolution** of such socio-technological systems. A proper exploitation of the emerging patterns can help to win the information overload problem for both providers and consumers, thus fueling a virtuous circle where users participate more actively to get more high-quality personalized services back. One of the main themes of this part will be a study on the possibility to make **predictions** on the evolution of such complex systems at both macroscopic and microscopic scales.

More in detail, introductory Chapters 1 and 2 present the context on information overload in social media and search systems and outline the basic techniques of social networks and **complex systems analysis** that are used later in the thesis. Based on the data extracted from many social media different in size and scope, we analyze the patterns of the evolution of online networked environments (Chapter 3) and the phenomena of **homophily** and **influence** occurring between the individuals in the network (Chapter 4). The possibility of predicting the creation of new social links in a network is explored in Chapter 5. The context of information overload in search engines is discussed in Chapter 6, where we show the potential of a topical **user profiling** in predicting future search activity. Finally, the strength of **collaborative paradigm** in overcoming the information overload in search systems is discussed in Chapter 7.

Part II deals with the **privacy** problems emerging in online social media. The growing volume of data gathered by online social networks has led to a dramatic collision between private, public and commercial spheres that have become very strictly connected. To address the issues of user information exploitation and leakage we propose a **distributed architecture** for social networking which reaches a good trade-off between the often conflicting requirements of security, privacy, and quality of service.

In Chapter 8 we present the problem and the major former attempts to solve it. In Chapter 9 we overview the state of the art of purely decentralized **peer-to-peer systems** and of their security issues. Then we describe, in a bottom-up fashion, all the architectural components we introduce to support privacy-preserving social networks, namely a **secure DHT** (Chapter 10), a suite of privacy and **access control** services (Chapter 11), and a **collaborative search** application that can efficiently work on the top of the stack and provide all the functions that are typically available in a centralized context (Chapter 12).

Part III concludes by outlining some new perspectives and challenges in the analysis and exploitation of complex online systems. We discuss the increasing entanglement between real and online worlds and the reciprocal effects that they have one another. Also, we show the outcome of

an experiment on **popularity** and **influence** in social media that reveals an interesting perspective on the **usage patterns** of social networks and highlights hazards given by some vulnerabilities of online social tools. Since the obtained results are not dependent on the architectural type of the social platform in use (centralized or peer-to-peer), they help to outline new challenges that are intrinsic to the social substrate and go beyond the privacy issues related to the techniques of user data management.

The material composing this thesis is drawn in large part from original publications of the candidate. Part I includes the contributions of the following papers:

- Link creation and profile alignment in the aNobii social network (SocialCom 2010) [8]
- Friendship prediction and homophily in social media (TWEB 2011) [16]
- Tagging relations to achieve complex search goals (NWeSP 2011) [298]
- Behavior-driven clustering of queries into topics (CIKM 2011) [10]
- Dynamics of link creation and information spreading over social and communication networks (submitted to TIST) [9]

Part II reports the results of the following publications:

- Tempering Kademlia with a Robust Identity Based System (P2P 2008) [11]
- Avoiding eclipse attacks on Kad/Kademlia: an identity based approach (ICC 2009) [213]
- Tagging with DHARMA, a DHT-based Approach for Resource Mapping through Approximation (HOTP2P 2010) [12]
- Secure and Flexible Framework for Decentralized Social Network Services (SeSoc 2010) [14]
- An identity-based approach to secure P2P applications with Likir (P2P Applications 2011) [13]
- LotusNet: tunable privacy for distributed online social network services (ComCom 2012) [15]

Part I

Ruling information overload with online search and social systems

Chapter 1

Information overload in the social network era

> *Future shock is the shattering stress and disorientation that we induce in individuals by subjecting them to too much change in too short a time.*
>
> Alvin Toffler, writer and futurist (1970)

The *Future Shock* is the formula used by Alvin Toffler to define the upsetting effect that *information overload* has on human mind. Information overload affects the ability of a person to take rational decisions or adopt rational behavior when she is subject to a great amount of data or when she is "*plunged into a fast and irregularly changing situation, or a novelty-loaded context*" [325]. In other words, when the amount of available information grows too much, the decision-making capability of an individual is shattered and could bring to a state of behavioral paralysis (Figure 1.1).

The term, "information overload" popularized by Toffler in his extensive work discussing the impact of the digital revolution on social and personal life, was known even before the so-called Information Era. The traits of the information revolution brought by the World Wide Web and its implications on the explosion of the amount of data accessible by any person connected to the Internet went even far beyond Toffler's imagination.

We can detect three macro-steps in the history of the Web that determined radical changes in the way we collect information from it. First, at the end of the 90s, the exponential rise of the number of Web pages induced the need for increasingly sophisticated tools for indexing and search; online search engines experienced an explosive growth of usage in a few years [157] and they soon became the main access points for the retrieval of online information. The second phase came before the mid of 2000s, with the rise of the *Web 2.0* paradigm [253]. The Web quickly shifted from the publisher-consumer structure (few providers producing content for the crowds) to a collaborative paradigm where every user can actively participate the Web and become a content contributor, using new instruments like Wikipedia, microblogging, or collaborative tagging systems. Last, the

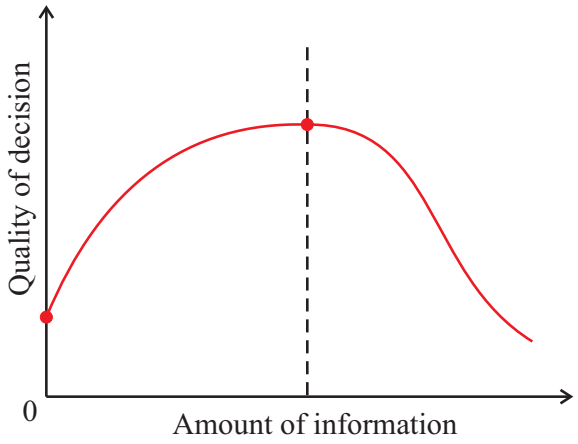


Figure 1.1: Pictorial representation of the information overload. The quality of a decision depends on the amount of the information available about the context in which the decision must be taken. Up to a certain threshold, richer information implies higher quality decisions. Above that threshold, the amount of information to be processed is too high for a human mind. As a consequence, every additional piece of information reduces the quality of the decision because the mind is forced to take mental shortcuts, thus risking over-generalization and an over-selection that often ends up in confirming prior points of view [309].

most recent stage of evolution of the Web is the transition between Web 2.0 and the so-called *Social Web*, that can be symbolically traced to the 2008 event of Facebook overtaking MySpace in terms of Internet traffic [19]. Since then, the growth of the social media has been explosive [145, 274] and social tools like Facebook, Twitter, LinkedIn, and Google+ (just to mention some of the major online social networks) have become the main means of interaction and information dissemination and acquisition for Web users [254, 166].

In a world where social experiences are augmented (when not replaced) by online social media, and every single user of the Web can generate, process and convey information, the problem of information overload becomes extremely difficult to manage. In fact, such countless data sources imply an extremely high rate of resource creation and an enormous redundancy in the publicly available information. By this time, information overload not only affects the individuals exposed to this plentiful stream of data, but neither entire organizations nor the processing capacity of the most advanced computer clusters are able to digest and synthesize the full corpus of data in some extremely information-overloaded domains like Web search and information exchange on online social media.

Such great data availability represents an opportunity that has never been experienced before. In first place, scientists can support their social interaction theories with data-driven analysis on extensive datasets. But mastering efficient techniques for data filtering, mining, and analysis has become a necessity also for service providers and, as a consequence, for their clients. Service providers need fast tools to understand the relevant information emerging from their user base so that they can predict some aspects of the evolution of the system and provide new services and,

possibly, make profit from them. On the other hand, consumers need support to filter data in a clever way to optimize their information gain.

There are two main paths can be followed to seize this opportunity. The first is the production of automatic data analysis techniques to unveil hidden patterns and model the dynamics of an information overloaded system. In this work we will extensively discuss and explore several networked systems that can be effectively analyzed through complex systems analysis techniques. The latter is to leverage the user mass to filter and redirect the information using the human computing power. This approach has the advantage that, in some contexts, human judgment can retrieve some relations between resources and facts that can be hardly recognized by automated procedures.

The main contribution of this Part is to show how information overload in complex web-based systems (e.g., social networks, social media, search portals, an so on) can be limited, channeled, and managed. We will describe how the general dynamics of a social system can be understood, modeled, and predicted using techniques from social network analysis, complex system analysis, data mining, and machine learning. We also show how the collaborative paradigm can be exploited to contrast information overload.

We want to show that there is a self-powered loop (or virtuous circle) between the amount of involvement of the user in a web service (in terms of data shared and human-computing power) and the services that the users can have back. In fact, service providers who have enough data are enabled to provide services like recommendation of new items or social contacts, and if there is enough participation of the users they can build innovative collaborative systems for smart information sharing and filtering.

More in detail, the structure of this Part is the following. First, in Chapter 2 we provide an overview on the techniques and tools for social network mining, and we report some of the major outcomes in complex systems analysis and modeling. In Chapter 3 we start a systematic study of the structure and dynamics of three different social media, discussing invariants and peculiarities of their evolutionary traces. In Chapter 4 we delve into the dynamics of link creation and profile alignment in the social network; in particular we find strong evidence of the presence of homophily-driven attachment and social influence and a reciprocal causality between the two phenomena. Based on these observations, in Chapter 5 we leverage the acquired knowledge on the evolutionary processes underlying the social graph to propose a contact recommender that is trained on both topological and user profile features. In Chapter 6 we shift our perspective to search systems and we tackle the problem of summarizing the corpus of queries generated by users in a more synthetic set of wider topics. We show how such summarized information can be used to profile users and to pose the basis for the modeling and prediction of their future activity. Finally, in Chapter 7 we deal with a very common kind of social search system (i.e., folksonomy) and we show how a proper model of collaborative knowledge management could dramatically expand its potential and effectiveness.

Chapter 2

Study of complex systems with network analysis

2.1 Pervasiveness of networks in real world

Historically, man has attempted to describe the world around him by finding regularities in what is perceivable. As a matter of fact, patterns of interaction between entities emerge clearly, at both macroscopic and microscopic scales, in most of the natural and artificial environments we are plunged in. Predator-prey relationship between living beings in natural ecosystems, the process of spreading of a virus in a population, the interactions that occur between two or more proteins to carry out a biological function, are all intelligible examples of *networked systems* as well as the map of the daily schedule of flights around the world, the social acquaintances of the citizens in a town or the connections between the routers of the Internet.

In general, every system that encapsulates a relation or interaction among some of its constituent elements admits some abstract representation in terms of a *network*, or, from a mathematical perspective, of a *graph*. A graph is formed by a set of *vertices* (or nodes) that identify the elements of the system, together with a set of *edges* representing some sort of relation between the vertices. Such very general theoretical framework can be applied to model a vast variety of *complex* systems. Even if it is hard to converge on a widely accepted notion of "complexity", a quite general definition has been given by Barrat et al. [41]:

> *"Complex systems consist of a large number of elements capable of interacting with each other and their environment in order to organize in specific emergent structures".*

Complex systems analysis is a relatively young discipline focused on the empirical study and systematic modeling of real-world networks with the goal of mining the information hidden by the complex patterns, understanding the underlying dynamics of interaction and evolution and learn how to improve the effectiveness and efficiency of artificial complex systems.

Probably, Social Network Analysis (SNA) has been one of the earliest disciplines that can be classified under the label of "complex systems analysis". Initially, SNA was a branch of sociology

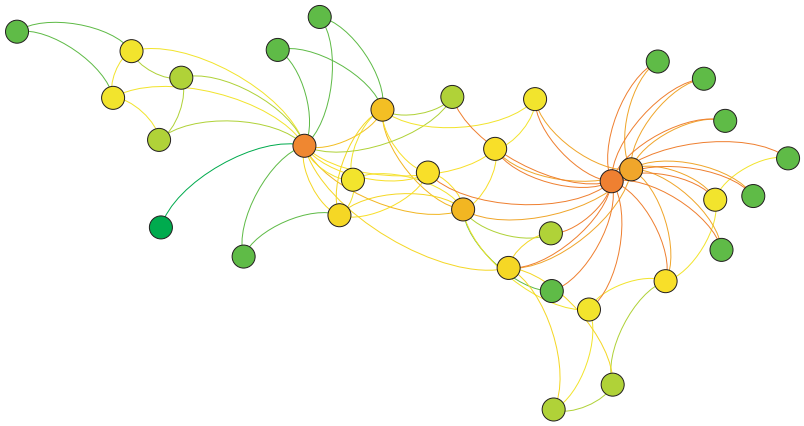


Figure 2.1: Zachary's karate club network [359], one of the most known "toy datasets" in social network analysis. It represents the social network of friendships between 34 members of a karate club in a US university in the 70s. Nodes are colored on a scale from green to red according to the number of their connections. Even in this tiny social network, complex patterns in the structure of the interpersonal relations are observable from the visualization. Picture made with the Gephi toolkit [42] (`gephi.org`).

(or, more precisely, an evolution of sociometry) that relied on notions of graph theory to understand the dynamics of social aggregations of people. Back to the earliest times of SNA, the techniques of data collection (mainly, surveys written off by the individuals under investigation) did not allow the analysis of huge networks. However, even very small environments may be driven by complex patterns (see Figure 2.1), and many techniques that are still widely used today in the analysis of huge datasets have been studied and developed from the analysis of small real-world cases [340].

More recently, the rapid scaling of Internet and especially of the Web in his collaborative facet, has provided a fresh source of social data that can be mined to describe the underlying patterns of big and very dynamic online social systems.

Such data allowed to test classic theories of SNA on a larger scale. For instance, the results of the notorious Milgram's experiment [226] on the six degrees of separation, according to which the number of hops that separates any two individuals in the graph of social acquaintances is surprisingly small (6 was the number estimated for US population), has been recently verified on the Microsoft Messenger [188] and Facebook [329]. Another example is the Dunbar theory that, based on the observation of the social dynamics of a group of primates and using a regression equation on their cortical size, claims that the "mean group size" that allow stable and profitable social relationships for humans is around 150 [99, 98]; quite surprisingly, the same theoretical cognitive limit has been verified in the Twitter follower network [129].

In addition to the study of virtual socio-systems, complex network analysis has been effectively applied to tackle social and humanitarian problems like the propagation of sexually transmitted diseases [201] and the proliferation of terrorist organizations [349]. In general, a graph-driven study of the so-called "dark networks" is useful to spot the weaknesses in the structure of undesirable aggregations of individuals to carry out targeted attacks able to disrupt the effectiveness of the

network with a minimal cost.

Structural analysis has been an area of great interest also on technological networks, like the Internet [106] or the Web. The rapid growth of such world-scale technological systems has raised for the first time (well before the viral diffusion of online social networks) the problem of dealing with the very high dimensionality of the data and the consequent difficulty in mining meaningful patterns. Foundational work on such systems has been made, for instance, in estimating the boundaries and the diameter of the World Wide Web [18] and in unveiling its shape that it has been shown to be organized in three main components (In, Out and Strongly Connected) surrounded by tendrils [65].

Around the turn of the millennium, the availability of datasets from many different sources and fields triggered a significant effort in the investigation of all kinds networked environments using the paradigm of the complex systems analysis. Such very interdisciplinary study allowed to discover very soon striking regularities in diverse networks. Several technological, biological or social self-organizing systems [242, 20] have been found to be characterized by the *small world* property [341], that determines a logarithmic growth of the average distance between pairs of nodes with respect to the total number of nodes. Many of these networks have been also found to be *scale-free*, namely characterized by a node connectivity distribution that decays as a power law [39, 69]. The discovery of these and many other invariants has given strength to the area of complex systems analysis as a cross-discipline science useful to detect and exploit ubiquitous hidden patterns in biological life and technological structures.

Since then, the perspective on the data-driven study of graph-based systems has widened considerably and the results of exploratory analysis of big networks [7, 160, 186], together with the increasingly richness of data on human activity, mobility, and interactions [297] opened the way to devise services that can profitably combine several data sources of interrelated complex systems to provide services to people. Besides the numbers of new recommendation and suggestion tools for online social networks, notable examples are the control of spreading of viruses in computer or human contact networks [261, 86], the monitoring of face-to-face interactions inside hospitals to minimize infections within the structure [158], the "outdoor advertising" on mobile phones [269] based on the mobility patterns of people in a city, and the spreading of information between mobile devices based on spontaneous affinities of users [300].

In this Part we use complex systems analysis to study social, concept, and similarity networks. In the following we provide some fundamental notions of graph theory and the descriptions of network analysis techniques and tools that are at the basis of our investigations.

2.2 Graph theory and social network analysis

2.2.1 Graph definition and basic terms

Graph theory is a branch of mathematics, whose origin can be traced back to Euler (see Figure 2.2), aimed at the study of structures formed by relations between objects. According to the definition

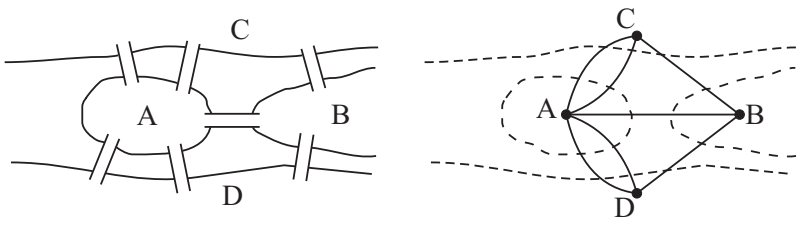


Figure 2.2: The problem of the Seven Bridges of Königsberg. Is there a walk through the city that crosses each bridge once and only once? Its negative solution was found by Leonhard Euler [103] in 1736 using an abstraction based on the notion of graph, where each land mass is reduced to a vertex and the bridges are the edges between them. Euler's proof laid the foundations of today's graph theory.

given by Bondy and Murthy [58], a graph is a triple $G = (V, E, \psi)$ composed by a set of *vertices* (or nodes) V, a set of *edges* (or links) E, and an incidence function ψ that associates a pair of nodes with each edge. Two nodes put in relation by an edge are commonly called *neighbors* and the set of neighbors of a specific node i can be denoted with $\Gamma(i)$. Graphs can be very intuitively drawn with circles to represent vertices and lines connecting the circles to represent the edges.

If the node pairs in the codomain of ψ are unordered (i.e., $\psi(e) = (i, j) \leftrightarrow \psi(e) = (j, i)$), then the graph is *undirected* (or *symmetric*), otherwise it is *directed*. In the directed case the edges can be also called *arcs*, the node from which the arc originates is called *predecessor* and the other one *successor*. The endpoints of an edge are said to be *incident* with the edge and are *adjacent* to each other.

An edge can connect nodes that are not necessarily distinct, so an edge can be a *loop* that originates and terminates in the same node. If more than one edge maps on the same pair of nodes, then the graph is a *multigraph*. On the contrary, a *simple* graph contains no loops and no multiple edges. When focusing on graphs with no multiple edges the definition of graph can be simplified to a pair $G = (V, E)$. Nodes and edges can be annotated with attributes, and specifically, a *weighted* graph has numeric weights associated to edges (or, more formally, exists a function $W : E \to \mathbb{R}$).

When the set of predecessors in the incidence function has a empty intersection with the set of successors, (i.e., $\psi(e) = (i, j) \to i \in V_1, j \in V_2, V_1 \cap V_2 = \oslash \wedge V_1 \cup V_2 = V, \forall e \in E$), then the graph is *bipartite*. Other examples of well-known structures are the *empty* graph ($V = \oslash, E = \oslash$), the *trivial* graph (with just one node), and the *complete* graph (where every possible pair of nodes is connected by one edge).

Similarly to sets, a graph includes *subgraphs* and set-like operations can be adapted to them. A subgraph $H = (E_H, V_H, \psi_H)$ of the graph $G = (E_G, V_G, \psi_G)$ is defined by the subset relations $V_H \subseteq V_G, E_H \subseteq E_G$. A subgraph of the graph G is *induced* by the node set $V' \subseteq V$ if it contains all the nodes in V' and all the edges of G between the nodes of V'; a edge-induced graph is determined by a set of edges and the nodes incident on them, instead.

A *walk* is a finite non-null sequence $W = v_0, e_0, ..., v_n, e_n$ whose terms are alternatively nodes

and edges, and such that, given $0 < i \leq n$, vertices v_{i-1} and v_i are adjacent. If all the nodes are distinct, then the walk is a *path*; if all the edges are distinct it is called *trail*. Two vertices are *connected* if a path between them exists in the graph; accordingly, the graph is connected if there is a path between any pairs of its nodes. The maximal connected subgraphs are called the *components* of the graph. Components containing just one node are commonly called *singletons*. When dealing with directed graphs, the definition of walk, path, and connected components can be adapted taking into account the directionality of edges. Connected components in a directed graph are called *strongly* connected components; *weakly* connected components, instead, are the components given by disregarding the directionality of arcs.

From a mathematical point of view, a graph can be defined in terms of a $|V| \times |V|$ adjacency matrix, where each entry (i, j) contains a binary value indicating the presence of a tie between nodes i and j or the weight of the connection in case of weighted graphs. The adjacency matrix is symmetric for undirected edges and the diagonal is undefined for simple graphs.

2.2.2 Topological measures on graphs

Complex network analysis is based on a core of topological measures that describe the main structural properties of the graph [340, 247]. In the following we review some of them.

Degree and strength.

The number of edges incident with a node is the *degree* of the node and it is usually denoted with the letter k. In case of directed networks, we can distinguish the in-degree k_{in} and out-degree k_{out}, respectively the number of edges for which the node is a successor or predecessor. For weighted networks, the sum of the weights of the edges incident on a node is the *strength* of the node; in-strength and out-strength are defined for weighted networks, similarly to in- and out-degree.

Density.

The ratio between the number of existing edges in a simple graph and the number of maximum edges is a measure of how densely the network is connected. Formally, the density is specified as:

$$D = \frac{2|E|}{|V|(|V|-1)}. \tag{2.1}$$

In directed networks density is divided by a factor 2 since the possible number of connections is double than the number in undirected graphs. Note that density is a measure that is strictly dependent by the size of the network. In real-world cases, the bigger is the network, the lower is the density. There is not a standard way to assess if a graph is *dense* or *sparse*. As a rule-of-thumb, a network is considered sparse when its average degree is much smaller than the number of nodes, $\langle k \rangle \ll |V|$, or when the orders of magnitude of the number of nodes and edges are approximatively the same, $|V| \sim |E|$.

Diameter.

The *distance* $\ell(i, j)$ between two nodes i, j in a graph is given by the shortest path between them. The maximum node distance among all the possible pairs of nodes is the *diameter* of the graph. Diameter is used to assess the width of the network; however, it is very susceptible to outliers, since just a single long shortest path can determine a high diameter. To avoid this

problem, the *effective diameter* measure have been proposed, that is the minimum distance such that it includes the 90% of the node pairs [257].

Clustering coefficient.

The structure of the neighborhood of a node reveals how much "local" nodes are clustered together. The *clustering* (or *transitivity* in social sciences) measures the tendency of the neighbors of a node to be connected to each other, thus forming a dense grid of triangles. In the context of human social networks, tendency to high clustering can be roughly summarized with the folk sentence "all the friends of my friends are my friends too". Quantitatively, clustering of a node i is measured by a coefficient $C(i)$ [341] computed as the ratio of the number of connections between i and its neighbors and the maximum number of such links. Let k_i be the degree of node i and e_i the number of edges between its neighbors; the clustering coefficient is then defined as:

$$C(i) = \frac{e_i}{k_i(k_i - 1)/2} \tag{2.2}$$

This measure is meaningful only for $k_i > 1$. If $k_i = 1$ then we consider $C(i) = 0$. The clustering coefficient of the network, which measures the overall degree of clustering of the graph, is simply defined as the average clustering:

$$\langle C(i) \rangle = \frac{1}{N} \sum_i C(i) \tag{2.3}$$

Centrality and centralization.

The role that specific nodes and edges have with respect to the global topology of the graph is one of the main insights to characterize the network. The importance of a node or edge is assessed by centrality measures [115], that may be defined on several structural features of the graph, like its connectivity or its position with respect to the other nodes.

The most commonly used centrality measure is the *degree centrality*, that coincides with the degree of the node.

$$C_D(i) = k_i. \tag{2.4}$$

The *closeness centrality* focuses instead on the distance of the target node to all the other vertices; centrality of a node i is defined as:

$$C_C(i) = \frac{1}{\sum_{j \neq i} \ell(i,j)}. \tag{2.5}$$

To account the importance of nodes that may not be well connected to the rest of the network but act as bridges between two or more components that are loosely tied to each other, the *betweenness centrality* is used:

$$C_B(i) = \sum_{h \neq j \neq i} \frac{\sigma_{hj}(i)}{\sigma_{hj}} \tag{2.6}$$

where σ_{hj} is the total number of shortest paths from node h to node j and $\sigma_{hj}(i)$ is the number of these shortest paths passing through i. The variation of the value of the centrality measures over all the nodes of the network (and with proper normalization factors) is called centralization, and provides a measure of how much the whole network is centralized. For instance, the degree centralization will be maximum in a star graph, where all the nodes are connected with a single

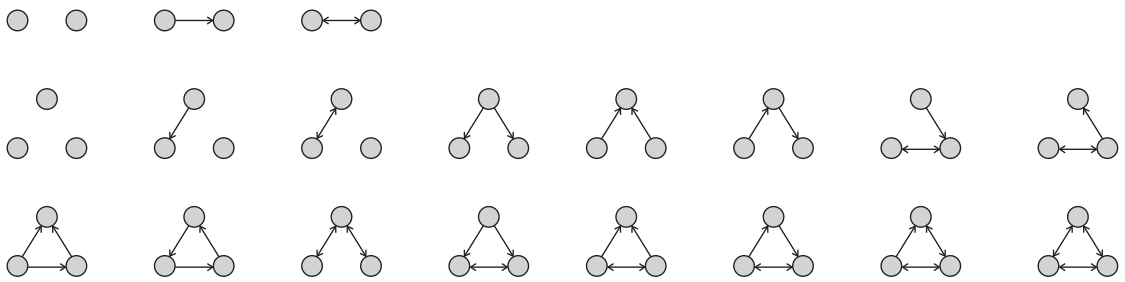


Figure 2.3: All the possible configurations of dyads and triads in a directed graph.

edge to a central node (and therefore the relative variation on the out-degree is maximum), and minimum in a ring graph, where each node is connected to two neighbors in a circle. Degree, closeness, and betweenness centralizations are defined respectively as:

$$C_D = \frac{\sum_i [C_D^{max} - C_D(i)]}{(|V|-1)(|V|-2)}, \tag{2.7}$$

$$C_C = \frac{\sum_i [C_C^{max} - C_C(i)]}{[(|V|-2])(|V|-1)]/(2|V|-3)}, \tag{2.8}$$

$$C_B = \frac{2\sum_i [C_B^{max} - C_B(i)]}{(|N|-1)^2(|N|-2)}, \tag{2.9}$$

where C_x^{max} denotes the maximum value of centralization in the graph.

Network motifs.

The analysis of small and well-identifiable structural components of the network helps to understand the meaning of the structure of the graph as a whole. Depending on the specific context, a configuration of edges can be mapped on some well-known behavior of the nodes. For example consider a directed network of email traffic, where $i \to j$ iff i has sent an email to j. In this case, the most trivial but yet meaningful pattern is a mutual tie $i \to j, i \leftarrow j$ that indicates some level of reciprocity that the individuals show in replying one to the other. This kind of analysis is indeed particularly useful in directed graphs, where much more configurations of connectivity are possible, and it is usually focused on sets of two or three nodes, namely *dyads* and *triads*.

There are three isomorphism classes for dyads ($i \to j$, $i \leftrightarrow j$, i and j disconnected) and 16 classes for triads (see Figure 2.3). More in general, configurations of n nodes are called *network motifs* [228]; studying complex motifs with $n > 3$ may make sense depending on what kind of natural phenomenon or technological structure the network is modeling. In most of the scenarios, the dyadic and triadic *census*, namely a statistic on the portion of each type of dyad and triad over all the possible instances of dyads and triads, can be useful to highlight what are the most prominent connectivity patterns in the graph.

Mixing patterns.

Structural and hierarchical ordering of many networked systems is often given by the tendency of individuals to be connected with others that share some feature in common. This pattern has been detected in datasets coming from different fields like human and computer sciences, ecology, and epidemiology and it is known as *assortative mixing*. Simmetrically, a *disassortative mixing* is detected in many other real world networks where individuals are connected more likely with others with different properties [243, 244, 249]. For instance, social webs are assortative with respect to

the age of the people and the network of routers in the Internet is disassortative with respect to the number of connections of each router.

Even if mixing patterns can be defined with respect to any attribute of the vertices, the most studied mixing pattern involves the node degree. This mixing measures the likelihood that nodes have neighbors with similar degree. One of the measures of correlation between degrees of neighbors is given by the Pearson assortativity coefficient [243]:

$$r = \frac{\sum_e j_e k_e/|E| - [\sum_e (j_e + k_e)/(2|E|)]^2}{[\sum_e (j_e^2 + k_e^2)/(2|E|)] - [\sum_e (j_e + k_e)/(2|E|)]^2} \tag{2.10}$$

where j_e and k_e denote the degree of the two nodes incident on edge e. Its values can range from -1 (perfectly disassortative network) up to 1 (perfectly assortative network). Nevertheless, Pearson coefficient can be misleading when a non-monotonous behavior of the correlation function is observed; in this case the measure gives more weight to the higher degree classes, which in several cases might not express correctly the variations of the correlation function behavior.

A more practical measure involves the average nearest neighbors degree of a vertex i:

$$k_{nn,i} = \frac{1}{k_i} \sum_{j \in \Gamma(i)} k_j. \tag{2.11}$$

Starting from this quantity for single nodes, we can define an average for all the nodes with the same degree class [260, 332]:

$$k_{nn}(k) = \frac{1}{N_k} \sum_{i|k_i=k} k_{nn,i}, \tag{2.12}$$

where N_k is the number of nodes with degree k. In the presence of correlations, $k_{nn}(k)$ identifies two classes of networks. If the value of $k_{nn}(k)$ increases with k, then highly connected nodes have a high probability of being linked with other high degree nodes, while a decreasing trend reveals a disassortative mixing.

2.3 Cohesive groups and clusters

A crucial problem in network analysis is the extraction of *clusters*, i.e., groups of nodes that share some common property or similar structural roles. In a graph context, a cluster (or *community*, or *module*) is intuitively a group of nodes tied by relatively dense, frequent or intense connections. To some extent, clusters can be considered as separate entities with their own autonomy, that can be often coherently explored independently of the graph as a whole.

In classical social network analysis, the concept of *cohesive subgroup* [340] is close to this intuitive notion. The simpler definition of cohesive subgroup is the *clique*, namely a subgraph with more than two vertices whose nodes are all adjacent (or, in other words, a complete subgraph). In this case the notion of cluster coincides with the notion of clique and the problem of finding the communities in a graph becomes equivalent to the problem of maximal cliques detection. The clique has well-specified mathematical properties and captures much of the intuitive notion of cohesive subgroup. Furthermore, cliques can overlap, thus allowing to define non-disjunct communities. However, the clique is the strictest definition of community and it is not suitable for many real-world networks.

The most intuitive relaxation of clique is the *k-plex*. The k-plex is a maximal subgraph containing n vertices in which each vertex is adjacent to at least $n - k$ vertices in the subgraph. In other words, each node in the group may lack edges to no more than k other group members. If $k = 1$ the k-plex is a clique; as the parameter k gets larger, each node is allowed more missing edges within the group. The dual definition of k-plex is the k-core, which is a subgraph whose vertices are adjacent to *at least* k other nodes of the group.

Looser notions of cohesive subgroups are not focused on the density of the group but on diameter-related features instead. *n-cliques*, for instance, are a maximal subgraph in which the largest geodesic distance between any two nodes is smaller than n. The smallest paths considered may also reside partially outside the group. If $n = 1$ the n-clique is a clique; the higher the parameter n is, the larger the community size will be. The subgraph induced by the nodes in the n-clique can be disconnected because two nodes may be connected only by a geodesic which includes nodes outside the group and by no path which is fully inside the n-clique. n-clans are a slight adaptation of n-cliques that impose that no geodesic distance between two nodes can be greater than n for paths *within* the group, thus avoiding the possibility of disconnection.

Instead of focusing just on the edges within the group it can be useful to consider also structural elements that lie outside the group. A *LS set*, for instance, is a cohesive subgroup that focuses on the frequency of edges among group members in relation to the ties to outsiders. Every subgraph S in the LS set must have more ties toward its complement $\overline{S} = LS - S$ than to the outside LS. LS sets can be nested hierarchically but they cannot overlap partially. LS sets can be extended to λ *sets*. Let $\lambda(i, j)$ be the number of ties that must be removed from a graph to leave no path between nodes i and j; the smaller is the value of λ the more susceptible is the pair of nodes to the disconnection. A set of nodes S is a λ set if any pair of nodes in S has larger λ than any pair of nodes with a vertex inside the set and one outside. Members of a λ set do not need to be adjacent, and since there is no restriction on the length of paths that connect two members of the set, it can be quite sparse over the network.

Even if the definitions of cohesive subgroup can help to understand how a module can be identified, they cannot be directly applied for graph clustering. Cohesive subgroups may not identify fully connected structures (e.g., λ sets) or may leave many nodes out of any cluster (e.g., cliques). In general, the previous definitions do not highlight well the community structure of the whole network, but instead are useful to detect important structures in the graph, like its main cores.

For this reason, ad hoc techniques for *network clustering* (often referred as *community detection*) have been developed and literally hundreds of algorithms have been proposed over the last few years [112]. The fundamental idea of community detection is to find a partition of the graph through agglomerative or divisive procedures such that some global (the entire network) or local (single partitions or nodes) goodness measure of the clusters is maximized.

Goodness indices have been defined on several structural indicators like the trade-off between inter- and intra-connectivity of clusters [271], betweenness [122], or other centrality indices [114], but for several years the most widely used metric was the *modularity* [248]. Formally, it is defined

as:

$$Q = \sum_i (c_{ii} - a_i^2). \tag{2.13}$$

C is a $k \times k$ matrix whose generic element c_{ij} is the fraction of all the edges in the network that connect vertices in cluster i to those in cluster j. Conversely, $a_i = \Sigma_j c_{ij}$ is the fraction of edges that connect vertices belonging to cluster i with nodes of other clusters. In a random graph the fraction of edges between clusters i and j is $a_i \cdot a_j$ and the fraction of the resulting edges that connect nodes within cluster i is a_i^2. Therefore, modularity measures the fraction of edges that connects vertices in the same cluster minus the expected value of the same quantity in a random network.

Even if finding maximum modularity partitions is an NP-complete problem [64], numerous algorithms based on suboptimal modularity maximization has been proposed [245, 97, 246, 278], until an intrinsic resolution limit of modularity in cluster detection was discovered [113]. Specifically, modules with size lower than $\sqrt{2|E|}$ cannot be resolved with modularity maximization procedures even in the extreme cases where communities are cliques connected by a single bridge to the rest of the network.

After this result, also in the wake of many other methods that proposed alternatives to modularity maximization [348, 256, 51], different fitness measures to define cluster quality have been introduced [178, 191, 180] and many recently proposed algorithms use local metrics to detect communities [112].

2.4 Network models

Macro statistical properties of networks allow to partition them in categories and to build models that can reproduce artificial networks with the same qualitative features. In the following, we present three well-known network categorizations with their corresponding models, focusing in particular on the diameter, the clustering coefficient and the degree distribution that the graphs originated by such models have.

Lattice ring network.

A simple deterministic models of graph topologies is given by the *lattice*, a network where nodes are connected according to regular grid-like patterns. The lattice ring network (or one-dimensional lattice) belongs to this class of graphs and it is originated by logically arranging nodes into a ring and connecting each node with the m closest nodes in the ring (with m even).

Nodes in lattice networks are densely connected locally and therefore the clustering coefficient is high. On the other hand, the shortest path connecting two generic nodes in the network is long on average and consequently the diameter is high.

Random network (Erdös-Rényi).

A random process of edge creation originates a random graph whose final degree distribution depends on the nature of the random process. Among the several random processes studied in graph network modeling [57], the most known is the one at the basis of the Erdös Rényi (ER) graphs [101, 102].

A ER graph can be generated given a number of nodes n and a probability of attachment p. The graph is generated through a memoryless process attaching every pair of nodes with probability p. The probability that a node has degree k is given by the binomial distribution:

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k} \tag{2.14}$$

and as n increases, the degree distribution tends to a Poisson distribution

$$P(k) \approx e^{-n \cdot p} \frac{n \cdot p}{k!} \tag{2.15}$$

meaning that the range of variability of node degree is relatively small.

The global connectivity of the graph depends on the n and p values, following a sharp threshold behavior. When $n \cdot p < 1$, it is very likely that the graph will be disconnected, with components of size $O(log(n))$. At the threshold value $n \cdot p = 1$ the graph will have with very high probability a connected component of size around $n^{2/3}$. Finally, if $n \cdot p > 1$, the graph will have almost surely a giant connected component containing the vast majority of edges. The giant component of ER graphs has very small diameters and low clustering coefficient. This means that the network can be walked across in a few hops but its node will tend not to form well-recognizable clusters.

Small world network (Watts-Strogatz).

Many real-world networks present characteristics that are between random and regular graphs. In particular, they exhibit a small diameter and a high clustering. In social networks, for example, social groups are highly clustered but two generic individuals are on average separated by few hops in the acquaintance network. To model this "small world" phenomenon, Watts and Strogatz [341] proposed a model (WS) based on a shortcut strategy on lattice ring networks.

Given a lattice ring graph, each node i is considered and each of its edges is replaced with probability p with a new edge that is rewired with a different endpoint chosen at random among the nodes in $V \backslash \Gamma(i)$. As the probability p increases, the macro characteristics of the graph change in a threshold-driven fashion. Approximatively, when $p \in [0, 10^{-3})$ the graph still preserves the main features of the lattice ring and when p is over 10^{-1} the shape of the graph becomes similar to a ER configuration. In the interval $[10^{-3}, 10^{-1})$ the graph assumes the desired small world properties. In particular, if compared with a random graph with the same size, this small world graph has a comparable average shortest path length but a much higher clustering.

Scale Free network (Barabási-Albert).

Watts-Strogatz graphs reproduce well the average path length and the clustering of many real networks, but they cannot model their degree distribution, that is much broader in real networks. To solve this problem, Barabasi and Albert [39] proposed the *preferential attachment* model of network growth.

Given an initial random network with m_0 nodes with degree greater than 1, the preferential attachment model grows the graph by adding a new node and connecting it to $m < m_0$ nodes with probability proportional to the degree of existing nodes. Specifically, the probability of connection with an existing node i is:

$$p_i = \frac{k_i}{\sum_j k_j}. \tag{2.16}$$

This "rich gets richer" strategy is similar to the growth process of real networks like the graph of World Wide Web, where very popular pages are linked by new pages with probability much higher than unpopular pages.

The average path length in the Barabasi-Albert model (BA) grows approximately logarithmically with the network size:

$$\langle \ell \rangle = \frac{\ln |V|}{\ln \ln |V|}, \tag{2.17}$$

and the clustering coefficient is higher than a ER graph with comparable size. The degree distribution of BS graphs is distributed as a power law of the form:

$$p(k) = c \cdot k^{-\alpha} \tag{2.18}$$

that defines a *scale free* distribution, meaning that the degree distribution is invariant under rescalings of the variable, or $p(ak) \propto p(k)$, with a constant. Applying the logarithm to the degree distribution we obtain:

$$\log(p(k)) = \log(c) - \alpha log(k), \tag{2.19}$$

that is the equation of a line with slope $-\alpha$. For this reason, plotting the degree distribution of a scale free network on a log-log scale produces a distribution that can be fitted linearly. It has been observed that the typical exponent of the power law degree distribution of many real-world graphs ranges between 2 and 3.

Even if clustering coefficient given by BA model is higher than in ER, it is still lower than the clustering in WS, whose clustering coefficient in much more adherent to the values found in real graphs. Therefore, both BA ans WS models fails in modeling some macro statistical properties of real networks. Nevertheless, such models gave a priceless contribution to the study and modeling of complex networks and provide useful abstractions for the comparison of properties of different systems.

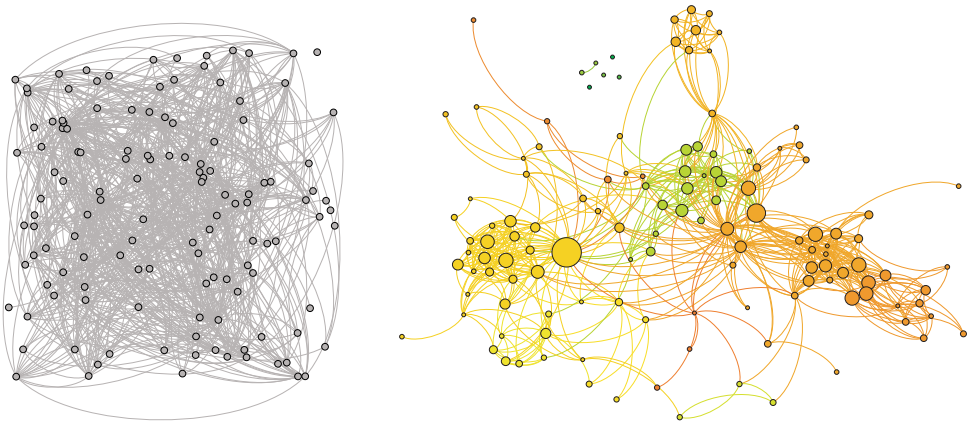
2.5 Visualization and analysis tools

Visualization of networks is an important phase of complex systems analysis. A good pictorial representation of a graph can highlight its most important structural components, logically partition its different regions, and point out the most central nodes and the edges on which the information flows more frequently or quickly.

The values of most of the metrics we defined in Sections 2.2 and 2.3 can be somehow represented using different nodes and edges colors, sizes and layouts. In Figure 2.4 we show how the visualization of a small graph can evolve from a random layout to a more clever, yet very simple presentation that conveys to the observer much information about the structure of the graph.

Many free tools for graph visualization have been developed in the last decade; among the most popular tools we mention:

1. Pajek [94] (`pajek.imfm.si`), one of the first visual exploratory tools for visualization and analysis of small graphs.



(a) Random layout

(b) Yifan Hu layout. Nodes are resized and colored

Figure 2.4: Visualizations of the author's ego network of Facebook contacts in 2009. From a representation with a random layout and no colors to a visualization where nodes are resized depending on their degree, colored according to their maximum-modularity cluster and arranged through the Yifan Hu graph layout [152]

2. Networkbench (`nwb.cns.iu.edu`), a tool to model and visualize networked datasets from different fields, in support of cross-disciplinary research.

3. Walrus [237] (`www.caida.org/tools/visualization/walrus`), visualizes graphs based on their spanning tree representation.

4. Gephi [42], sponsored as the "photoshop for graphs" (`gephi.org`). It provides an advanced GUI for visual manipulation and a vast API useful also for streaming visualization of dynamical graphs.

5. GUESS [4] (`graphexploration.cond.org`), visualization and analysis tool based on Gython, a domain-specific language that supports operators that can deal directly with graph structures in an efficient and intuitive way.

6. GleamViz (`www.gleamviz.org`), specifically designed to simulate and visualize spreading of infectious diseases across the globe.

Furthermore, many graph analysis packages like iGraph (`igraph.sourceforge.net`), networkx (`networkx.lanl.gov`), and R (`www.r-project.org`) provide network visualization tools or plugins.

Chapter 3

Dynamics of complex social systems

3.1 Unveiling evolutionary patterns in online social networks

Macroscopic dynamics of collaborative and social systems emerge from the aggregation of the behavioral footprints generated by the activity of the users and their interactions. Online social networks, together with online systems for content organization and sharing, entangle cognitive, behavioral, and social aspects of a user community through an underlying technological platform. The resulting "ecosystems" provide new possibilities to mine and investigate the various processes at play in the interactions of individuals, and to study the ways in which users relate with the information they share. Understanding and predicting their global dynamics allows us also to profile more accurately the individual actors, thus enabling a wide range of recommendation and personalization services.

Key open questions deal with understanding the concepts of *similarity* and *influence*, tracking the emergence of shared semantics, and determining the interplay between social proximity and shared topical interests among users. The emergence, spreading, and stability of any shared concept depend critically on these factors. As observed by danah boyd [63],

> *"In a networked world, people connect to people like themselves. What flows across the network flows through edges of similarity. The ability to connect to others like us allows us to flow information across space and time in impressively new ways, but there's also a downside. Prejudice, intolerance, bigotry, and power are all baked into our networks. [...] In a world of networked media, it's easy to not get access to views from people who think from a different perspective. Information can and does flow in ways that create and reinforce social divides. Democratic philosophy depends on shared informational structures, but the combination of self-segmentation and networked information flow means that we lose the common rhetorical ground through which we can converse."*

We see a pressing need for an investigation of these issues. Social media supporting many user-

profile features are especially relevant to a data-driven analysis of these questions, because they stimulate users to provide much information about themselves. For instance, services supporting *tagging* provide light-weight semantic annotations in the form of freely chosen, user-generated terms [125]. Social annotations based on tags are valuable for research because they externalize the three-way relation between users, items of interest (resources), and metadata (tags). Usage patterns of tags can be employed to monitor interests, track user attention, and investigate the emergence and spread of shared concepts through a user community. Moreover, several social media platforms support explicit representations of social links between users, making an objective definition of social proximity available. They also combine several aspects of user activity, such as exposing *resources*, belonging to discussion *groups*, and *communicating* to other users.

Online social systems have long been observed in their static network properties [230] and in the dynamics of their evolution [174, 187]. Fluctuations in time of network topological features like diameter, clustering coefficient and mixing patterns [7] have been explored in depth through data-driven analysis of large-scale real world systems. Previous work on link characterization have studied the patterns that describe the creation of links and how social ties features evolve in time [229, 181, 343].

More specifically, we find several other studies on the evolution of online social systems and correlations between different user features. Leskovec et al. [189] present a study on the Microsoft Messenger network, showing correlation between user profile information and communication patterns. The role of groups as coordination tools in Flickr is investigated by Prieur et al. [265]. They point out a strict relation between the density of the social network and the density of the network of tag co-usage among group members. Kumar et al. [186] perform a comparative study on the microscopic evolutionary dynamics between several social networks, in which a special emphasis is placed on the arrival process of new nodes and on the dynamics of attachment.

Influence of social contacts on browsing patterns in Flickr has been analyzed by Lerman and Laurie [184] and van Zwol [331], who provide insights into the activity patterns of users. Correlation between topical overlap among user interests in tagging systems and other indicators of social behavior has also been explored in CiteULike and Connotea systems [294]; since these systems lack an explicit social network component, the collaboration relations determined by the participation in the same discussion group can be considered as social substrate.

Recently, findings from social network analysis have been corroborated and expanded by the study of communication networks — also denoted as *activity* networks [82] or *interaction* networks [346] — that often coexist with social networks. The comparison of the graph of user-to-user interactions with the social network reveals several overlaps and similar connectivity patterns driven by reciprocity and triangle closure [188].

However, differently from the social networks, activity networks are much more dynamic and reflect inconstant trends in user interaction and information flow. Communication patterns have shown to be strongly clustered on an interaction frequency basis and to change quickly over both micro and macro time scales, even if the structural features of the activity network remains stable over time [334]. It has been observed that the average interaction level with neighbors in the social

network is very low [50] and often decreasing with time [334]; in agreement with this, thorough studies on Facebook interaction graph [346] reveal that social links that are effectively exploited for user-to-user communication are a minority. Such results confirm the intuition that social ties are not always good proxies to extract information exchange patterns. For an accurate description of the dynamics of online communities it is thus necessary to supplement the study on the social network with an analysis on the communication network.

Our work originates directly from this previous research knowledge. This Chapter, together with Chapters 4 and 5 analyzes important aspects of the evolution of online social networks using a data-driven approach on real datasets taken from three popular online social media. The three datasets differ significantly by size, category of resources, and the precise ways in which users tag resources and relate to each other.

Besides giving an overview of what are the structural properties that can be inferred by a static analysis of the social graph of such datasets, we focus on three phenomena that drives an important part of the dynamics of social environments. Namely, we will discuss how *communication* between individuals affect the social structure of the network, we explore the role that *homophily* has in the creation of social connections, and we analyze the *influence* patterns that occur in the network.

Our results highlight the heterogeneity of user activities and the correlations in the various metrics measuring the different activities of a single user. We also show the existence of non-trivial mixing patterns and we expose the substantial levels of topical similarity that exist among users who are close to each other in the social network and provide a causal relationship between homophily, influence, and such overlap.

Our findings shed light on the fundamental processes that drive link creation and that determine the evolution of features associated to the individuals in the social substrate. We can leverage such theoretical findings to provide ad-hoc services for the end users. In particular, we discuss a number of link prediction techniques aimed to a contact recommendation service, also known as "friend suggestion". Based on a number of topical similarity measures, we evaluate the performance of predictors based on some ground truth similarity of user profiles and on temporal data of the growth of social graphs.

3.2 Datasets

In the following, we report on the main features of our datasets and we describe the data retrieval methods we used to build them. Depending on the type of analysis we are interested in, we will focus from time to time to one specific dataset or on another, also according to their peculiarities and unique features. Table 3.1 summarizes the sizes of our three dataset.

3.2.1 Flickr

We collected the tagging information about the pictures uploaded in Flickr between January 2004 and January 2006 by means of API methods (`flickr.com/api`). The crawling effort was distributed by splitting the above time interval into smaller time windows to be crawled independently. A global

Dataset	Users	Triples	Tags	Tagged items	Groups
Flickr	130,840	90,723,412	1,420,656	20,599,583	92,301
Last.fm	90,049	6,971,166	194,763	894,615	69,306
aNobii	140,687	7,328,494	171,126	1,137,191	4,799

Table 3.1: Dataset statistics

tag knowledge base, initialized with a minimal set, was shared between parallel crawlers. Crawlers issued search queries limited to their specific time interval to retrieve information about photos marked with the tags stored in the common database. New tags were added to the shared database as they were discovered by individual crawlers.

Separate crawls were made to explore group affiliations and the social network. In Flickr jargon, social ties are called *contacts*; they are *directed* and do not require acceptance by the linked user. The overall crawl was performed during the first half of 2007.

Our analysis will focus on the network of about 130 thousand users for whom we have tag, group, and contact information.

3.2.2 Last.fm

In Last.fm, each user is linked to *friends* through undirected links that are established given the consent of both endpoints. Users also have a public list of *neighbors*, computed by the system as recommendations for potential new friendship contacts. An affinity value, ranging from 0 to 1, is also assigned to each member of the neighbor set. Users can annotate songs, artists or albums with tags, and can create or join groups. Users also have a public *library*, i.e., a list of the artists they have listened to. User profile information includes an optional geographic specification at the country level.

We used both API calls (`last.fm/api`) and web crawling methods to build the dataset. The API can be used to retrieve user profiles, friendships and neighborhood relationships and a list of the 50 top artists in the user library (i.e., those with the highest playcount). The API does not allow for the collection of a user's complete activity and group affiliation information, so we extracted the (*user, item, tag*) triples and the group membership relations via web crawling and scraping. The user set we consider was selected by a BFS crawl of the friendship network. The crawls took place in January 2010. We started from three randomly chosen users and for each of them we performed a crawl up to those nodes that resided 4 hops away. The corresponding snapshots include approximatively 100 thousand users each, with an overlap of about 20% between them. Since we found that the results of our analysis are consistent across the three samples, we report the findings for a single representative one.

Recently, the Last.fm API was extended with a similarity function, called *tasteometer*, which, given in input a pair of users or artists, returns an affinity score ranging from 0 to 1. This value is different from the one provided by the neighborhood score and, most of all, it can be computed for *any* pair of users or artists. Jointly with the crawling activity, we retrieved the tasteometer values for a large set of user pairs to compare the performance of our tag-based similarity functions

in the link prediction task with the performance of the tasteometer. Further details are given in Section 5.2.

3.2.3 aNobii

Users in aNobii (`www.anobii.com`) fill their digital book collections with titles selected from the public aNobii book database, which at the and of 2011 contained the metadata (such as publication year, authors, etc.) of about 30 millions different publications, written in 49 different languages. Each personal book collection is partitioned into a *library*, which is a set of titles that the user is reading or has already read, and a *wishlist* that lists the books that the user wants to read in the near future. Every book in the library can be marked with a reading status (e.g. finished reading) and can be annotated with arbitrary tags, a rating (from 1 to 5 stars) and a review.

Users can also provide public information about their profile, such as gender, age, marital status, and a detailed specification of their geographic location including country and hometown. Affiliation with thematic, user-generated groups is also possible.

Channels of social interaction are another crucial component of aNobii. Two different types of social ties can be established between users: *friendship* and *neighborhood*. The aNobii website suggests that people should be friends if they know each other in real life. Users should establish neighborhood ties with people who have a library they consider interesting. Surprisingly, although these two types of social links are formally different, they are equivalent from a structural point of view. In fact, they both are *directed* and can be created without any consent of the linked user, who is not even notified when a new incoming tie is established. Furthermore, both links activate a monitoring on the linked user's library that triggers notifications on library updates. Given this strong structural similarity, and since the two types of links are *mutually exclusive*, in the following we deal with the *union* between friendship and neighborhood networks and we generically refer to the union network as the aNobii social network. Users can write on the *message wall* of any other individual, no matter if they have a relationship on the social network. Self-posting is also allowed. The wall message history is public and the last five messages received are displayed on the main profile page.

We expored the aNobii social networks through web crawling and collected all the public user data trough web scraping. Crawling started in December 2009 from a random seed of users and followed the social links in a forward BFS fashion. We iterated the crawling procedure several times, 15 days apart, to take several snapshots of the network. We explored the entire giant strongly connected component and the out component of the social network. We collected each user's profile information, group affiliations, library, and tag assignments through web scraping.

3.3 Structural analysis

As we mentioned earlier, in most social media the activity of users has many facets. In Flickr, for instance, users can upload pictures and tag them, participate in groups, and comment on photos. In Last.fm, users can listen to music and tag songs according to their characteristics and personal

	Friendship	**Neighborhood**	**Union**	**Communication**
Nodes	126,859	77,357	140,687	80,303
Links	557,259	633,640	1,187,656	574,285
Loops	0	0	0	22,579
Reciprocation	0.60	0.43	0.54	0.61
$< k_{out} >$	4.4	8.2	8.4	7.2
$< w >$	1	1	1	1.8
$< m >$	-	-	-	12.9
WCC size	121,143	76,760	140,687	75,965
SCC size	81,292	41,063	100,492	38,336
Density	$3.4 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$	$6.0 \cdot 10^{-5}$	$8.9 \cdot 10^{-5}$
Average SPL	7.3	4.7	5.3	4.8
Diameter	25	15	20	17
Degree centr.	0.0072	0.0875	0.0486	0.0650

Table 3.2: Statistics on friendship network, neighborhood network, the union between them (i.e., the full social network) and communication network in April 2011. SPL=shortest path length; WCC=weakly connected component; SCC=strongly connected component; $\langle k_{out} \rangle$ average out degree, $\langle w \rangle$ average edge weight (applies only to the communication network, see Section 3.3.4), $\langle m \rangle$ average number of messages in the shoutbox.

tastes. In aNobii, users can add books to their libraries, tag them, join groups, and create a list of books they wish to read.

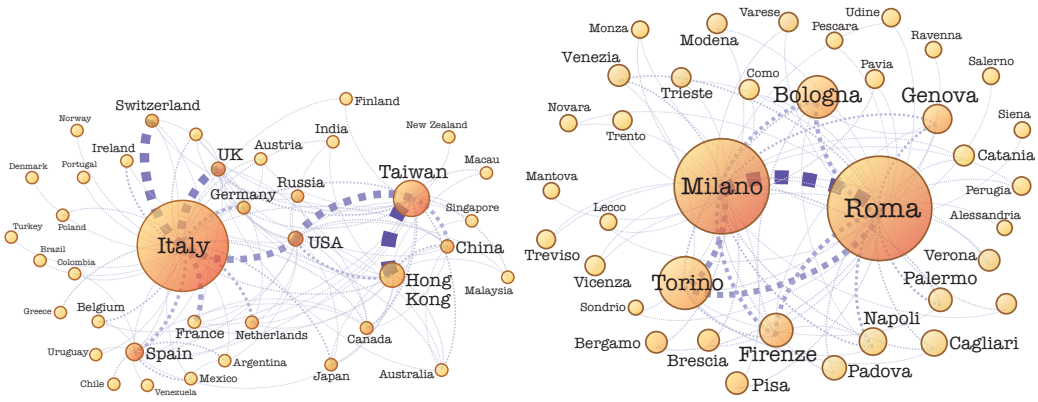
Since social networks are explicitly built by users, we can also consider the number of friendship relations to be a measure of activity in each social media we consider. When links are directed, we consider out-degree as a proxy for *activity*, while in-degree measures *popularity*.

In this Section, we first analyze the activity patterns of individual users, and show their considerable heterogeneity. We also investigate the correlations between various activity indicators.

3.3.1 Macro structural properties

Since the aNobii dataset includes a snapshot of the whole social graph, we focus on it for a preliminary analysis of the macroscopic statistical properties that define the network.

Macro topological properties of friendship and neighborhood networks are similar, but with some structural differences. As shown in Table 3.2, both networks have a high percentage of reciprocated links and a strongly connected kernel that includes the majority of nodes. However, the neighborhood network is slightly smaller, denser, and has higher degree centralization [340]. Its size is smaller because neighborhood ties tend not to be used by less active members and it is more centralized because of very popular libraries with many "followers": the range of variation of the in- and out-degree are broader for neighborhood than for friendship (for the in-degree, the maximal values are 1708 for neighborhood and 453 for friendship; for the out-degree, the maximal values are respectively 6537 and 705). These differences reveal that the two social ties are used



(a) Country graph: communities with less than 20 members and edges with weight less than 20 are not shown.

(b) Italian towns graph: communities with less than 100 members and edges with weight less than 100 are not shown.

Figure 3.1: Graphs of home countries and towns of aNobii users. Nodes are scaled according to the size of the communities and width and colors of edges are proportioned to the number of links that connects nodes between the communities

slightly differently by users; however, when discussing about properties that apply to comparable quantitative and qualitative extent to both networks, it is more convenient to consider the *union* between them. In the following, for simplicity, we will refer to the union network as the aNobii social network.

As a direct result of their structural differences, the diameters of the two networks (computed as the maximum shortest path length) are appreciably different. Still, they are both very high if considered that similar diameter values have been found for many other online social networks with much greater size [230]. The strong geographical clustering of the social network is the main reason behind this feature. The country-level graph of the social network depicted in Figure 3.1(a) reveals that the network has two main geographic communities, namely Italy (with roughly 60% of users) and Far East (composed by Taiwan, Hong Kong and China, that include less than 30% of users altogether). Since these two clusters are loosely connected to each other, the network has a dual core structure where connection between the two cores is mostly mediated by smaller communities (e.g., the USA cluster). Paths between individuals residing in different cores are thus longer if compared to a more ordinary single core configuration and, consequently, the diameter is higher. Narrowing down the view on town-level graphs inside clusters, the intra-cluster connections appear denser and structured around a single core of nodes (see Figure3.1(b)). Of course, since aNobii is focused on books, language is the main reason that leads to this sharp separation.

Besides the geographical location, aNobii profiles contain a rich information about users. User activity, along with social ties, can be measured by several indicators. Not surprisingly, the most popular activity is filling the library with books (94% of users have at least one book). Approximatively 50% of users added at least one book in the wishlist and roughly the same portion of users is member of at least one group. Books are annotated with reviews by around 40% of users and are rated by the 75%. Tagging activity is quite unfrequent. Finally, around 75% of users declare

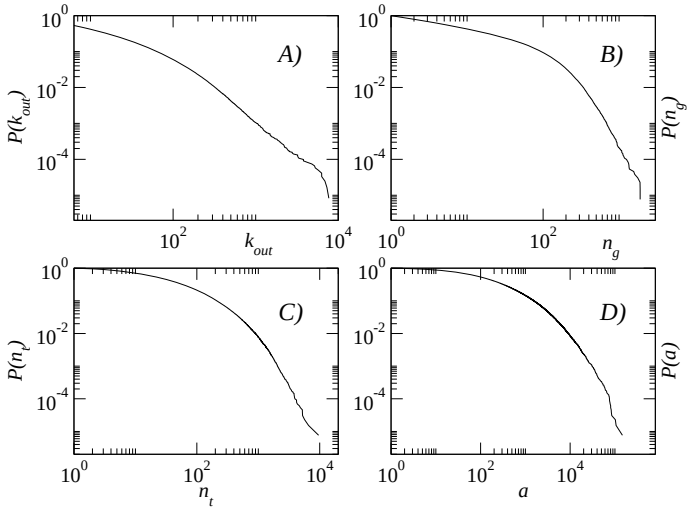


Figure 3.2: Flickr complementary cumulative distributions of (A) the number k_{out} of neighbors of a user, (B) the number n_g of groups of which a user is a member, (C) the number n_t of distinct tags per user, and (D) the number a of tag assignments per user.

at least one friend or neighbor.

3.3.2 Heterogeneity and Correlations

Figures 3.2 and 3.3 show the distributions of the number of neighbors in the Flickr and aNobii and the probabilities of finding a user with a given number n_t of distinct tags in her vocabulary, a total tagging activity a, belonging to n_g groups, and having (for aNobii) n_b and n_w books in her library and wishlist, respectively.

All these distributions are broad, spanning multiple orders of magnitude, showing that the activity patterns of users are highly heterogeneous. For each activity measure most users have little activity while some are extremely active, and all intermediate values are represented. No characteristic or "typical" value of the activity can be sensibly defined as evident from a standard deviation that is orders of magnitude larger than the average, for each activity measure.

Given this high level of disparity between users, a natural question arises about the correlations between the different types of activity: "do users who have many neighbors also use many tags, belong to many groups, and so on?" The simplest way to examine this issue is to compute the average activity of a type for users having a certain value of another activity type. For instance, we can measure the average number of distinct tags for users having k neighbors in the social network adapting Equation 2.12 to the tag dimension:

$$\langle n_t(k) \rangle = \frac{1}{|u : k_u = k|} \sum_{u:k_u=k} n_t^u \,, \tag{3.1}$$

where n_t^u is the number of distinct tags of user u. For the case of aNobii, we show in Figure 3.4 that all types of activity have a clearly increasing trend for increasing values of the out-degree; users who have more contacts in the social network tend also to be more active in terms of tags and groups. Overall, the various activity metrics are all positively correlated with one another.

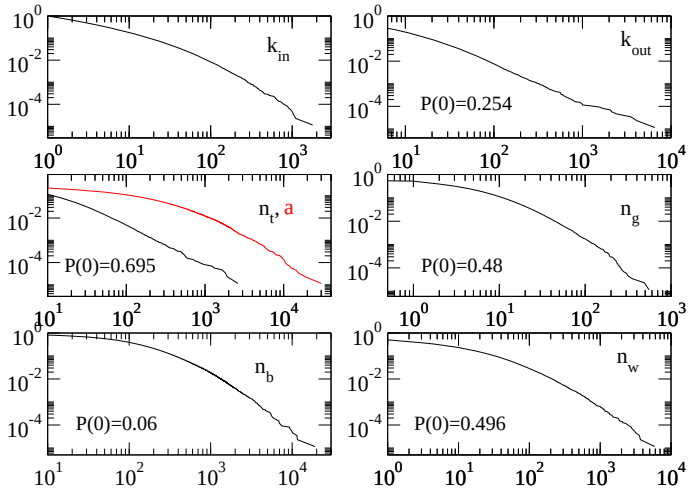


Figure 3.3: Complementary cumulative distributions of the measures of activity of aNobii users: in-degree k_{in} and out-degree k_{out} in the social network, number of distinct tags n_t and tagging activity a (total number of annotations by a user), number of group memberships n_g, number of books in a user library n_b and in a user wishlist n_w.

For instance, we show the correlation between the number of books in the library and the number of annotations or the number of group affiliations. Analogous results hold also for the other two datasets.

A slightly different to present the same result is to plot binned data with errorbars (Figure 3.5). In this kind of representation we show the cases on aNobii and Flickr. A third way to denote the correlation is the Pearson correlation coefficient [243]. In Flickr, the Pearson correlation coefficients are all positive, denoting positive correlation: 0.349 between k and n_t, 0.482 between k and n_g, 0.268 between k and a, 0.429 between n_t and n_g, 0.753 between n_t and a, and 0.304 between n_g and a.

Despite these correlations, large fluctuations are still present. First, the strong fluctuations at large degree values are due to the smaller number of highly-connected nodes over which the average is performed. Notably, users with a large number of social contacts but using very few tags and belonging to very few groups can be observed. We can investigate in more detail the degree of correlation between activity types through the conditional probabilities of the type $P(n_t|k)$, i.e., the probability for a user to have n_t tags, knowing that she has k neighbors in the social network, where the average $\langle n_t(k)\rangle$ is simply the first moment of this conditional distribution. As shown for some examples in Figure 3.6, these distributions, although narrower than the distributions shown in Figures 3.2 and 3.3, remain broad. This shows that, despite the strong correlations observed, users with a given activity level in the social network remain quite heterogeneous.

3.3.3 Mixing Patterns

While the previous analysis concerns the correlations between the diverse activity levels of a single user, another important question concerns the correlations between the activity metrics of users

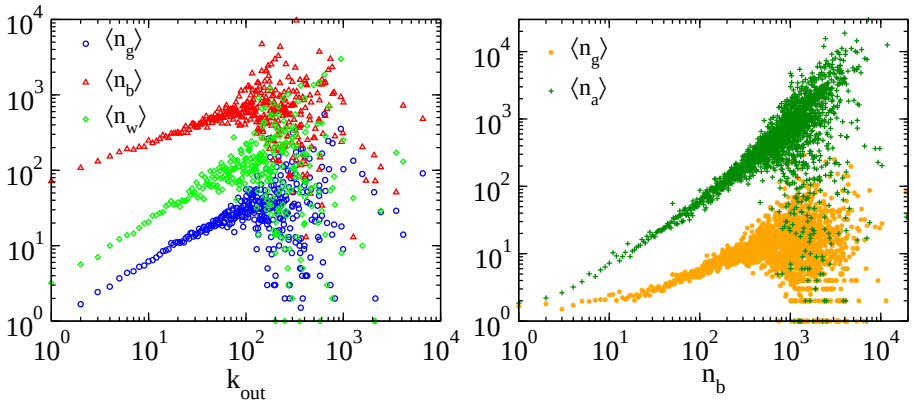


Figure 3.4: Correlations between different activities in aNobii. *Left*: average number of group memberships $\langle n_g \rangle$, of books in library $\langle n_b \rangle$ and wishlist size $\langle n_w \rangle$ against the number of social out-links k_{out}. *Right*: average number of group memberships and tag annotations $\langle n_a \rangle$ against the number of book in the libraries.

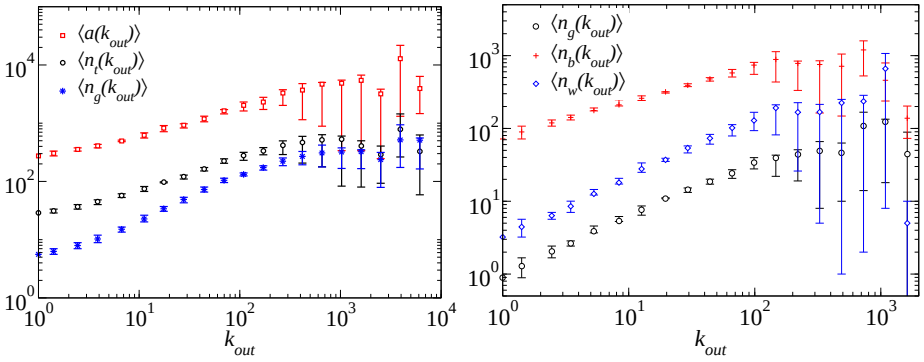


Figure 3.5: Left: Average number of distinct tags ($\langle n_t \rangle$), of groups ($\langle n_g \rangle$), and of tag assignments ($\langle a \rangle$) of users having k_{out} out-neighbors in the Flickr social network. Right: Correlations between the activity of aNobii users and their number of declared friends and neighbors: group memberships $\langle n_g \rangle$, library $\langle n_b \rangle$ and wishlist sizes $\langle n_w \rangle$, averaged over users with k_{out} out-links, vs k_{out}. The data has been log-binned: the symbols indicate the average, and the errorbars the 25 and 75 percentiles for each bin.

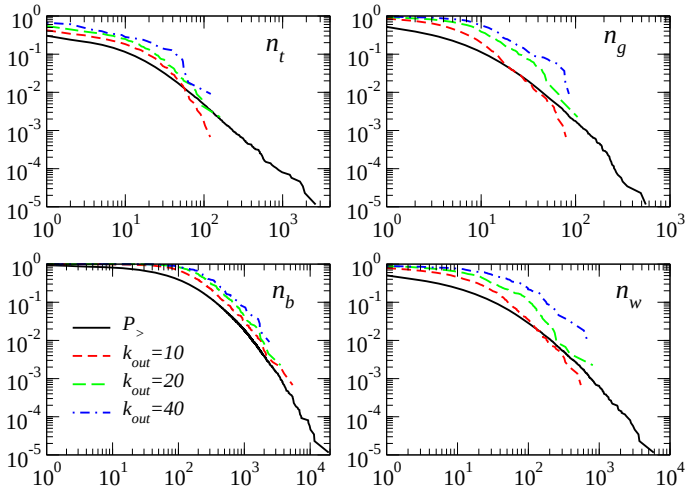


Figure 3.6: Complementary cumulative conditional distributions $P(n|k_{out})$ in aNobii, where n is the number of tags n_t, of groups n_g, of books n_b, and wishlist size n_w, compared with the global cumulative distributions $P_>$ (black lines). Even among the subset of users with a given k_{out}, a strong disparity is still observed in the amount of activity.

who are linked in the social network. This is a long-standing problem in the social sciences, ecology and epidemiology that is captured by the notion of assortative (or disassortative) mixing patterns we described in Section 2.2.2.

Mixing patterns can be defined with respect to any property of the nodes. In the case of social media, since each user is endowed with several properties characterizing his activity and social connectivity, it is interesting to characterize mixing patterns with respect to each of these properties. In the present case, we can characterize the mixing patterns concerning various activity types.

As seen in Section 2.2.2, one of the most investigated mixing pattern involves the degree of nodes. However, we can generalize the average nearest neighbors degree presented above for the case of any profile feature. We define the average number of tags of u's nearest neighbors $n^u_{t,nn} = \frac{1}{k_u}\sum_{v\in\mathcal{V}(u)} n^v_t$, and, similarly, the average number of tag assignments used by her nearest neighbors, $a^u_{nn} = \frac{1}{k_u}\sum_{v\in\mathcal{V}(u)} a^v$, the average number of groups to which her nearest neighbors participate, $n^u_{g,nn} = \frac{1}{k_u}\sum_{v\in\mathcal{V}(u)} n^v_g$, and, in the case of the aNobii dataset, the average number of books read by her nearest neighbors, $n^u_{b,nn} = \frac{1}{k_u}\sum_{v\in\mathcal{V}(u)} n^v_b$ and the average wishlist size of her nearest neighbors, $n^u_{w,nn} = \frac{1}{k_u}\sum_{v\in\mathcal{V}(u)} n^v_w$.

By analogy with the case of $k_{nn}(k)$, we can compute the average number of distinct tags of the nearest neighbors for the class of users having n distinct tags,

$$n_{t,nn}(n) = \frac{1}{|u : n_t(u) = n|}\sum_{u:n_t(u)=n} n^u_{t,nn} \ , \tag{3.2}$$

and the average number of tag assignments used by the nearest neighbors for the class of users with a tag assignments,

$$a_{nn}(a) = \frac{1}{|u : a(u) = a|}\sum_{u:a(u)=a} a^u_{nn} \ . \tag{3.3}$$

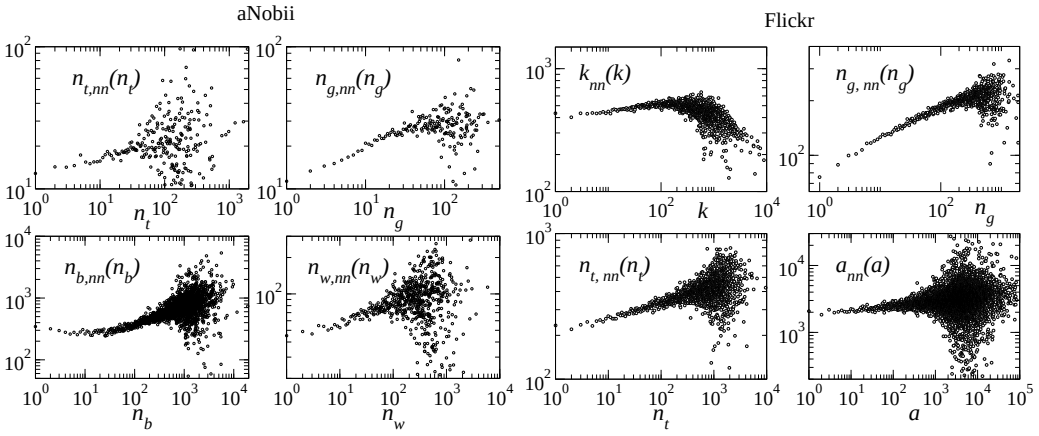


Figure 3.7: Mixing patterns in the aNobii and Flickr dataset: average number $n_{t,nn}(n_t)$ of distinct tags of the nearest neighbors of users having n_t distinct tags; average number $n_{g,nn}(n_g)$ of groups of the nearest neighbors of users belonging to n_g groups; average number $n_{b,nn}(n_b)$ of books of the nearest neighbors of users who have read n_b books; average wishlist size $n_{w,nn}(n_w)$ of the nearest neighbors of users who have a wishlist of size n_w; average out-degree $k_{nn}(k)$ of the nearest neighbors of users having out-degree k; and average number $a_{nn}(a)$ of distinct triples of the nearest neighbors of users having a distinct triples.

Analogous definitions can be used with respect to all other profile features. Similar formulae can be used to define the average number of groups of the nearest neighbors for the class of users who are members of n groups, $n_{g,nn}(n)$, the average number of books of the nearest neighbors for the class of users who have read n books, $n_{b,nn}(n)$ and the average wishlist size of the nearest neighbors for the class of users who have a wishlist of size n, $n_{w,nn}(n)$.

Figure 3.7 shows clear assortative trends for several measures for both the aNobii and Flickr datasets, as in other social networks [244, 223]. Similar results are obtained for Last.fm (not shown). As before, large fluctuations are observed for large activity values, because of the small number of very active users. But we see for all activity measures that the average activity of the neighbors of a user increases with the user's own activity —the activities of socially connected users are correlated at all levels.

3.3.4 Communication and interaction networks

Often, ties in social media are not categorized based on the intensity or on the type of the connections, mainly to allow users for an easy management of their acquaintances list. However, in a social context, ties might have different strength and meaning, depending on the information that flows on them and from the features that describe the individuals they connect. To reach a deeper understanding of social dynamics, the information on the social connections must be complemented with other relational data. In this respect, the communication network carries a useful information to augment the description of the social substrate as given by the user-declared "friendship" or "neighborhood" ties: some user-declared ties might not be the support of any communication, and

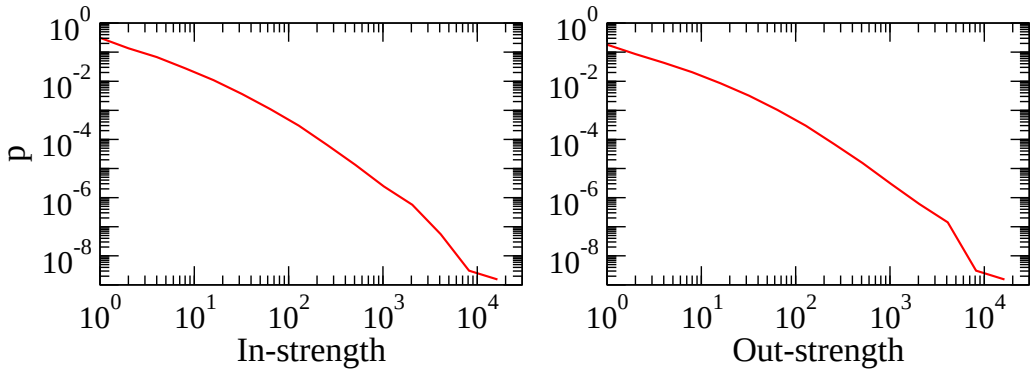


Figure 3.8: Distributions of in- and out-strength in the communication network

communication may occur between users that are neither "friends" nor "neighbors".

The most extensive way in which the communication history between individuals can be defined is through a *multigraph*, where each edge carries the information about a single message sent. To the purpose of our study we consider an aggregation over time and thus we define the *communication graph* as a directed graph where each edge between two nodes is weighted by the number of messages sent between these nodes. The communication graph is dynamic, as the frequency of messages exchanged by two users might change, with periods of inactivity followed by bursts of messages. The detailed study of this dynamics would be very interesting but goes beyond the scope of the present study, so that we consider an aggregation over the whole data set time window, and consider for each pair of nodes the total number of messages. More formally, we define the communication graph as:

$$G_c = (\langle U, E \rangle, W) | (u_1, u_2) \in E \Leftrightarrow u_1 \overset{msg}{\longrightarrow} u_2 \wedge W : E \rightarrow \mathbb{N}. \tag{3.4}$$

Similarly to previous work [346], we observe that macroscopic structural features of communication graph are analogous to those of the social networks. Degree distributions are very close to those found for the social networks (not shown) and the strength distributions (i.e., number of received or sent messages) shown in Figure 3.8 reveal an expected broad behavior. The statistics shown in Table 3.2 indicate that this graph has high reciprocation and centralization. Note that the communication network has self-links since it is possible for a user to write messages on her own shoutbox. Users keeping alive conversation threads on a single shoutbox or announcements published by the shoutbox owner are the main causes of this phenomenon. This behavior concerns however only 28% of the users, and the self-links represent only 4% of the total number of links. In Figure 3.9 we also observe that the social connectivity and the amount of books in the library are correlated with the activity on the communication network. A strong correlation is found also between in-degree and out-degree in the communication network.

As shown in Table 3.3, the overlap between social and communication graphs is significant but far from being complete. More than 75% of the socially connected pairs lack of any form of public communication and, conversely, around 25% of the communication channels are established between non connected users. We call *interaction graph* the portion of the social graph that overlaps

		Social\Comm	Comm\Social	Social∩Comm
#Nodes	Friendship	57,456	10,901	69,402
	Neighborhood	20,792	23,739	56,564
	Union	63,719	3,336	76,967
#Edges	Friendship	461,774	478,801	95,484
	Neighborhood	435,396	376,046	198,239
	Union	894,946	281,581	292,704

Table 3.3: Overlap between social networks and communication network.

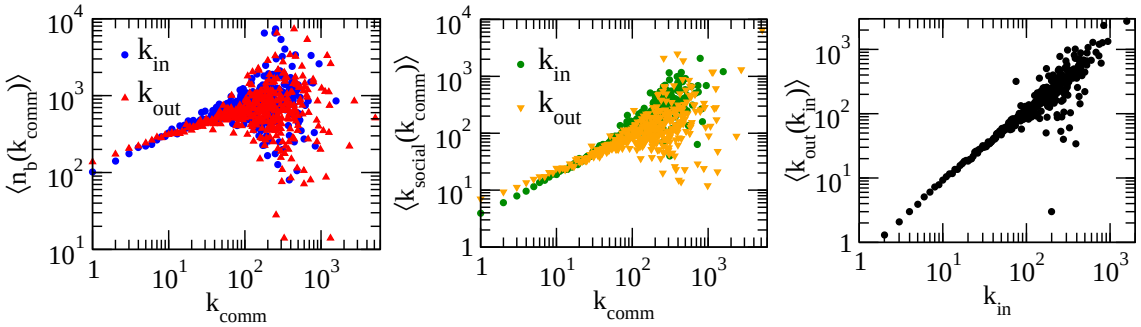


Figure 3.9: Correlation plots revealing the main motivations for message exchange. The number of incoming and outgoing messages is heavily correlated with the number of social acquaintances (k_{social}) and with the size of the library (n_b). The high reciprocation in communication implies also a strong correlation between in- and out- degree in the communication network.

with the communication network (i.e., $Social \cap Comm$ in the notation of Table 3.3).

3.3.5 Topical Alignment

The previous analysis has focused on the amount of user activity, as quantified by several metrics, and on the corresponding correlations and mixing patterns. To understand the interplay between the social network and user activities, it is necessary to take into account not only the amount, but also the nature and content of the user activities.

Assortative mixing patterns suggest indeed a propensity to local alignment between connected nodes, even though just quantitatively. We call instead *topical alignment* a static property of the social network for which individuals that are close in the social graph are more *qualitatively similar* if compared to those residing far. To compare users from this perspective, we therefore focus here on the *topical similarity* between user profiles as measured by the shared features —tags, groups, books, songs, and so on— in their profiles.

Before exploring the local similarity patterns, a first natural question regards the possible existence of some amount of *global* similarity between the users of a given social media. For instance, in the context of tags, a simple test for the existence of a globally-shared vocabulary can be performed by selecting pairs of users at random and measuring the number of tags they share, n_{st}.

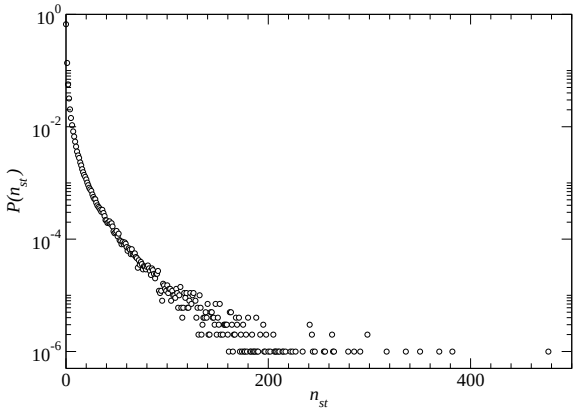


Figure 3.10: Probability distribution of the number of shared tags for two randomly chosen Flickr users. The probability to have no tags in common is $P(0) \approx 0.67$, but the overall distribution is broad.

In the case of Flickr, this measure shows that there is actually no shared tag vocabulary; this is not very surprising, given that Flickr is a narrow folksonomy (see Section 5.2) and the broad range of interests of the users. The average number of shared tags is only about 1.6 in Flickr, and the most probable case is the absence of any tags shared by the selected users (Figure 3.10).

Despite the lack of a globally shared profile, a number of mechanisms may however lead to *local* similarity of user profiles, in terms of shared tags, groups membership, books, musical tastes, and so on [223]. The presence of a social link suggests some degree of shared context between the connected users, who are likely to have some interests in common, or to share some experiences, and who are moreover exposed to each other's content and annotations. As an example, Table 3.4 shows the 12 most frequently used tags for three Flickr users with comparable tagging activity. User A and user B have marked each other as friends, while user C has no connections to either A or B on the Flickr social network. All of these users have globally popular tags in their tag vocabulary. In this example, the neighbors A and B share various interests (expressed by the tags in bold).

From this perspective, it is necessary to define robust measures of profile similarity between two users u and v, regarding the various types of activity. The first and simplest measure is given by the number of shared items for each activity. For instance, we can consider the number of common tags n_{ct} of the tag vocabularies of u and v, the number of common groups n_{cg} to which both u and v belong, the number of common books in their libraries or wishlists n_{cb}, n_{cw} for aNobii, and the number of common songs n_{cs} for Last.fm. These measures may however be affected by the amounts of activity of the users; two users who apply many tags may have more tags in common than two less active users, as it is more probable to find common items in two long lists than in two short ones.

For instance, let us consider two users with 100 tags each, and having 10 of them in common. The number of shared tags is 10 in this case, but represents just 10% of their tagging activity. Two users with the same 5 tags, on the other hand, have $n_{ct} = 5$, i.e. less than in the previous

Table 3.4: Tags most frequently used by three Flickr users; A and B are friends.

User A	User B	User C
green	**flower**	japan
red	**green**	tokyo
catchycolors	kitchen	architecture
flower	**red**	bw
blue	**blue**	setagaya
yellow	white	reject
catchcolors	fave	sunset
travel	detail	subway
london	closeupfilter	steel
pink	metal	geometry
orange	**yellow**	foundart
macro	zoo	canvas

case, but this represents 100% of their activity. In short, such simple measures are not normalized, and we therefore also need to consider measures that compensate for the heterogeneity in the amounts of activity. Therefore we need to normalize profile similarity measures to compensate for heterogeneous activity.

Let us first consider the case of the tags. Following the study by Cattuto et al. [75] we regard the vocabulary of a user u as a *feature vector* W whose elements correspond to tags and whose entries are the tag frequencies for that specific user's vocabulary, i.e., w_{ut} is the number of resources tagged with t by u. To compare the tag feature vectors of two users, we use the standard cosine similarity [292] defined as

$$\sigma_{tags}(u,v) = \frac{\sum_t w_{ut} w_{vt}}{\sqrt{\sum_t w_{ut}^2}\sqrt{\sum_t w_{vt}^2}}\,. \tag{3.5}$$

This quantity is 0 if u and v have no shared tags, and 1 if they have used exactly the same tags, in the same relative proportions. Because of the normalization factors in the denominator, $\sigma_{tags}(u,v)$ is not directly affected by user activity.

Similarly, we can define the cosine similarities for group memberships and for books. Since a user belongs at most once to one group, and adds a book only once to her library, the elements of the group and book vectors are binary, and the similarities reduce to

$$\sigma_{groups}(u,v) = \frac{\sum_g w_{ug} w_{vg}}{\sqrt{n_g(u) n_g(v)}};\quad \sigma_{books}(u,v) = \frac{\sum_b w_{ub} w_{vb}}{\sqrt{n_b(u) n_b(v)}}, \tag{3.6}$$

where w_{ug} is 1 if u belongs to group g and 0 otherwise, and w_{ub} is 1 if u has book b in her library and 0 otherwise.

Figure 3.11 gives an indication of how the similarity between users depends on their shortest path distance d on the social network, by showing the average similarity of two users as a function of d. In aNobii the average number of shared books is rather large for neighbors (close to 20), but it drops rapidly as d increases, and is close to 0 for $d \geq 4$. Similar results are obtained for the number of common groups and tags, and hold for Last.fm and Flickr as well. The cosine similarities display

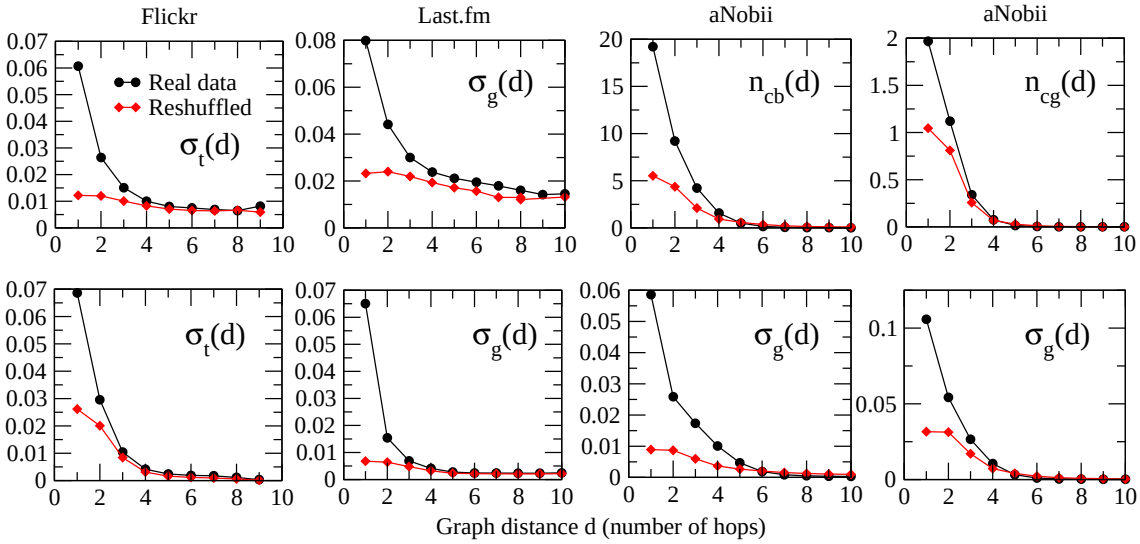


Figure 3.11: Average similarity of various activity indicators (tags t, groups g, books b) as a function of the distance on the social graph, in the three systems considered. The similarity can be computed simply with the number of common items ($n_{cX}(d)$) or with the cosine similarity ($\sigma_X(d)$). The diamonds correspond to the null model discussed in this Section.

the same decreasing trend as the distance along the social network increases.

The shortest path distance between two users gives the minimum number of steps to navigate from one user to the other on the online social network. However, this measure of topological proximity between users can be sensitive to the addition or removal of a single link, and does not take into account the fact that more than one path can connect the users. To overcome this issue, the personalized PageRank [143] of one user v with respect to another user u can be considered. This algorithm essentially gives the probability, for a random walker starting from the profile page of user u, to visit the profile page of v, and therefore is a more robust measure of social network proximity. As shown in Figure 3.12, the topical similarity between users increases with their social proximity, which is consistent with the trend of Figure 3.11.

To gain more insight into the entanglement between similarities and distance on the social network, we present in Figures 3.13 and 3.14 the probability distributions of the selected similarity measures for pairs of users at social distance d. The figures clearly shows the dependence of all distributions upon the distance of the users along the social network: for users who lie at small distances on the social network, rather broad distributions are observed for the number of shared tags, groups, or books. As the distance d along the network increases, the distributions become narrower. Two comments are in order: first, the distributions of n_{st} at short distances reach much larger values of n_{st} than in Figure 3.10 (the same is observed for the number of shared groups or books). The reason is that, when choosing a random pair of nodes (as in Figure 3.10), one is unlikely to select two neighboring nodes. Second, at any distance, the most probable value of n_{sg} or n_{st} is 0, even if the distributions are broad, and this probability increases with d. For instance, for Flickr users, the probability that two users do not share any tag is $P(n_{st} = 0) = 0.1$ if the users are

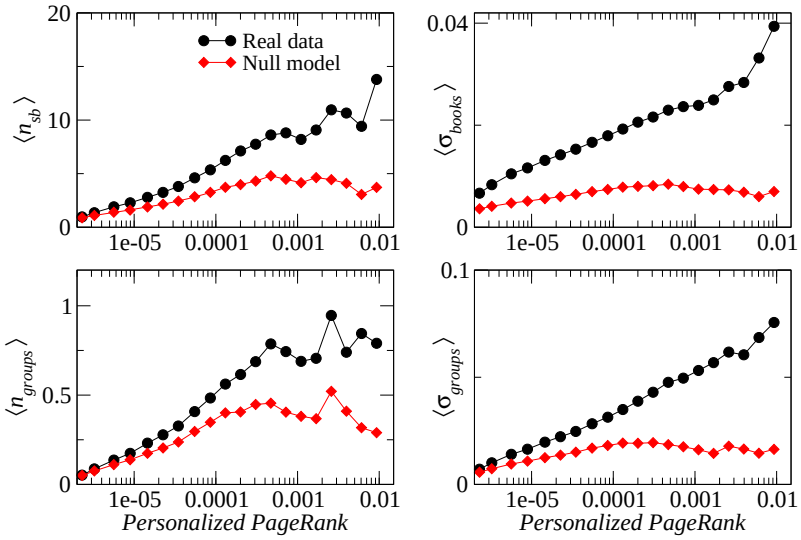


Figure 3.12: Average library (top) and group (bottom) similarity between two aNobii users as a function of the personalized PageRank of one user with respect to the other user. The diamonds correspond to the null model discussed at the end of Section 3.3.5.

neighbors ($d = 1$), 0.17 if they are at distance $d = 2$, 0.37 at $d = 3$. For groups, $P(n_{sg} = 0) = 0.17$ at $d = 1$, 0.4 at $d = 2$, 0.74 at $d = 3$. The distributions of cosine similarities between users at distance d show similar features: they mostly span the whole interval of possible values, but the probability of high similarity becomes smaller when d increases. The trends are similar for Last.fm (not shown).

The presence of assortative mixing patterns in the social network, with respect to the intensity of users activity, makes it necessary to investigate in more detail the observed local similarity of profiles. It could indeed be the case that such assortativity, by a purely statistical effect, yields an *apparent* local similarity between the tag vocabularies of users. For example, even in a hypothetical case of purely random tag assignments, it would seem more probable to find tags in common between two large tag vocabularies than between a small one and a large one. Furthermore, as we have shown, users who are more active have more friends, and their friends are also more active, therefore similarity with their friends may depend on their greater activity alone.

To discriminate between effects simply due to the assortativity and those due to actual profile similarity, one has to construct a proper *null model*, i.e., an artificial system that retains the same social structure as the one under study, but lacks any feature similarity other than the one that may result from purely statistical effects. This is done by keeping fixed the social network and its assortativity pattern for the intensity of the activity, but destroying socially-related feature similarity by means of a random permutation of profile items.

For instance, we proceed in the following fashion for the tags: (i) we keep the social network unchanged, preserving each user's degree k; (ii) we shuffle the tags among users in such a way as to preserve each user's number of tag assignments a as well as number of distinct tags n_t. This guarantees that the distribution of frequencies of tags is left unchanged. For group membership

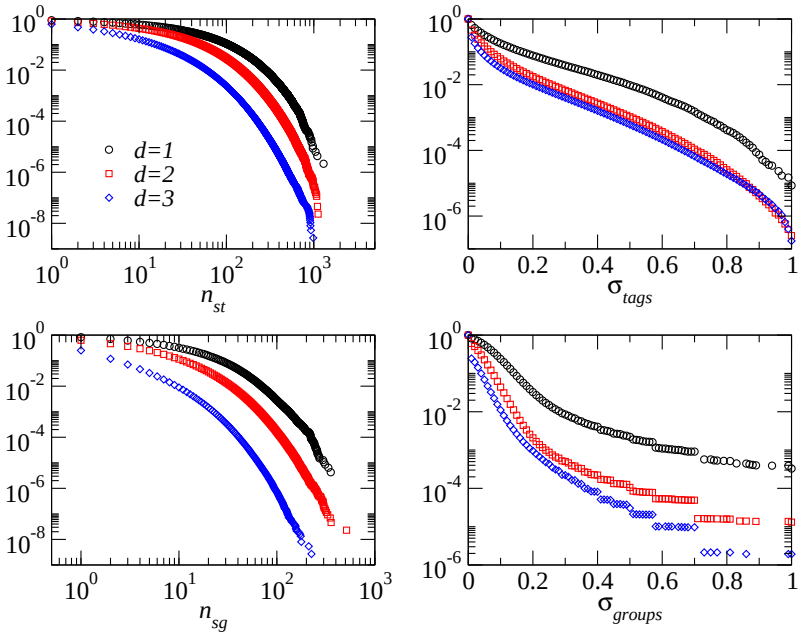


Figure 3.13: Left: Complementary cumulative probability distributions of the number of shared tags and groups for two Flickr users lying at distance d on the social network, for different values of d. Right: Complementary cumulative probability distributions of the cosine similarity between the tag vocabularies and group memberships of two Flickr users.

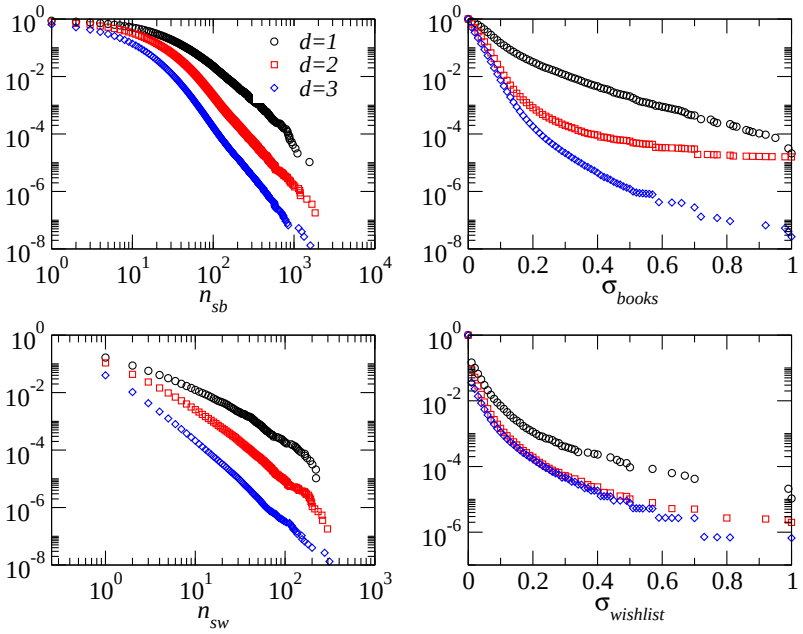


Figure 3.14: Complementary cumulative distributions of the number of shared books and (left), and of the similarities in the lists of books (right), in the libraries (top) and in the wishlists (bottom) of aNobii users lying at distance d on the social network, for various values of d.

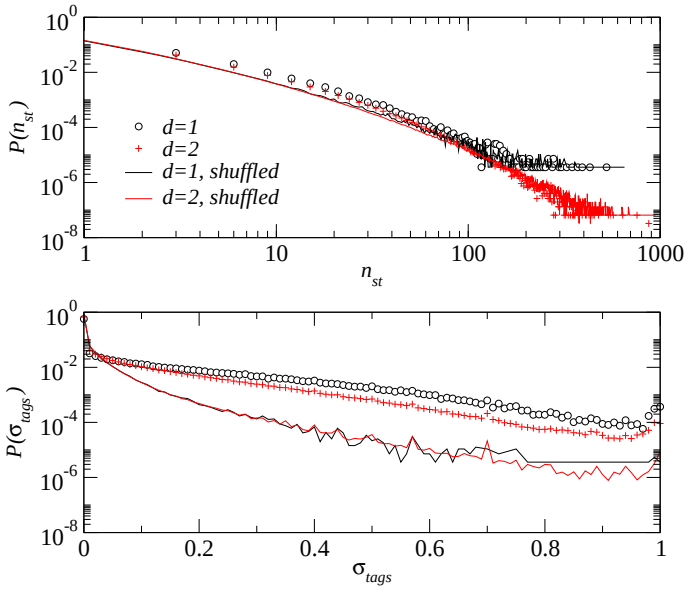


Figure 3.15: Top: probability distributions of the number of shared tags of two Last.fm users lying at distance d on the social network, for $d = 1$ and $d = 2$ (symbols), and for the same network with shuffled tags (lines). Bottom: same for the distributions of the cosine similarities of the tag vocabularies.

and for books, we can proceed in a similar way except we preserve in the shuffle each user's number of groups n_g and number of books n_b. Such a procedure is in the spirit of null models for detecting the importance of patterns in networks [220] and of random models of networks with given degree distributions or correlation patterns [233, 73, 301].

Using the null model defined above, we measure the similarity between users at distance d on the social network in the same way as for the original data. As Figure 3.11 shows, the average number of shared books in libraries and wishlists, as a function of the distance d, shows a similar trend to the original (non-shuffled) data. Similar curves are obtained for the number of shared tags and groups. For neighboring users, and also for next-to-nearest neighbors, the average numbers of shared tags or groups are generally significantly lower in the null model, but the distributions are very similar, as shown in Figure 3.15(top). The assortative mixing between the amount of activity of neighboring users is therefore enough to yield a strong topical similarity *as simply measured by the number of shared tags, groups or books*. The case of cosine similarity is different: as shown in Figure 3.11, the average cosine similarity in the null model does not depend as strongly on distance in the social network. Analogously in Figure 3.12 we see that the overlap measures in the null model are affected by social proximity, unlike cosine similarity. Figure 3.15(bottom) also shows that the distributions of σ_{tags} are very different for the original and shuffled data, and do not depend on distance in the case of the shuffled data.

We conclude that the topical overlap measured by the cosine similarity is a genuine non-random effect and it is not only due to the assortative mixing.

The same analysis can be performed on all the features of the users' profiles. For instance, the

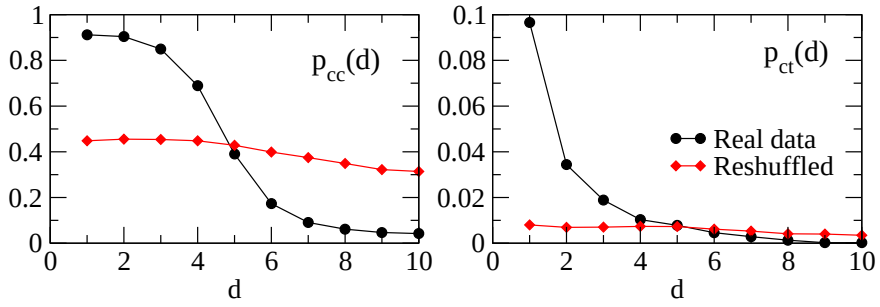


Figure 3.16: Fraction of pairs of users at distance d in the union network residing in the same country or town. In both cases data from the network with reshuffled links are shown.

relationship between the geographic attributes and the distance on the social graph are explored in the right plots of Figure 3.16 that show the probability that two users at distance d on the social graph are from the same country or town. Again, to disentangle this signal from statistical effects (given for example by the imbalance of the number of users in each nation) we use as null model a random network with the same degree sequence as the original network but reshuffled geographic attributes. The alignment on the nationality feature is strong up to a distance of 4 hops and a strong effect is observed as well for towns, most of all for directly connected users.

This result suggests that people preferentially establish social ties with others who speak the same language, but also that the social selection process is driven by the geographic proximity (e.g., people that reside in the same town). In particular, 90% of the social edges connect users from the same country and there is a 10% probability that two connected users are from the same city. This result indicates a decreasing trend of the probability of connection with geographic distance, as also found in other online social networks that are not based on a particular interest (here, the books) but have broader scopes [200, 182].

As we have seen previously, the fact that two users are connected does not automatically mean that they exchange information through messages. It is therefore of interest to compare the topical alignment on the social links that effectively are the support of communication ("Social∩Comm", in the notation of Table 3.3) with the alignment along the subset of social links on which no communication is observed ("Social\Comm", in the notation of Table 3.3). Figure 3.17 shows that the former is larger than the latter, but only slghtly: interestingly, strong alignment effects exist even on a network along which no explicit communication flows, and are almost as strong as in the network of communication.

3.3.6 Evolution of the network

Our aNobii temporal dataset allows us to study some regularities that emerge by the growth of the social network in time. Since the activity of profile updating does not change the statistical properties that we presented before, we focus just on the topological features of the graph. In Table 3.5 we report the evolution of some network parameters in a time span of 2 and a half months, with a granularity of 15 days. In each snapshot the largest component grows constantly

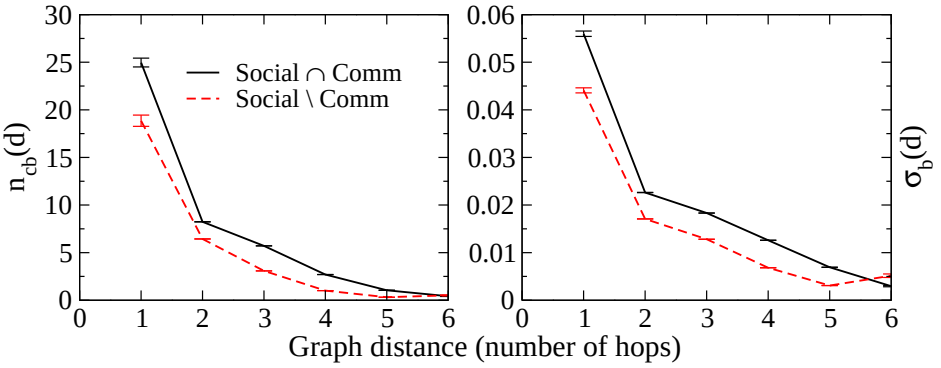


Figure 3.17: Average number of common books (left) and cosine similarity (right) against distance on the interaction graph and on the graph made by pure social contacts only. Standard error bars are shown in each curve.

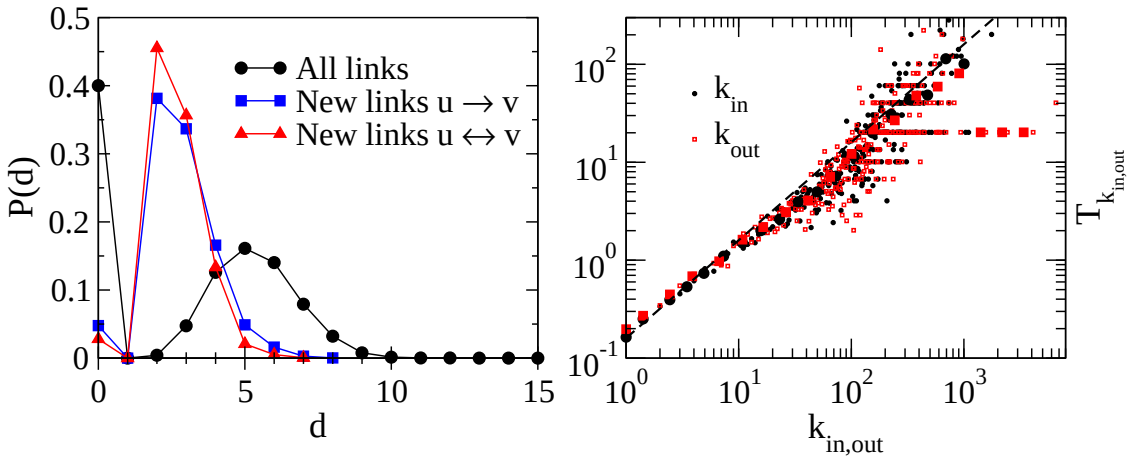


Figure 3.18: Left: distribution at snapshot 4 of the distances of nodes which become linked between snapshots 4 and 5, compared with the distribution at snapshot 4 of distances between all pairs of users. The points in $d = 0$ give the portion of pairs of nodes between which no directed path exists. Right: Measure of the preferential attachment. The dashed line represent a linear relationship.

due to the connections of new nodes to the graph core and new ties are also created between existent users. Node and edge deletion are much rarer events.

We classify newly created edges among existing nodes in three categories. $u \to v$ denotes the category of unidirectional links while $u \leftrightarrow v$ represents the new reciprocal links. "Reciprocated" denotes instead the new links from a node u to a node v, such that a link from v to u already existed. Links of the type $u \to v$ and $u \leftrightarrow v$ can be further described as "Simple closure" and "Double closure" ties respectively if they close at least a directed triangle. This fits the expectation that in a social network links are often established toward "friends of friends". This *triangle closure* phenomenon is evident also by looking at the distribution at time t of the distances of nodes that become linked at time $t + 1$. The comparison between such distribution and the distribution of distances between all the node pairs in the network (Figure 3.18 left) reveals that the process of

	$1 \to 2$	$2 \to 3$	$3 \to 4$	$4 \to 5$	$5 \to 6$
New nodes	2241	2121	1911	3214	3567
Removed nodes	239	222	230	220	684
New edges	19472	18324	17618	24805	26883
Removed edges	642	763	713	782	700
$u \to v$	5409	4942	5259	6546	6357
Reciprocated	1016	1155	1285	1526	1688
$u \leftrightarrow v$	1809	1597	1604	1924	2235
Simple closure	2070	1976	2143	2497	2382
Double closure	955	904	877	1027	1141

Table 3.5: Evolution of some quantities from one snapshot to the next.

social partner selection is very skewed on the topological vicinity of the user. In particular, more than 40% of the new arcs close triangles and more than 80% are established between nodes residing at distance at most 3.

Besides triangle closure, another phenomenon that underlies link creation in dynamic graphs is *preferential attachment*, i.e. users with large number of connection are preferentially chosen to establish a social link [17]. We test this hypothesis using the following method [241]. Let us denote by T_k the a priori probability for a newcomer to create a link toward a node of degree k, between time $t-1$ and t. Given that at time $t-1$ the degree distribution of the $N(t-1)$ nodes is $P(k, t-1)$ (i.e., there are $N(t-1)P(k, t-1)$ nodes of degree k), the probability to observe a new link from a new node to a node of degree k between $t-1$ and t is $T_k P(k, t-1)$. Therefore, we can measure T_k by counting for each k the fraction of links created by new nodes that reach nodes of degree k, and dividing by $P(k, t-1)$. As shown in Figure 3.18 (right), we obtain a linear behavior, both when considering for k the in and the out-degree (which are strongly correlated). This is a clear signal of a linear preferential attachment.

Clearly, users do not have any knowledge of the overall network topology at anytime, so they cannot be more motivated to connect to the most connected users. It is more likely that this preferential attachment arises from the fact that a new user creates links not only towards another user but also towards some of this user's neighbors. It has been shown that this locally-driven connection pattern results in effective preferential attachment [168, 173]. Indeed, we verified in our dataset that many newcomers join the network by creating links to pairs of already connected users.

Chapter 4

Homophily and influence dynamics of complex social systems

4.1 Causal connection between similarity and link creation

Similarity effects between the members of social groups, or between individuals sharing a social link has long been observed and studied in sociology [223]. The increasing availability of data from online social networks has created ideal laboratories for testing and quantifying such social phenomena and theories [323, 190].

As often discussed in the social sciences, similarity can emerge for different reasons, which are summarized in two scenarios: *link selection* (or *homophily*) and *social influence* [183, 22, 223, 303]. The former scenario considers that social links are preferentially created between individuals who are already similar and choose each other for establishing the social link precisely because they share some degree of similarity. In the latter scenario, individuals become more similar over time because they influence each other. Disentangling these scenarios is a delicate matter that requires longitudinal data sets, as social influence implies a temporal evolution of a relationship [27, 303].

In particular, investigations on the interplay between homophily-driven creation of social connections and the influence that neighbors exert on each other's behavior have been made by Crandall et al. [91] on the Wikipedia collaboration network. Authors show that editors are likely to establish direct communication if they start having many editing activities in common. On the other hand, the interaction between them is found to result in reciprocal influence that is measurable in terms of further alignment between their activities. Here we address a similar problem but on a social media that is very different nature from Wikipedia, and our focus is on the analysis of the profile features, shared metadata, and topicality rather than on collaboration patterns.

In the previous Chapter we observed topical alignment as a static property of the network. Here we investigate the cause of this phenomenon more in detail. Since we verified that topical alignment is not purely due to assortative patterns, we can ascribe this phenomenon to homophily or to social influence. Given the temporal dataset from aNobii we can verify if such phenomena

	$\langle n_{cb} \rangle$	σ_b	$\langle n_{cg} \rangle$	σ_g
$d_{uv} = 2$	9.5 (0.2)	0.02	1.12 (0.61)	0.05
$u \rightarrow v$	12.9 (0.16)	0.04	1.1 (0.6)	0.08
$u \leftrightarrow v$	18.5 (0.06)	0.04	1.67 (0.44)	0.11
Simple closure	18.2 (0.09)	0.04	1.81 (0.45)	0.1
Double closure	23.4 (0.03)	0.05	2.2 (0.36)	0.12

Table 4.1: Average similarity for snapshot $t = 4$ of pairs forming new links between t and $t + 1$, compared with the average similarity of all pairs at distance 2 at t. The similarity is measured by the number of common books n_{cb} or groups n_{cg}, and by the corresponding cosine similarities σ_b and σ_g. The numbers in parenthesis give the probability to have similarity equal to 0.

are present.

To check whether homophily-driven attachment is present we compare the average similarity, computed at time t, between pairs of nodes residing two hops away in the social graph ($d = 2$) and between pairs of users who create e social connection between t and $t + 1$. We observe in Table 4.1 that pairs of users that are going to get connected are more similar than the average of all the nodes that simply reside two hops away in the graph, especially if we consider bidirectional links and closing triangles. This result applies for all the similarity measures used. The probability that two users at distance 2 have 0 similarity is also much smaller for users who become linked between t and $t + 1$.

The picture emerging from this analysis and from the results of the structural analysis of the datasets is the following: users connect to others residing close in the social graph, very often neighbors of neighbors; moreover, these individuals have more similar profiles than the average pairs of users at distance 2. In this respect, one can infer that the first causal effect of topical alignment is homophily, namely that similarity of users partly drives the creation of new links.

Verifying the presence of social influence requires instead to study the evolution in time of the similarity between connected user profiles. In Figure 4.1 we plot the average similarity for library and group membership features for pairs connecting between t and $t + 1$. Before the link is created the similarity score is rather stationary and a sudden jump is observed when the connection is created; the similarity then continues to grow, albeit at a slower rate. The following scenario emerges from this result: after a link creation, newly connected individuals take inspiration from each other for new books to read and new groups to join. The direct consequence of this reciprocal influence is a further alignment of the profiles.

To summarize, our analysis on the dynamics of social aggregations show the presence of a *bidirectional* causal relationship between social connections and similarity. Higher similarity determines a higher connection probability and, on the other hand, users who get connected become more similar due to the influence that new acquaintances exert on one another. These results applies not only for collaboration networks [91], but also for the more general case of interest-based networks such as aNobii, where the similarity between users is evaluated on the basis of profile items, shared metadata, and topics of interest.

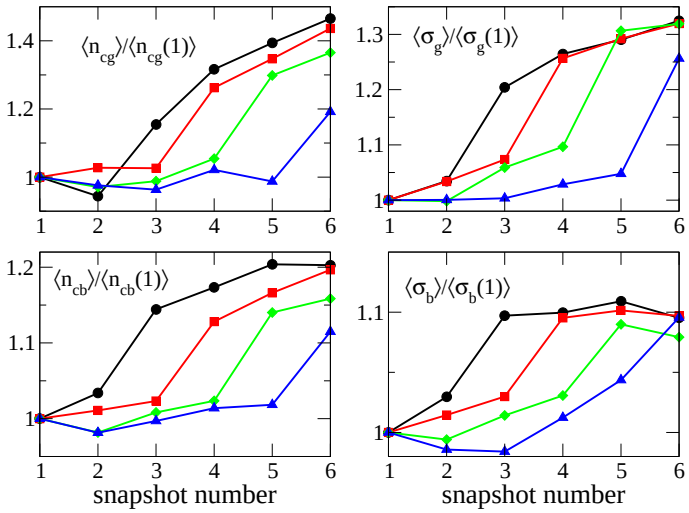


Figure 4.1: Evolution of the average similarity of user profiles, as measured by the numbers of common books or groups, and by the cosine similarity. Similarity is shown for links created between t and $t+1$, for $t = 2$ (black circles), 3 (red squares), 4 (green diamonds), 5 (blue triangles), normalized by the average similarity in the first snapshot. Values are quite stationary before t_0, and clear jumps are observed between t and $t + 1$.

4.2 Information spreading and influence

Influence can also be investigated from a different angle, focusing on items rather than on users. The influence observed at the time of a link creation might indeed remain effective for the whole life span of the social link, and, at any time, may lead a user to adopt a new item (in particular a book) from her neighbors and, in turn, to influence others to adopt the same item. This phenomenon gives origin to adoption cascades that can be studied within the more general scope of information spreading [41]. Better understanding the spreading of items on the network can shed a clearer light on the overall role of influence in the online social network.

The task of capturing the dynamics of information spreading and influence that occur in networked environments has received much attention recently. Diffusion models of word-of-mouth processes have been developed in the past to enhance viral marketing strategies [167]; more recently, due to the large diffusion of social media, detection of influence patterns and of influential individuals have become important also to capture dynamics of interaction in social networks and especially in real-time information networks.

Analysis of information propagation in Flickr [78] showed that, in spite of the common expectation about the quick and wide spreading of information delivered through word-of-mouth, diffusion is limited to individuals who reside in the close neighborhood of the seed user and the spreading process is very slow. Analysis of message cascading on Twitter has been used to estimate the degree of influence of users [354]; the most influential among a pair of users is determined using the difference between some activity metric, like the number of followers or number of tweet replies. In partial disagreement with this study, it has been shown that the number of followers (or of social

contacts in general) does not imply a high influence degree [77].

Instead of representing the influence as an infection phenomenon between connected individuals, Yang and Leskovec [352] recently proposed a linear influence model which is agnostic on the network structure and relies only on the time of the contagion; they show its accuracy in predicting influenced nodes. These observations imply the presence of a hidden contagion web which is different from the observed social network [128]. Based on similar observations, other probabilistic models that represent influence effects between peers disregarding social links structure have been proposed [29].

A crucial task in the analysis of influence patterns is to discern real influence by unobserved factors, like homophily or confounding variables, that can induce statistical correlation between the behaviors or the profiles of connected users even without one being influenced by the other. Shuffling or randomization tests on user features are commonly used to detect a signal of influence inside noisy patterns of correlation between pairs of users [22].

In this perspective, we study the static and dynamic properties of the book graphs $G(b)$ defined as the social subgraph induced by the users having the book b in their library or wishlist. We differentiate the analysis by classes of book popularity using the cardinality of the set $A(b)$ of the users who adopted the book b (i.e., the nodes in $G(b)$). In particular, given the book popularity distribution, we introduce three popularity classes, namely the *rare* ($|A(b)| \in [10, 500)$), the *middle* ($|A(b)| \in [500, 1000)$), and the *popular* ($|A(b)| \geq 1000$). The boundaries of the popularity classes are chosen based on the empirical observation of the popularity distribution of books. Even neglecting very rare books with less than 10 readers we have more than 1.5M of book graphs to study.

In this perspective, we study the static and dynamic properties of the book graphs $G(b)$ defined as the social subgraph composed by the users having the book b in their library or wishlist and by the links between them. We differentiate the analysis by classes of book popularity using the cardinality of the set $A(b)$ of the users who adopted the book b (i.e., the nodes in $G(b)$). In particular, given the book popularity distribution, we introduce three popularity classes, namely the *rare* ($|A(b)| \in [10, 500)$), the *middle* ($|A(b)| \in [500, 1000)$), and the *popular* ($|A(b)| \geq 1000$). The boundaries of the popularity classes are chosen based on the empirical observation of the popularity distribution of books. Even neglecting very rare books with less than 10 readers we have more than 1.5M of book graphs to study.

We first report some static properties of the book graphs. Figure 4.2 shows the broad distributions of the numbers of nodes and edges; book graphs can be disconnected and more than 10% of them are composed just by singletons, even if this is observed only for graphs with at most 100 nodes. Patterns of connectivity in books graphs can be detected by measuring topological graph features depending on the network size. In Figure 4.3 we report the relative size of the greatest connected component ($S_{gcc}/|A(b)|$), the relative number of connected components ($N_{cc}/|A(b)|$), and the clustering coefficient (C) against the size of the book graph $|A(b)|$, for every book. For the sake of comparison, for every point in the scatterplot we also depict two twin points representing the same topological measure calculated for two different random graphs. The first is an Erdős-Rényi (ER) graph with the same number of nodes and edges and the second is a random subgraph

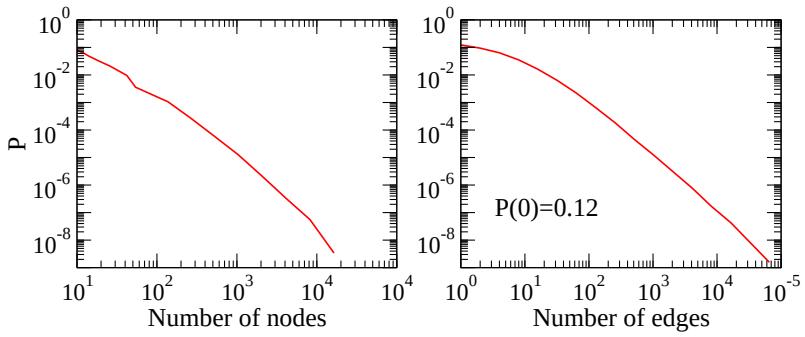


Figure 4.2: Distribution of number of nodes and edges in the book graphs $G(b)$.

of the social network with the same number of nodes. In particular, the random subgraph is used as a null model that represents the adoption of a book as an aleatory process with probability $|A(b)|/N$ (with N the total number of users).

Book graphs exhibit a weaker connectivity but a more clustered shape than the corresponding ER graphs. The relative number of connected components slowly decreases with the size but remains considerably higher than in ER graphs with the same size; as a consequence, the relative size of the greatest component asymptotically stabilizes around a value slightly under the case of the ER graphs. Conversely, real book graphs are much more clustered than their randomized versions. The comparison with the random-node-graph null model reveals instead that a random adoption would give origin to much more fragmented book graphs with smaller clustering, thus suggesting that book graph may be originated by a process of expansion and enforcement of clustered cores of readers.

The dependance of the same quantities on the average number of outgoing ties $\langle k \rangle$ in the subgraph can also be measured; in Figure 4.4 we focus on the relative S_{gcc} and on the average clustering coefficient C. The size of the largest component grows steadily with the average number of neighbors, while a relatively rapid transition from 0 to 1 is observed as $\langle k \rangle$ crosses 1 in the ER graphs with the same size (corresponding to the percolation transition of ER graphs), and no significant size increase is detected in the random-node-graphs. Also the clustering coefficient grows almost linearly with $\langle k \rangle$, while it remains very low in the ER network and in the random-node-graph. This results suggest that the connectivity in book graphs is not driven by a threshold on the average node connectivity (as is the case in ER graphs); instead, for any level of average connectivity and graph size, a non-negligible portion of isolated components are detected. This tends to indicate that several users adopt a book independently, without being directly influenced by their online social contacts. However, the strongly clustered nature of $G(b)$ suggests that independent adoption by distinct users is not the only driving force of book adoption, and that a process of "contagion" between users might have taken place in the shaping of the subgraphs of adopters $G(b)$.

To have a better idea of the extent to which a user might be led to adopt a book through the influence of her social neighborhood, it is necessary to analyze the temporal evolution of the $G(b)$ graphs. We call $G(b,t)$ the social subgraph of users having book b at time t. $G(b,t)$ can evolve

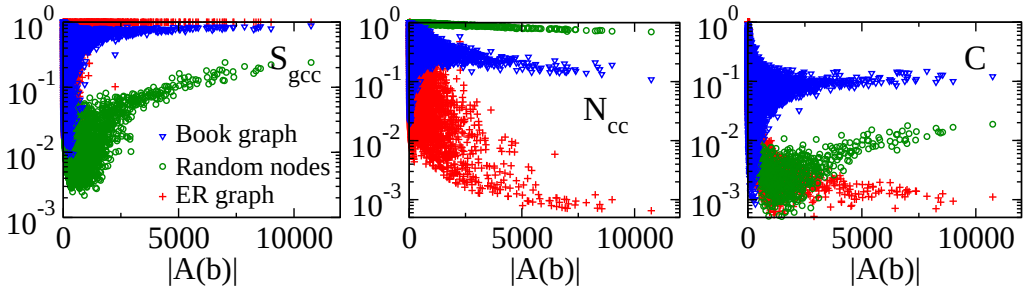


Figure 4.3: Scatterplots of number of relative size of the giant connected component (S_{gcc}), relative number of connected components (N_{cc}), and clustering coefficient (C) vs. number of nodes in the $G(b)$ graph. For each graph, values for the corresponding ER graph with the same number of nodes and edges are reported for comparison.

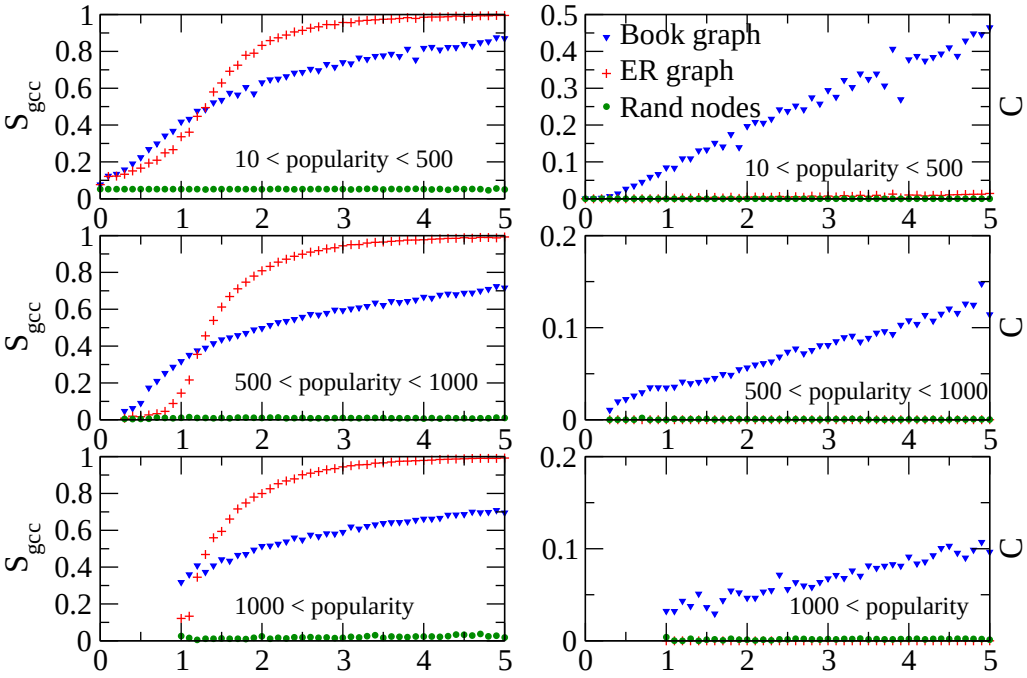


Figure 4.4: Relative size of giant connected components (S_{gcc}) and clustering coefficient (C) vs. average out degree ($\langle k \rangle$), averaged at fixed values of $\langle k \rangle$. Comparison values from corresponding ER graphs is resported.

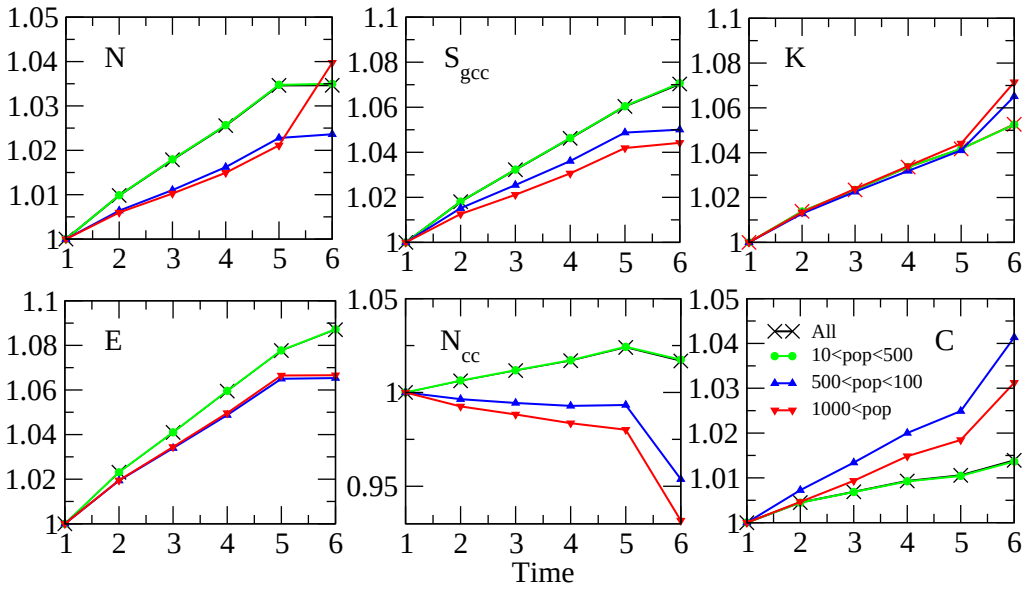


Figure 4.5: Evolution of structural properties of book graphs in time. Number of nodes (N), number of edges (E), size of the giant connected component (S_{gcc}), number of connected components (N_cc), average out-degree (K) and clustering coefficient (C) are shown. All values are normalized on the initial value ($t = 1$).

because of new users arriving in the social network who have b in their library, users leaving, or users adding/removing b to/from their library. For the purpose of detecting influence patterns, we disregard the newcomers (who might or not fill their own library with the books they have read) and users leaving the network, and focus on the graph $G^*(b,t)$ restricted to the users who are present in all the considered snapshots. Moreover, for simplicity, we neglect the (very rare) event of book deletion: once a book is adopted by a user, we assume that it is present in her library at anytime in the future. In this context, we formally define the set of *adopters* of a book b between time $t-1$ and t as $A*(b,t) = G^*(b,t) \backslash G^*(b,t-1)$.

In Figure 4.5 the evolution of some properties of $G^*(b,t)$ graphs is shown. Most of the values (N, E, K, C, S_{gcc}) grow in time, revealing the expansion and the increase of density and cohesion of the greatest component. The only exception is observed for the decreasing trend in the number of connected components for the graphs of the books with medium or high popularity. This can be explained by the fact that if a book is widespread over the social network it is more likely that a new adopter can create a bridge between two components of $G^*(b,t-1)$, thus reducing their number.

Finally, for every adopter, we measure the fraction of users that could potentially have played an influence in the book adoption process. If a book is adopted in the time span $[t, t+1]$, the users that may have influenced the adopter are her out-neighbors who already have that book in their library at time t [1]. We specifically focus only on the out-neighbors because users are explicitly

[1] We disregard here the possibility of interactions between users taking place outside the social network. It is clear that what can be inferred from the analysis of the online social network are only tendencies and indications, and

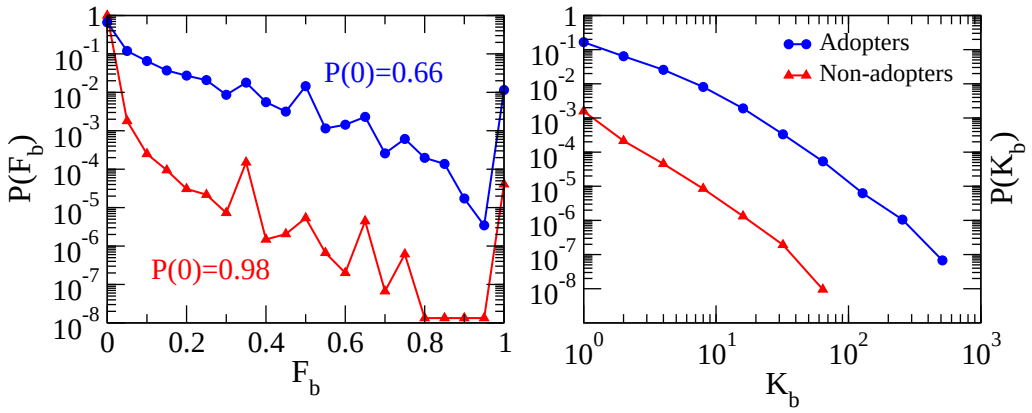


Figure 4.6: Distributions of number (K_b) and portion (F_b) of neighbors on the social network having book b at time t for adopters of book b between t and $t+1$ and non-adopters. For adopters, the same distributions computed on the interaction network only are shown.

notified of all their new book adoptions, while a user may not be aware of her in-neighbors and of their activity. Consequently, we denote the number of user u's out-neighbors at time t having book b as $K_b(u)$ and the fraction of such users over all u's out-neighbors as $F_b(u) = K_b(u)/K_{out}(u)$. The distributions of K_b and F_b for the users u who adopt b in $[t, t+1]$ are shown in Figure 4.6, together with the same distributions restricted to the users u who still do not have adopted b at $t+1$. The curves for the two user categories are very different for both measures, thus revealing that users that adopt a particular book have been exposed, on average, to a higher number of users that previously put that book in their libraries. In particular, the probability of having no out-neighbors at t with the target book in their library is much lower for the adopters (0.66) than for the non-adopters (0.98). Furthermore, as shown in Figure 4.7, the average K_b at fixed values of K_out is much higher for adopters than for non-adopters, even if a positive correlation is find in both cases, meaning that adopters are considerably more exposed, on average, to other users exposing the adopted book. Such clear differences between the two cases of adopters and non-adopters represents *a very strong evidence of the presence of an influence effect in the process of book adoption*.

Interestingly, the vast majority (74%) of adopters with $F_b > 0$ exhibit values smaller than 0.2, and the average value of F_b for these adopters is rather small (0.189); on the other hand, the numbers K_b of neighbors of an adopter who already have the book b are broadly distributed. This tends to support two distinct hypothesis: the first one is that only a rather small number of neighbors are really influential among the neighborhood of a user; the second is that the important criterion in the adoption of a book (an "influence threshold") is not the bare number of neighbors who have adopted a book, but the corresponding fraction among all out-neighbors, and that the influence threshold in such context is rather low.

As previously mentioned, users are notified of the adoption of a book by their out-neighbors: information flows therefore in an automated way along the friendship and neighborhood links. It is

that no absolute proof of influence effects can be obtained, as one cannot rule out effects external to the network.

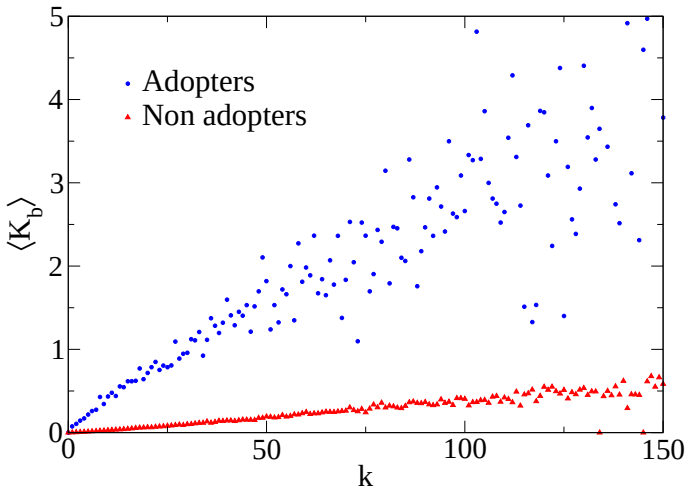


Figure 4.7: Number (K_b) of out-neighbors on the social network having book b at time t, averaged for the users having the same number (k) of out-neighbors. The cases for adopters and non-adopters are shown.

thus interesting to compare the potential existence of influence effects in the book adoption process along the social links that do not support additional (non automated) communication between the users ($Social \backslash Comm$) with respect to the case of social links that do ($Social \cap Comm$). To this aim, we compute the probability of adoption at time t of a book b given a fixed number of neighbors who already have b at time $t-1$, formally: $P_a(b,t|K_b)$ s.t. $K_b = |\Gamma_{out} \cap G^*(b,t-1)|$, where Γ_{out} is the set of out-neighbors of u.

The computation of P_a for the pure social network must use out-neighbors because the information (i.e., automatic notifications) flows contrariwise the direction of the edges. In the interaction network instead, both directions should be taken into account because a message sent from u to v may imply a particular interest of u in v's library or, conversely, that u is proactively suggesting a book to v. For this reason in the Interaction network we consider two separate cases where K_b is computed considering the set of in-neighbors Γ_{in} or out-neighbors Γ_{out}.

Figure 4.8 shows the values of $P_a(b,t|K_b)$ averaged over all books and time steps, for the pure social network ($Social \backslash Comm$) and the Interaction network ($Social \cap Comm$).

Interesting features emerge: (i) the probability of adoption is very small if $k = 0$ (less than $2 \cdot 10^{-4}$), and increases very rapidly as the number of out-neighbors having the considered book at $t-1$ increase.; (ii) this probability tends to saturate as k increases above 20, showing that additional increase in the number of out-neighbors reading the book do not increase the user's adoption probability; (iii) the probability of adoption at fixed number of out-neighbors reading the book is much larger for out-neighbors with whom an explicit communication is established; (iv) when focusing on interaction ties, receiving messages from a number of early adopters of a book b implies a higher probability of adoption of b than sending messages to the same number of owners of b.

The first result is a strong indication in favor of the hypothesis of effective influence between

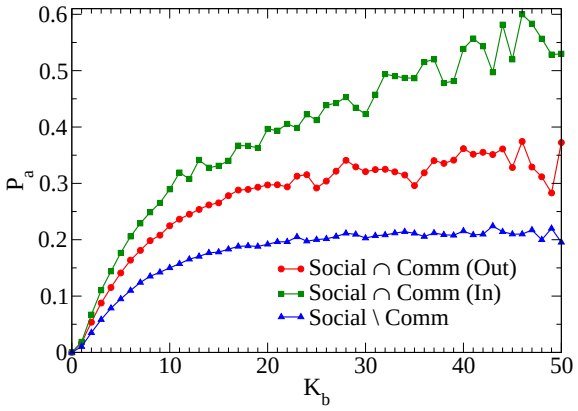


Figure 4.8: Probability of a book adoption averaged on all the books and snapshots, at fixed values of neighbors that are earlier adopters. Curves are depicted for the pure social network and for the interaction network.

neighbors on the social graph. The second indicates that the number of influential neighbors is limited, in support of the first hypothesis outlined above. The third result supports a scenario in which direct suggestions from neighbors with whom an explicit communication exist have a stronger influencing power than the automated notification system and, in particular, the fourth result suggests that adoption is often triggered by direct recommendations received by earlier adopters.

Chapter 5

Link prediction

5.1 Toward friendship forecast service

Studying the dynamics of complex systems is useful to understand and model the mechanisms at the basis of their evolution. Discovering the fundamental gears that move a complex social network allows to perform *predictions* on the future status of the system [17]. Even though in recent years the topic of prediction in social systems is becoming wider and tend to correlate several interconnected systems with different features and dimensions (see for example the work by Bollen et al. about the prediction of the stock market based on Twitter sentiment analysis [56]), the main efforts in this field have been spent to predict some generic features of a node or edge in a network [5] and, in particular, the presence of a link at some time in the future.

Predicting the presence of a link between two entities in a network is one of the major challenges in the area of *link mining* [120]. Such edge-related mining task is usually defined as *link detection* [90] when it aims to disclose the presence of unobserved or unknown links on a static network or as *link prediction* when it aims to foresee whether a connection will arise in the future between two nodes that are unlinked at the current time; we preserve this terminology in the following.

Seminal work on link prediction was presented by Liben-Nowell and Kleinberg [197, 199]. They identify structural properties of the graph which can be used to build a ranking of the node pairs based on their structural similarity, which is in turn exploited to predict future interactions in scientific collaboration networks. Results show a good accuracy over a random predictor. Several slight variants of this approach has been adopted (e.g., work by Pavlov and Ichise [262]). In contrast, another early work by Popescul et al. [263] focused on link detection using a classifier trained on the *feature vectors* that describe the nodes of the graph. The authors point out also the skew of the accuracy prediction due to the high sparsity of the link matrix, and perform a study on the alteration of the accuracy depending on the ratio between positive and negative samples in both training and test set.

Combining structural graph similarity measures and simple node features in a supervised learning approach to link prediction has been also tried in the past [142], showing the improvement of the prediction performance compared to predictors aware of the topological features only. Geo-

graphical proximity between nodes [250] and groups affiliation [362] have been effectively used as node features as well. Recently, some tests have been done also on the predictive power of some network clustering algorithms in the link prediction tasks [290]; however, reported results are not very encouraging.

The best-known topological measures of structural similarity between pairs of nodes are reviewed and refined by Zhou et al. [363] and Lü et al. [206]; the authors compare several structural similarity metrics for link prediction and detection, in terms of accuracy and computational efficiency. New local measures, namely the *resource allocation* and the *local path*, are proposed and showed to be efficient and accurate in link detection. Efficiency of structural proximity metrics on graphs is addressed also by Song et al. [310].

Detection of links based only on the information extracted from the folksonomies embedded in social media is performed by Schifanella et al. [299]; similarity measures explicitly designed for the three-dimensional folksonomic space are used to compute a lexical proximity between users and, consequently, to predict a connection between them. A similar context is considered by Leroy et al. [185], who leverage the group membership information from Flickr to build a probabilistic graph useful to detect the hidden social graph with a good accuracy.

The problem of detecting both unknown links and missing node attributes in a network is addressed by Bilgic et al. [53]. They propose an iterative method which refines at each step the prediction of one of the two features considered leveraging the information gained on the other feature at the previous step.

The role of temporal aspects in prediction is explored by Tylenda et al. [328], who exploit the information of recent interaction between individuals to improve the prediction accuracy. Dunlavy et al. [100] use a matrix-tensor method to predict links that will be created in the future. Their analysis is focused on networked systems with well-detectable period patterns in their underlying periodic structure.

Huan et al. [153] define a cycle formation model for social graphs that relates the probability of the presence of a link with its ability to form cycles. The parameters of the model are estimated using the generalized clustering coefficients of the network. The power of the model is evaluated on the Enron email dataset. Another probabilistic network evolution model aimed at link prediction is proposed by [165]. The idea is that links appear in the network due to a copying process where status labels associated to edges are copied from one node to another with a probability that is dependent on the relative topological position of the two nodes.

Prediction of future links in a question-answering bulletin board service is performed by [238], showing network proximity scores calculated from local topological information to be accurate predictors of future links.

Clauset et al. [83] present a hierarchical decomposition algorithm for network clustering which can also be applied to predict missing interactions in networks. The generated dendrograms determine the probability of connection for every pair of vertices. Links are predicted between pairs that have high probability of connection within the hierarchical random graphs but that are unconnected in the observed network. This technique yields positive results on several small networks.

Link prediction can also be based on features that describe user profiles, based on the principle that people with similar tastes are more likely to establish social contacts. Caragea et al. [71] propose an ontology-based classification of user features and show that the semantics captured by the ontology can effectively improve the performance of a topology-based machine learning classifier for social link prediction. Li et al. [194] propose a method to cluster Delicious users extracting implicit relations based on the similarity of their tag vocabulary.

In a survey of link prediction techniques, Lu et al. [208] compare several structural similarity metrics in terms of accuracy and computational efficiency.

Even if the majority of papers is focused on link prediction on simple (directed or undirected) graphs, a few techniques have been developed also for different kinds of networks. Work has been made in link detection on weighted networks [207, 121], bipartite networks [100, 47, 175] and signed social graphs [190]. Very recently, an approach that combines supervised learning and random walks has been shown to have a good accuracy for both prediction and recommendation of new links [32].

Finally, some approaches based on probabilistic models such as relational Markov networks [324] and probabilistic relational models [119] deserve to be cited. However, these approaches have not been proved to be scalable and they have not been extensively tested on real-world datasets.

5.2 Social link detection

The findings on homophily reported in Chapter 4 naturally lead us to the hypothesis that the presence of a social tie could be inferred relying only on the topical similarity between users. In this section, we test this hypothesis focusing on the Last.fm and aNobii datasets. We do so for several reasons. First, in these two cases, the data include information on the user libraries, in addition to groups and annotations. Second, a prediction based on the tagging information is more meaningful in a *broad* folksonomy, in which users can label the same (global) set of items and pick from this global set to form their libraries. This condition allows us to deal with similarity based on shared content as well as shared vocabulary. Last.fm and aNobii are broad folksonomies, as any user can tag any artist or book. This is not the case for Flickr, which is considered a narrow folksonomy, as users normally tag only the pictures they upload themselves.

Third, the Last.fm and aNobii datasets have peculiarities that allow to draw interesting conclusions regarding link prediction task. They both have a specification of users' mother tongues; as we shall see, language is a feature that can considerably affect prediction, and that should thus be taken into account when accuracy is measured. Additionally, Last.fm provides the *tasteometer* score, a user-to-user similarity metric computed by the system. While the tasteometer algorithm is not public, and therefore we do not have precise information of how its values are computed, we have verified empirically that it is largely independent from social information and it is based only on listening patterns. Consequently we can fairly compare the prediction accuracy achieved by user-to-user topical similarity measures to the one obtained by the system-provided similarity metric.

5.2.1 Methodology

In a first stage we focus on the task of *link detection*, that can be defined as follows. Given a sample of users $U_s \subseteq U$, we want to detect the presence or absence of a social link for every pair $(u, v) \in \{U_s \times U_s | u \neq v\}$. When detecting the link between a pair (u, v) we suppose to have all the information about the network topology but the (u, v) edge and about the *features* that describe the user profiles. We deal in particular with four different features: *groups*, *library*, *tags*, and tagged *items*. Note the difference between *items* and *library* features. In aNobii, tagged items are a subset of the whole set of books in the user's library, while in Last.fm tagged items can be tracks, albums or artists and the library is composed only of the top 50 artists in the user's global playlist. Tags and items can be directly extracted from the three-dimensional triple space through aggregation (details are provided in Section 5.2.2).

We take into account each feature separately, so each user is described with a single *feature vector*. For each pair of users in U_s, we compute a similarity value between their feature vectors using the metrics defined in Section 5.2.2. In the case of Last.fm we also have the system-provided tasteometer similarity. Next, we sort the node pairs in decreasing order of their similarity score. The pairs with the highest topical similarity are those that we suppose are the most likely to be connected with a social link. For this reason, we predict the presence of a tie for every user pair whose similarity value is greater than or equal to a threshold value σ. To evaluate the accuracy of our the link detection, we check the presence of each detected link in the social network and we count the number of true positives and false positives. As the value of σ decreases, a higher number of links is predicted, leading to an increase in the number of both true positives and false positives. We test the accuracy of our predictor for all the significant values of σ. The similarity measure that performs best for the prediction task is the one that achieves the best ratio between true positives and false positives, across all the possible threshold values. To quantitatively measure the prediction performance for the whole set of threshold values we consider ROC curves [108] and we compare the area under the curve (AUC) achieved by the different features and similarity metrics considered. ROC curves are commonly used in the machine learning community for link prediction [83].

Given this setting, it is important to select a significant sample of users U_s. Intuitively, one could choose $U_s \equiv U$. The problem is that, since the social graph's density is very low, the full social matrix $U \times U$ is extremely sparse, thus leading to a very low ratio of potential true positives.

The problem of studying greatly biased datasets is well known by data miners and it is a common issue also in social link prediction, due to the intrinsic sparsity of social graphs [198, 120, 190, 208]. It has been shown that the AUC is a good measure for performance evaluation when there is a strong class skew [314]. However, in the first part of our evaluation we want to minimize the sparsity problem in order to compare the predictive power of different features in a less biased setting.

We thus restrict our initial analysis to several smaller subsets, each composed of 500 users sampled on the basis of one of two criteria. First, we extracted a *Most Connected* set for each feature, composed of the nodes with the highest out-degree and that have at least one element for

the considered feature. Second, we sampled a distinct *Most Active* set for each feature, containing the 500 users with the largest number of elements for that feature. More in detail, we chose the sets of users with the highest number of groups, with the highest number of objects in their libraries, and, for both item and tag features, we chose the set of the 500 taggers with the highest number of triples.

The Most Active sampling provides the best scenario in which to explore the effectiveness of link prediction based on topical similarity. Furthermore, given the correlation between user activity and social connectivity (see Figure 3.5), the Most Active nodes typically have a rather high degree, thus ensuring a relevant number of intra-sample social connections. As a result, the density of our social network samples ranges from 0.02 to 0.07, which is three–four orders of magnitude higher than the full networks, whose order of magnitude is around 10^{-5}. Besides, to perform a test on the average case, we do not take into account simply a random set of users: we consider a set of high-degree nodes instead, thus circumventing the density problem.

In a second phase of the evaluation, we expand our observations with a sensitivity analysis to show how much the prediction accuracy is affected by the density, user activity, and size of the sampled subgraph, thus disentangling the evaluation from possible skew due to the narrower sample of active and connected users. Then, in Section 5.3 we will discuss the task of link prediction and contact recommendation.

5.2.2 Similarity Metrics

To model the task of predicting social links we need to define measures of profile similarity between users. In particular, we have to select a robust similarity metric for the features that characterize the activity of users. In relation to group membership and library features we follow the approach in Section 3.3.5 that computes similarities by way of the standard cosine similarity as formalized in Equation 3.6.

For the remaining features we adopt the framework by Markines et al. [215] that represents the system as a *tripartite* graph involving users, tags, and resources (e.g., books, songs, photos, etc.). A *triple* is a ternary relation between a user u, a tag t, and a resource r, and a set of triples is what we call a folksonomy F. We then define similarity measures $\sigma(u, v)$ where u and v can be two resources, tags, or users. Here we focus on similarity functions where u and v are two users.

Since measures for similarity and relatedness are not well developed for three-mode data such as folksonomies, we consider various ways to obtain two-mode views of the data. In particular, we consider two-mode views in which the two dimensions considered are dual — for example, users can be represented as sets of tags or resources. The process of obtaining a two-mode view from a folksonomy is called *aggregation*. Next we present different aggregation strategies and the set of similarity functions we adopted.

Aggregation Methods

In reducing the dimensionality of the triple space, we necessarily lose correlation information. Therefore, the aggregation method is critical for the design of effective similarity measures; poor aggregation choices may negatively affect the quality of the similarity by discarding informative correlations. Focusing on user similarity, we can aggregate across one of the tag or resource dimensions, obtaining a description of a user as a vector of, respectively, resources or tags. We consider four approaches to aggregate user information: *projection*, *distributional*, *macro*, and *collaborative* aggregation. To simplify our exposition, in the following definitions we will adopt an aggregation across resources, meaning that a user will be represented as a vector of tags; analogous mechanisms apply when tags are selected as the aggregation dimension. An extensive discussion on these aggregation approaches can be found in the work by Markines et al. [215].

Projection. The simplest aggregation approach corresponds to the projection operator $\pi_{u,t}(F)$ in relational algebra, assuming the triples are stored in a database relation F. Another way to represent the result of aggregation by simple projection is a matrix with binary elements where rows correspond to users (as binary vectors, or sets of tags) and columns corresponds to tags (as binary vectors, or sets of users).

Distributional. A more sophisticated form of aggregation stems from considering distributional information associated with the set membership relationships. One way to achieve distributional aggregation is to make set membership fuzzy, i.e., weighted by the Shannon information (log-odds) extracted from the annotations. Intuitively, a tag shared by two users may signal a weak association if it is very common. For example, let U be the set of users and U_t the users that annotate with t. We will use the information of tag t defined as $-\log p(t)$ where

$$p(t) = \frac{|U_t|}{|U|}. \tag{5.1}$$

Another approach is to define a set of *frequency-weighted* pairs (u, t, w_{ut}) where the weight w_{ut} is the number of resources tagged with t by u. Such a representation corresponds to a matrix with integer elements w_{ut}, where rows are user vectors and columns are tag vectors. We will use both of the above distributional aggregation approaches as appropriate for different similarity measures.

Macro. To compute an average function in class-partitioned datasets (e.g., documents partitioned into categories), micro- and macro-averaging approaches are possible. Micro-averaged scores are calculated considering the contribution from each element in each class. In contrast, macro-averaged values are obtained by first calculating the function for each class and then taking the average of the results. Micro-averaging gives equal weight to every element while macro-averaging gives equal weight to every class. Both approaches are broadly used in text mining [109]. By analogy, distributional aggregation can be viewed as *micro-aggregation* if we think of resources as classes. Each annotation is given the same weight, so that a more popular resource would have a larger impact on the weights and consequently on any derived similarity

measure. In contrast, macro-aggregation treats each resource's annotation set independently first, and then aggregates across resources. This will allow the similarity calculation to be *incremental*, breaking the dependency on global frequencies. In relational terms, we can select the triples involving each resource r, and then project, yielding a set of pairs for r: $\{(u,t)_r\} = \pi_{u,t}(\sigma_r(F))$. This results in per-resource binary matrices of the form $w_{r,ut}$. These matrix representations $w_{r,ut} \in \{0,1\}$ are used to compute a local similarity $\sigma_r(u,v)$ for each resource r. When defining the Shannon information of a feature, the feature probability $p(t)$ must be replaced by a conditional probability $p(t|r)$. Finally, we macro-aggregate by voting, i.e., by summing across resources to obtain the global similarity. Macro-aggregation does not have a bias toward resources with many annotations. However, in giving the same importance to each resource, the derived similarity measures amplify the relative impact of annotations of less popular resources.

Collaborative. Macro-aggregation lends itself to the exploration of collaborative filtering in folksonomies while the computation remains incremental. Thus far, we have only considered feature-based representations when working with a tripartite representation. That is, a user is described in terms of its tag or resource features. If two users share no feature, all of the measures defined on the basis of the aggregation schemes will yield a zero similarity. In collaborative filtering, on the other hand, the fact that one or more users vote for (or in our case annotate) two objects is seen as implicit evidence of an association between the two objects, even if they share no features. The more users share a pair of items, the stronger is the association. We want to consider the same idea in the context of user similarity in folksonomies. If many resources have been annotated by the same pair of users, even with different tags, the two users might be related. Likewise, if two users apply the same tags, even to annotate different resources, the two users might be related. We can capture this by adding a feature-independent local similarity to every pair (u,v) of users in macro-aggregation. In practice we can achieve this by adding a special "resource tag" t_r to all users that tagged r. This way all of r's users have at least one annotation in common. However, the information of such special tag would be $-\log(p(t_r|r)) = -\log(1) = 0$. To ensure that the special tag makes a non-zero contribution to the local similarity $\sigma_r(u,v)$, let us redefine the odds of tag t for resource r as

$$p(t|r) = \frac{|U_{t,r}|}{|U_r| + 1} \tag{5.2}$$

which is always less than 1 so that $-\log(p(t_r|r)) > 0$. Likewise, if many resources contain the same pair of tags, the two tags might be related even if they share no users.

Similarity Measures

We wish to explore several information-theoretic, statistical, and practical similarity measures. Each of the aggregation methods requires revisions and extensions of the definitions for application to the folksonomy context. Prior work [215] shown that distributional aggregation yields better accuracy than projection, and collaborative aggregation yields better accuracy than macro-

aggregation, with the same computational complexity, we focus on distributional and collaborative aggregations. For brevity, we show definitions only for the similarity measures in the distributional case. These definitions are based on feature probabilities $p(x)$ defined in Equation 5.1. The definitions of the local similarities for collaborative aggregation are similar except that the feature probabilities are replaced by the conditional probabilities defined in Equation 5.2. We suppose that $u, v \in U$ represent users and X_u, X_v are their vector representations. Of course, the attributes of X depend on the aggregation dimension. In the following formulas we consider the case of aggregation across resources, i.e., the users are denoted by vectors of tags with tag elements w_{ut}. Recalling that all measures are symmetric with respect to resources and tags, we simplify the notation as follows:

Matching. The distributional version of the matching similarity is

$$\sigma(u, v) = - \sum_{t \in X_u \cap X_v} \log p(t). \tag{5.3}$$

Overlap. Distributional overlap is given by

$$\sigma(u, v) = \frac{\sum_{t \in X_u \cap X_v} \log p(t)}{\max(\sum_{t \in X_u} \log p(t), \sum_{t \in X_v} \log p(t))}. \tag{5.4}$$

Jaccard. Distributional Jaccard similarity is defined as

$$\sigma(u, v) = \frac{\sum_{t \in X_u \cap X_v} \log p(t)}{\sum_{t \in X_u \cup X_v} \log p(t)}. \tag{5.5}$$

Dice. Distributional version of Dice is defined as

$$\sigma(u, v) = \frac{2 \sum_{t \in X_u \cap X_v} \log p(t)}{\sum_{t \in X_u} \log p(t) + \sum_{t \in X_v} \log p(t)}. \tag{5.6}$$

Cosine. For the distributional version of the cosine, it is natural to use the frequency-weighted representation

$$\sigma(u, v) = \frac{X_u}{||X_u||} \cdot \frac{X_v}{||X_v||} = \frac{\sum_t w_{ut} w_{vt}}{\sqrt{\sum_t w_{ut}^2} \sqrt{\sum_t w_{vt}^2}}. \tag{5.7}$$

This formula is equivalent to Equation 3.5.

Maximum Information Path. The last measure we consider is *Maximum Information Path* (MIP) [217]. The MIP similarity is an extension of traditional shortest-path based similarity measures [270] and Lin's similarity measure [202]. MIP differs from traditional shortest-path similarity measures by taking into account Shannon's information content of shared tags (or resources). Lin's similarity measure only applies to hierarchical taxonomies, such as the case when bookmarks are organized in folders and subfolders. However, when the folksonomy includes non-hierarchical annotations, Lin's measure is undefined while MIP similarity is well defined and captures the same intuition. The association between two objects is determined by the ratio between the maximum information they have in common (most informative shared feature) and the information they do not share. Because of the dependency on log-odds, maximum information is not defined for projection aggregation.

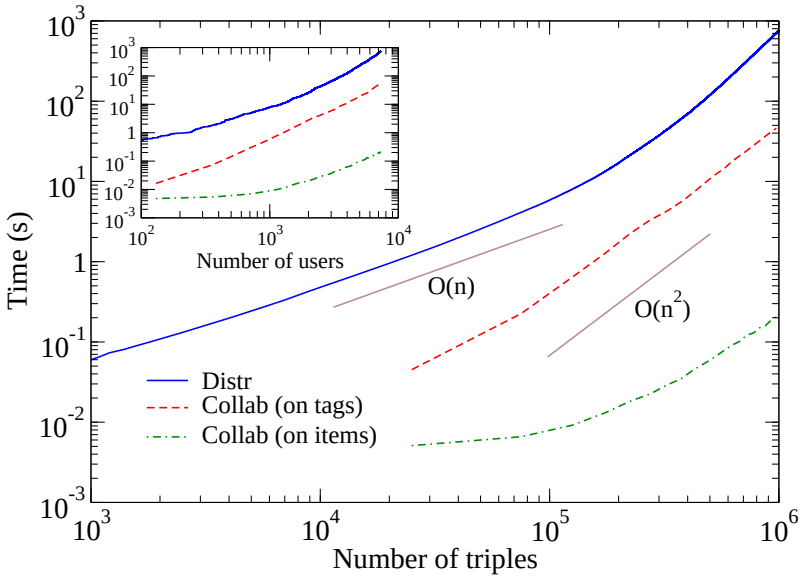


Figure 5.1: Scalability of the MIP similarity computation for distributional and collaborative aggregations. Computation time of collaborative aggregation is shown for two different aggregation dimensions (items and tags). The figure reports the CPU time (in seconds) against the number n of triples (and users, in the inset). Two guides to the eye representing $O(n)$ and $O(n^2)$ complexity are shown in gray.

We define MIP for the distributional case as

$$\sigma(u,v) = \frac{2\log(\min_{t\in X_u\cap X_v}[p(t)])}{\log(\min_{t\in X_u}[p(t)]) + \log(\min_{t\in X_v}[p(t)])}. \tag{5.8}$$

Prior work also explored mutual information and found it to be competitive but expensive [215], therefore we exclude it from the present analysis.

Computational Complexity

When computing similarity in large systems, the issue of scalability becomes crucial. The most important factor that affects scalability in the computation of the similarity matrix is the aggregation method adopted. In the following, we perform a computational complexity analysis focusing on distributional and collaborative aggregations, since they proved to be far more effective than other known aggregations [219]. The major difference between distributional and collaborative is *incrementality*. With distributional aggregation, similarities must be recomputed from scratch whenever new triples are added to the system, as frequency weights must be updated. Conversely, collaborative aggregation allows for incremental computation because each new triple affects only the contribution that the incoming tag or resource (depending on the aggregation dimension) gives to the overall similarity matrix.

From a practical point of view, we can consider as scalable those measures that can be updated as a stream of incoming annotations is received. However, since the update time clearly depends on how many user pairs' similarity scores are affected by the new triples, we should study how the

update time changes as the system size grows. We resort to an empirical analysis to examine how the update complexity scales with the number of users and triples in the system. Figure 5.1 shows the complexity for the two different flavors of collaborative aggregations and a single representative case for distributional aggregation, since in this case the aggregation dimension does not impact performance. The experiment is performed on the aNobii dataset, using the MIP similarity.

We observe that up to around 10^5 triples the computational time is roughly linear with the system size; when the size of the system grows further the time becomes quadratic. Nevertheless, the computational advantage in using the collaborative paradigm instead of the distributional one is evident. Moreover, it is clear that collaborative aggregation over items outperforms aggregation over tags; this result is basically due to the different distributions of items and tags over users. When a new triple is added, the contribution of the value of the aggregation dimension (tag or item) of that triple to the overall similarity matrix should be recomputed. The greater the number of users who have that tag or item in their triple sets, the higher the number of pairs whose similarity score must be updated. In the considered sample the number of resources in the triple set is one order of magnitude larger than the number of distinct tags. Similar ratios hold in any big folksonomy. For this reason, it is more likely that the addition of a triple containing a very popular tag will affect many more users (and consequently the similarity between them and others) than a triple with a very popular item.

5.2.3 Link detection with single features

We computed the similarity metrics on the Most Active and Most Connected sets of users for the four features considered, on the aNobii and Last.fm datasets. For the two folksonomy-related features, items and tags, we combined all the aggregation methods with all the similarity metrics defined above (except for the projection-MIP combination, which is not defined). For groups and library features we calculated the similarity using matching, overlap, Dice, Jaccard, and cosine metrics; note that these features do not require any aggregation and MIP is not defined for them.

We also considered two additional metrics as baselines. First, we queried the Last.fm API service for the tasteometer scores related to the same most active and Most Connected samples used for items and tags. Second, we computed the similarity in terms of number of common neighbors (CN) for the Most Connected samples, which, in this case, overlap with the Most Active samples since the number of connections is the feature considered. We introduce this widely used metric to compare the performance of network-based and feature-based similarities. Among all the known network-based metrics we opted for CN because, despite its simplicity, it has been shown to be an effective detector of social ties [241] and it is a local measure, whose computation is scalable.

Altogether we obtained 133 user similarity networks that we used to evaluate as many social link detections. For brevity, we next report only on a selection of representative cases, restricting our evaluation to the best-performing instances. AUC values are summarized in Table 5.1. Note that since the Last.fm library provided via the API has a size bounded to 50 artists, we cannot identify the Most Active users for this feature, therefore we omit the Most Active sample for the library feature. Not surprisingly, most of the highest AUC values are achieved for the Most Active

		Last.fm		aNobii	
Feature	Similarity	Active	Connected	Active	Connected
Baselines	Tasteometer	0.734	0.759	-	-
	Common neighbors	0.927	-	0.854	-
Items	Distrib cosine	0.663	0.560	**0.915**	**0.655**
	Distrib MIP	**0.749**	0.559	0.878	0.649
	Collab MIP	0.589	**0.613**	0.652	0.561
Tags	Distrib cosine	0.579	**0.625**	0.652	0.554
	Distrib MIP	0.697	0.618	0.651	0.560
	Collab MIP	**0.698**	0.559	**0.916**	**0.648**
Groups	Cosine	**0.810**	**0.677**	**0.662**	**0.690**
Library	Cosine	-	**0.769**	**0.923**	**0.768**

Table 5.1: AUC values for Last.fm and aNobii social link detections calculated for the four user features. The user samples considered are the most active with reference to the considered feature and the most connected users that have at least one element for that feature. The Last.fm results refer to one of our three snapshots; results for the other snapshots are consistent. The tasteometer similarity is calculated for the same most active and most connected sets used for items and tags. The feature vectors for items and tags are obtained through distributional or collaborative aggregation over the folksonomy. Shown in bold are the best results for each combination of dataset, sampling method, and feature.

samples, due to the greater amount of information available.

We note that the detection potential is affected by the aggregation process, but without a clearly interpretable pattern. Regarding the folksonomy-based features, we find that the MIP similarity often outperforms the cosine metric, as well as the other measures (not shown). For the Most Active Last.fm users represented through items, MIP similarity outperforms the tasteometer baseline.

Detections based on groups and libraries perform even better. For groups, we note a lower accuracy in the aNobii case compared to Last.fm, due to the relatively low cardinality of the group set; in fact, in aNobii we have about $3,000$ groups, against about $70,000$ groups in Last.fm. Inevitably, a lesser range of choice corresponds to a greater uniformity in the group affiliation behavior, thus making it more difficult to infer social connections. Lastly, the library results suggest that this is the best feature for detection purposes, when applicable. Since in Last.fm the library feature vectors have at most just 50 elements each, the implicit social information carried by the elements in the library is very high. When the cardinality of the feature vector is unbounded, like in the Most Active scenario in aNobii, the AUC values become even higher. Of course, for aNobii, we expected the library-based detection to be more accurate than the item-based ones, simply because the set of books in the library is a superset of the tagged books that can be retrieved from the folksonomy. Nevertheless, we observed that peak AUC values in aNobii are in part determined by the particularly strong geographically-biased clustering of its social network, which we discuss

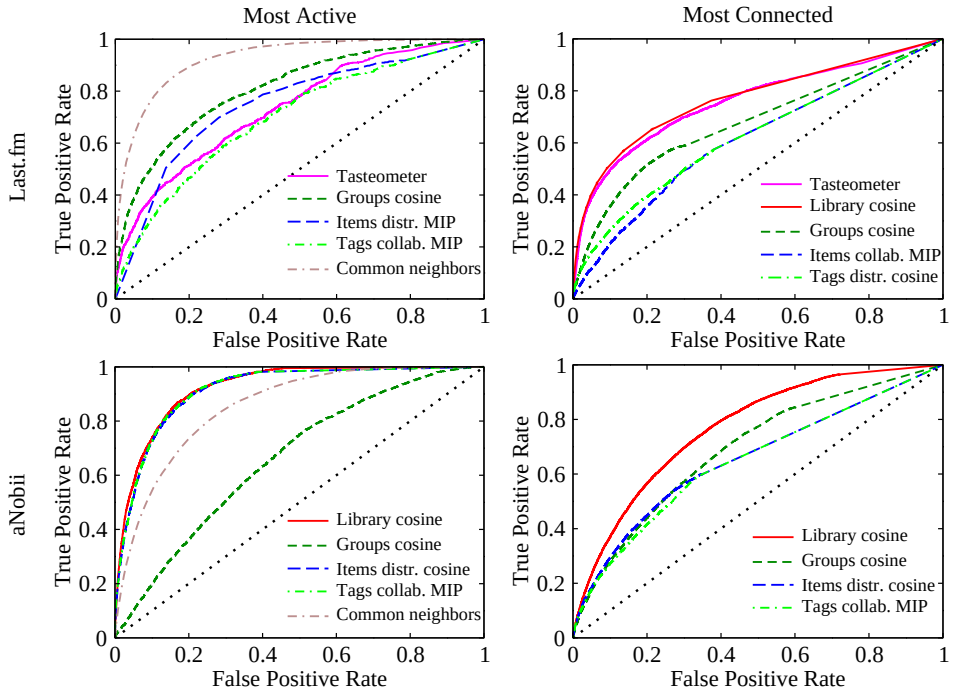


Figure 5.2: ROC curves comparing the accuracy for the best feature-based detections in both datasets. The tasteometer baseline curve is also shown for Last.fm.

in detail in Section 5.2.5. Finally, we observe that the detection based on common neighbors can be very accurate when a lot of information about social contacts is available. Indeed, in Last.fm's Top Active sample, the CN metric performs best, while in aNobii the library and folksonomic features are more accurate.

In Figure 5.2 we depict a summary comparison between the ROC curves of the best performing detection measures, i.e., those shown in bold in Table 5.1.

Sensitivity Analysis

The analysis on small user samples is useful for comparing the effectiveness of different metrics under different boundary conditions of connectivity and activity. However, to show that the results are not biased by this sampling procedure, we performed a sensitivity analysis for detections made on the aNobii dataset, using the library feature. (Similar results hold for other features.) We first looked at the effect of sample size on accuracy. Top plots in Figure 5.3 show the AUC values obtained for sample size up to 5000 users. Together with AUC, we also measure the precision at top N, i.e., what fraction of the pairs which have the N highest similarity values are actually connected; this is a metric which is commonly used to complement the AUC [32]. The detection accuracy is stable and does not depend significantly on sample size.

Second, to study how the detection accuracy depends on the activity and connectivity values of the samples, we collected several samples of the same size (500 users) but with decreasing activity (starting from the Most Active sample) and connectivity (starting from the Most Connected sample). Results are depicted in bottom plots of Figure 5.3. Since in many samples the number of

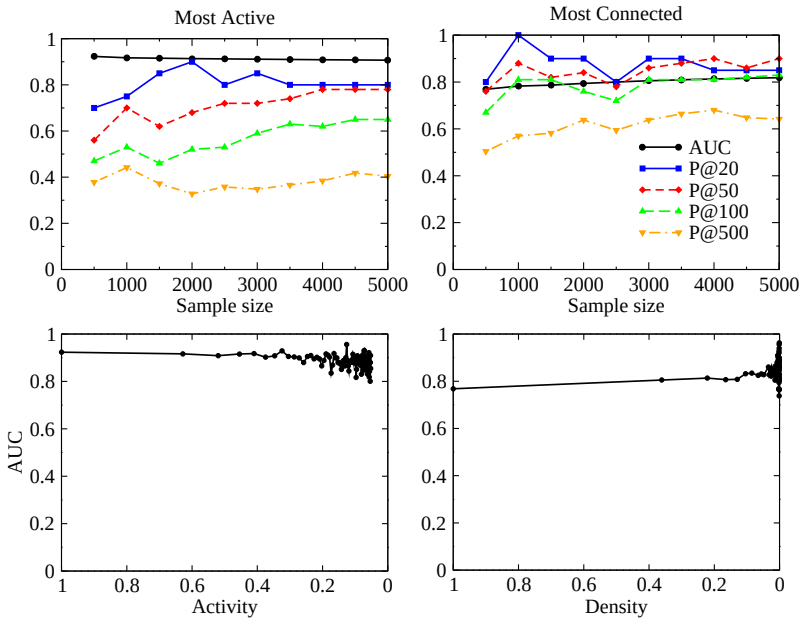


Figure 5.3: Sensitivity analysis of link detection based on the library feature in aNobii; the Most Active (left) and Most Connected (right) cases are considered. Top plots shows how AUC and precision at N change as the sample size is increased up to 5,000 users. Bottom plots show how AUC in different samples of size 500 but with decreasing values of density and activity (normalized on the highest value); the noisy patterns at the end of the lines are observed because the detection on very sparse samples is much more sensitive to slight variations in the connectivity patterns.

connected users is very low, and the classic definition of precision at top N is not meaningful when the number of links is less than N, we present only the AUC values. We observe that even when activity and connectivity are greatly reduced, the detection accuracy is stable and remains high.

Discussion

The overall picture that emerges from the experiments reveals some interesting results. First of all, the strong correlation between social linking and user activity, resulting in a noticeable homophily phenomenon, can be profitably exploited to accurately infer the structure of the social network given only information on user features. Considering various features results in quite different detection performance, as seen in Figure 5.2.

The only disadvantage of social link detection based on folksonomic information is that in many systems a considerable portion of users does not use tags: 50% of users are taggers in Last.fm, only 30% in aNobii. However, when tagging information is available, results are encouraging. For folksonomy-based features (items and tags), our detection methodology based on the MIP similarity metric tends to work better than the other similarity measures considered. This result holds across different collaborative tagging systems. When users are active (described by a high number of triples), we observe that good results are achieved with a distributional approach when aggregating over tags and with a collaborative approach when aggregating over items. In other

words, from a collaborative filtering perspective, knowing that two users share a tag is more informative for detecting their social link. In the distributional scenario with item representation, the accuracy compares favorably with that achieved by the Last.fm tasteometer, which is computed from a complete knowledge of the user profile.

It is interesting to notice that the ranking of aggregation methods by detection accuracy is not consistent across datasets. For example, in the Most Connected scenario, the distributional and the collaborative approaches behave differently in the two datasets considered (see Table 5.1). This means that even folksonomies with the same macro-structural properties (broad folksonomies, with similar numbers of users, triples and tags) can be characterized by inequivalent tagging patterns that lead to different performance of the detection techniques.

Detections made from the group feature can lead to even more accurate results. Groups behave very well if the total number of groups in the system is not too small with respect to the user population; furthermore, compared to the number of taggers, a larger portion of users in social systems take part in thematic groups, thus allowing the detection for a wider user set. Even if we do not focus here on the causality aspects linking social connections with homophily, the high AUC values obtained for the group feature could reasonably lead us to conjecture that groups are effective means of socialization, i.e., people know each other through groups.

The best performing profile feature is the library. Aside from the surprisingly high accuracy obtained for the Most Active set, where the information is maximal, the most important outcome is that the analysis of library feature vectors is very significant also in cases when considerably less information is available. From this viewpoint, the Most Connected scenario in Last.fm is particularly revealing because the accuracy is very high even if users are described with feature vectors containing at most 50 artists from their libraries. In a nutshell, the detection task performs best if it relies on the main feature that denotes the social network topic; in our case study, books for aNobii and artists for Last.fm.

Finally, the common neighbors baseline seems to be a very good detector of social links, sometimes performing even better than all other profile features. This is in part expected because the common neighbors measure captures the probability to form a *triadic closure*, which is a relevant phenomenon of attachment in social networks [241].

Next, we explore a hybrid approach that combines profile features and network-based features to achieve even better link detection accuracy.

5.2.4 Link detection with combined features

The common approach to link detection based on multiple features relies on machine learning techniques. Detection is seen as a binary classification problem that can be solved with a classifier trained on the features that describe the nodes. Positive and negative examples are chosen among pairs of nodes that are connected or disconnected, respectively [208].

Here we adopt this approach by selecting $10,000$ positive samples and as many negative samples from the set of aNobii users who have at least one instance of each feature in their profile and with at least one outgoing edge. The features considered are simply the similarity scores, computed as

described previously in this section. We use the J48 decision tree from WEKA [140] as a binary classifier and perform a 10-fold cross validation on our sample. We run the classifier for each profile feature and for the common neighbors separately, and then combine together all the profile features exclusively, before finally adding the information on common neighbors. We consider the AUC together with accuracy, False Positive and False Negative rates.

The results are shown in Table 5.2. The main observation is that combining different features results in a noticeable boost in detection power. In particular, the combination of different profile features gives about a 14% improvement in AUC over the best performing profile feature taken individually, while adding topological information, like the number of common neighbors, leads to an additional 4% improvement. Thus, even if link detection from network-based features can lead to accurate results, the best performance is achieved by their combination with profile features.

	Tags collab MIP	Groups	Library	CN	Profile feaures	All features
AUC	0.785	0.807	0.811	0.844	0.924	0.963
Accuracy	0.786	0.809	0.812	0.846	0.877	0.915
FP Rate	0.177	0.143	0.335	0.031	0.109	0.072
FN Rate	0.251	0.240	0.041	0.277	0.137	0.099

Table 5.2: Detection power of single and combined features using a decision tree on a balanced set of 10,000 positive and negative samples extracted from the aNobii dataset.

5.2.5 Language Community Analysis

The very high accuracy of our social link detection in aNobii motivated us to further inspect the reasons behind such a strong performance. We suspected that the results were somehow affected by the strongly clustered structure of the aNobii social network.

In fact, aNobii is split among two main groups: the Italian community (about 60% of users), and the Far East community, representing Hong Kong and Taiwan (about 20% of users). Since the type of literary items consumed by the great majority of people is strictly entangled with their mother tongue, the intersection of topical interests between the two communities is very small and prevalently limited to few worldwide best sellers. In addition to this, aNobii allows the insertion of annotations with any language character set. As a result, users fill their libraries with books written in their own language and they are motivated to annotate them using terms from their mother tongue vocabulary. Given the topical similarity between neighbors (see Section 3.3), and given that the two communities have very different topical interests, these two main groups turn out to be almost disconnected in the social network.

We show that this scenario directly affects the performance of our folksonomy-based detection method, considering as a representative example the Most Active sample of users. The same qualitative considerations hold for other samples and features. We split the Most Active set of taggers into clusters on a country level (within the set of top 500 taggers, 249 are Italian and 137 are Taiwanese) and we calculate some basic measures for pairs of users that reside in the same country or in different countries (we neglect users who do not specify a location in the profile).

	aNobii			Last.fm		
	Links (shuffl.)	Tags σ	Items σ	Links (shuffl.)	Tags σ	Items σ
Intra	85% (38%)	$3.4 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$	17% (9%)	$1.4 \cdot 10^{-1}$	$1.7 \cdot 10^{-2}$
Inter	15% (62%)	$4.7 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	83% (91%)	$1.4 \cdot 10^{-1}$	$1.5 \cdot 10^{-2}$

Table 5.3: Statistics on language communities that compose the Most Active users who declare country of origin in aNobii and Last.fm. We report on the portion of links that reside inside a cluster or, conversely, connect users belonging to different clusters in the real social network vs. its shuffled version (in parenthesis). The average cosine similarity for tag and item sets computed between pairs of users residing in the same or in different communities is reported as well.

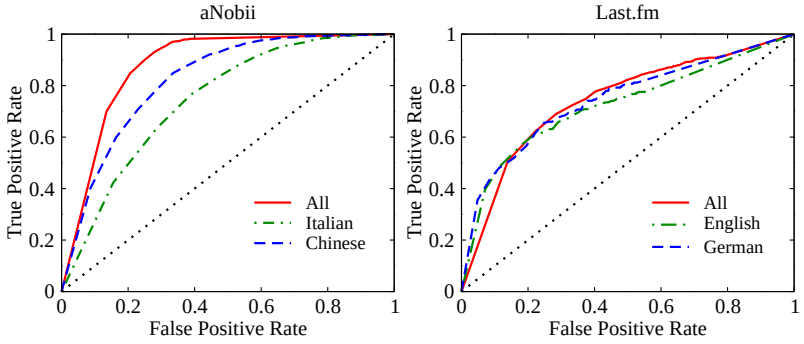


Figure 5.4: ROC curves comparing the link detection within different language communities in aNobii and Last.fm. The user samples considered are composed by the top 500 taggers in the whole system (All) or considering a single language community (Italian, Chinese, English, German). In all cases we used the MIP similarity metric using a distributional aggregation over tags.

For every user pair, we compute the cosine similarity between their vocabularies and between their item sets, and we measure the portion of inter- and intra-cluster links. To show that the portion of links residing inside a language community is not simply due to statistical properties caused by the size imbalance between different communities, we repeated the same measure on a shuffled version of the social graph, where each node keeps its out-degree but rewires its links at random. Statistics are summarized in Table 5.3. We notice that in aNobii the portion of inter-community links in the real network is considerably smaller than in the shuffled network, meaning that the clusters are nearly disconnected from each other due to language homophily. Furthermore, on average, the similarity between pairs of users, calculated for both tags and items, is much lower for users belonging to different language clusters compared to pairs of users that reside inside the same cluster. The average inter-community topical overlap is thus very small. Hence, the detection task is simpler with respect to a more homogeneous setting because both the social network and the folksonomy features are clustered according to language.

The effect of language on facilitating link detection can be verified by focusing on communities with homogeneous language. We performed the detection task on subsets of 500 active users within language communities. The comparison between the ROC curves is depicted in Figure 5.4 for the MIP metric, using the aggregation on tags (the result is qualitatively the same for the

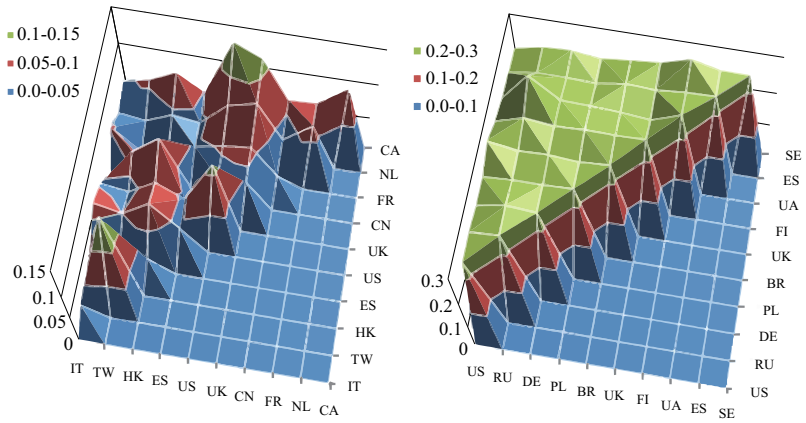


Figure 5.5: Cosine similarity between the tag vocabularies used by the 10 biggest geographic communities in aNobii (left) and Last.fm (right). A tag vocabulary is the set of tags obtained by merging all the tag sets of all the users in the community. Color intensity denotes different classes of similarity values.

other cases). The detections are significantly more accurate in the mixed community than in the homogeneous communities: the detection task is simpler in the former case because links between the two communities will almost never be detected, thus considerably decreasing the number of false positives. Language thus plays a key role in the detection task for multi-language communities.

To further confirm this observation, we performed the same tests on Last.fm, taking into account the most active users from the two largest language groups in our snapshot: the German community and the English-speaking community, composed by the union of users from USA, UK and Australia. As Figure 5.4 shows, in this case the detection accuracy is not clearly affected by the language. Such a different result is well explained by the statistics reported in Table 5.3. Compared to aNobii, the language communities are fuzzier in Last.fm since the level of language homophily is considerably lower and close to that of the random shuffled network; furthermore, there is no clear difference between inter- and intra-cluster feature similarities. Therefore, language does not play a substantial role in the link detection accuracy.

The impact of language clustering on the tagging behavior in the two social networks considered is also shown in the heat maps in Figure 5.5. Here, for both aNobii and Last.fm, we computed the cosine similarity between the tag vocabularies in use by the 10 most populated geographic communities. The results clearly show an overall low similarity value between the aNobii communities, except for those countries that share the same language (e.g., Canada, United States, and United Kingdom or Taiwan and Hong Kong). On the contrary, the Last.fm geographic groups are more homogeneous, with a substantially higher similarity and no clear distinction on the basis of their official languages. A very similar scenario is found for similarity on item sets (not shown).

Linguistic constraints are more tangible in aNobii than in Last.fm, or in books compared to music. The tagging behavior and the literary tastes of aNobii users are strongly affected by their language, while in Last.fm, users across different languages tend to have more tags and more music items in common. In synthesis, language can strongly impact the tagging behavior; the level of

homophily in a social network depends on the extent to which language is correlated with topical interests, which in turn depends on the nature of the objects shared between users.

5.3 Social link prediction and recommendation

In Section 5.2 we showed the potential of profile and topological features to detect links on the social network. Here we go a step further and we aim to solve the more challenging task of *link prediction*. In particular we use the acquired knowledge of network at time t to predict the creation of new links between t and $t + \Delta$. Prediction is intrinsically more difficult than detection because many features of the network can change in $[t, t + \Delta]$, and the prediction can be based only on the data up to time t. Conversely, in detection we have the complete information of network topology and users features up to the moment considered for link detection.

Moreover, we deal with link prediction from the even more challenging perspective of *contact recommendation*, that is a very hard task due to the extreme sparsity of the problem (see Section 5.3.2). We will focus on the analysis of the aNobii social network because of the temporal dimension we have in its dataset. However, the methodology for personalized contact recommendation that we propose could be directly implemented on any other social media.

5.3.1 Feature selection and training

In Section 5.2.4 we presented a machine learning approach to combine different features to the end of link detection and we showed the improvement that this combination gives over the detection with single features. Here we carry one the same general framework but we study more in depth the predictive power of a more exhaustive set of profile and topical features by measuring the accuracy gain that each feature gives to the overall prediction.

The features that we consider are determined by the three main evolutionary patterns of the social graph that we previously detected.

1. *Proximity-driven link creation.* In the vast majority of cases, new neighbors are chosen among the nodes at distance 2 (i.e., closing triangles) or 3 in the social graph. Restricting the analysis to pairs that reside near in the graph may miss some potential new connections but dramatically lowers the time of practical algorithms for partner recommendation.

2. *Strong interaction links.* Users are influenced and inspired more by the social contacts with whom they carry out a regular communication. Taking into account the strength of the interaction links rather than (or in addition to) pure social ties could improve the prediction.

3. *Homophily-driven attachment.* Users create new connections preferentially with their most similar acquaintances. Similarity is a notion that involves all the different facets of the user profile (from geographic location to favorite books).

A list of features that synthesizes these three points is shown in Table 5.4. Most of the topological features presented have been used independently in the literature to predict links in undirected

Feature	Description	Rank
Location	Binary attribute, whether u and v belong to the same city	14
Gender	Binary attribute, whether u and v belong to the same gender	15
Age	Absolute difference of ages	12
Library	Cosine similarity between library vectors	***5***
Groups	Cosine similarity between group membership vectors	***7***
Group size	Size of the smallest group the two users have in common	***6***
Vocabulary	Cosine similarity between sets of tags used	16
Contact list	Cosine similarity of the vectors of social contacts	***2***
Outdegree	Sum of the out degrees ($k_{out}(u) + k_{out}(v)$)	11
Preferential attachment	Product of the out degrees ($k_{out}(u) \cdot k_{out}(v)$)	13
Common neighbors	Number of common neighbors, directed case ($CN(u,v) = \lvert\Gamma_{out}(u) \cap \Gamma_{in}(v)\rvert$)	***4***
Triangle overlap	$\frac{CN(u,v)}{\Gamma_{out}(u)}$	***1***
Reciprocation	Binary attribute, whether the inverse link (v,u) is already present	***9***
Resource allocation	$\frac{1}{k_out(u)} \sum_{z \in (\Gamma_{out}(u) \cap \Gamma_{in}(u))} \left(\frac{1}{k_{out}(z)}\right)$ [363]	***3***
Local path	Linear combination of common neighbors and common distance-2 neighbors ($CN + \epsilon \cdot CN_2$) [363]	10
Weighted flow	$wf(u,v) = CN(u,v) + \frac{\sum_{x \in CN(u,v)} \min(w(u,x), w(x,v))}{CN(u,v)}$	***8***

Table 5.4: List of features used in the prediction of a directed link between generic users u and v, along with their description. $\Gamma_{in/out}(u)$ denotes the set of u's in/out neighbors, $k_{out}(u) = |\Gamma_{out}(u)|$, and $w(x,y)$ is the weight of the tie between x and y. The rank reported is the result of the Chi Squared attribute selection method applied to our test set; the bold font of the rank indicates that the corresponding feature has been selected for the restricted feature set.

networks but can be easily adapted to the directed case. To take into account as well the information about the weighted interaction network, we introduce a new index, the *weighted flow*, inspired by previous work on generalized degree centrality in social networks [252]. It is defined as:

$$wf(u,v) = CN(u,v) + \frac{\sum_{x \in CN(u,v)} \min(w(u,x), w(x,v))}{CN(u,v)}. \tag{5.9}$$

Assuming that weights on arcs denote some information flow passing between nodes, weighted flow combines the definition of common neighbors with the normalized sum of the minimum flow of information passing from the arcs connecting the two target nodes through their common neighbors. Applied to the interaction network, this metric measures both the number of potential communication channels between the two nodes and the amount of information that could have been possibly exchanged between them using their directed common neighbors as proxies.

All the features can be combined to the end of prediction through a supervised machine learning approach using WEKA [140]. Among all WEKA's classifiers we pick the best performing Rotation Tree classifier [281] trained with all the available features to detect whether a pair of users will be connected in the future or not. The positive sample of the training set is built by about $10k$ pairs of all the users that reside at distance 2 on the social graph at snapshot 1 and get connected before snapshot 6. The negative sample is given by as many pairs residing 2 hops away at snapshot 1 and that do not become connected. We focus on distance-2 neighbors because in the link recommendation task we will restrict our prediction to the closest disconnected neighbors for computational efficiency reasons. Note that taking into account only distance-2 pairs makes the prediction task harder than selecting the connected and disconnected pairs at random; this is because, due to the topical alignment described in Section 3.3.5, pairs of users at distance 2 along the social graph, even if they never become linked, have a much larger similarity than randomly chosen pairs of users.

The performance results of a 10-fold cross validation on the training set are given in Table 5.5. The combination of structural and profile features leads to an appreciable improvement of the precision of the prediction, for all the performance indexes considered. However, among all the features, some have weaker prediction power and thus can be neglected without affecting the overall performance. In particular, the Chi Squared analysis for feature selection [204] indicates a ranking of predictive potential (see Table 5.4) in which features like vocabulary, gender, and preferential attachment have much less relevance than others like the contact list or the library. In particular, we notice that features based on the triangle closure phenomenon are the most predictive. Table 5.5 shows that the prediction accuracy remains very stable when using only the top 9 features, and we therefore use this restricted set of features in the following.

5.3.2 Contact recommendation

A contact recommendation service should be able to serve suggestions in real-time and on demand. Screening all the users that are not connected with the client requires a too high computational effort to meet this requirement. Therefore, we adopt a local search limited to the distance-2

Features	FP rate	FN rate	Accuracy	F-value	AUC
Profile	0.279	0.364	0.679	0.678	0.741
Structural	0.241	0.298	0.730	0.730	0.805
All	0.223	0.264	0.757	0.757	0.835
Restricted	0.219	0.279	0.751	0.751	0.826

Table 5.5: Prediction performance on the training set using the Rotation Forest classifier, 10-fold cross validation, and four different combinations of features. Positive and negative samples are balanced in size.

neighborhood of the target user; among those potential contacts, the system outputs a fixed number N of suggestions.

To evaluate the effectiveness of this approach we build a test set of active users who establish at least 20 new social ties between snapshots 1 and 6 with other users that reside at distance 2 at snapshot 1. For each active user u, we apply our classifier to every pair $(u, v)|d(u, v) = 2$ and, from the set of positively labeled pairs, we select N contacts to compose the recommendation list. The list is sorted according to the confidence score given by the classifier for each prediction.

The number of actual ties created after time 1 by the sampled users is around $3k$, against more than $650k$ potential users residing at distance 2 of these users in the first snapshot. Such very high sparsity in the data (density is less than 0.005) makes the recommendation task particularly hard. Figure 5.6 depicts the variation of the precision with the size of the recommendation list. As baseline, we report the precision of two other predictions performed in an unsupervised way, by sorting the recommendation lists according to the number of common neighbors or to the cosine similarity between libraries.

Another attempt of tackling the link prediction problem from a recommendation perspective has been made in the Facebook social network [32]. The evaluation of the recommendation is very similar to ours with respect to the size of the network sample, the time span of the prediction and the activity of the target users. Among all the experiments that the authors report, recommendation through logistic regression combining several structural graph features compares well to our approach. Nevertheless the precision is much higher than in the aNobii case (precision at 20 is around 7.50 against 1.50 in the present case). The main reason is due to the different sparsity of the problem. Specifically:

- In the same time span, the average number of new contacts per user in Facebook is more than six times higher than in aNobii (26 new links in Facebook vs. 4 in aNobii).
- the portion of new contacts residing at distance >2 in aNobii is around 0.4, while in the Facebook dataset it is negligible.
- in contrast to the Facebook social network, the aNobii network is directed and the predictions must take into account the directionality of the edge.

In a nutshell, in Facebook users are much more active and faster in establishing new contacts and they focus much more on their distance-2 neighbors, thus increasing the number of potential

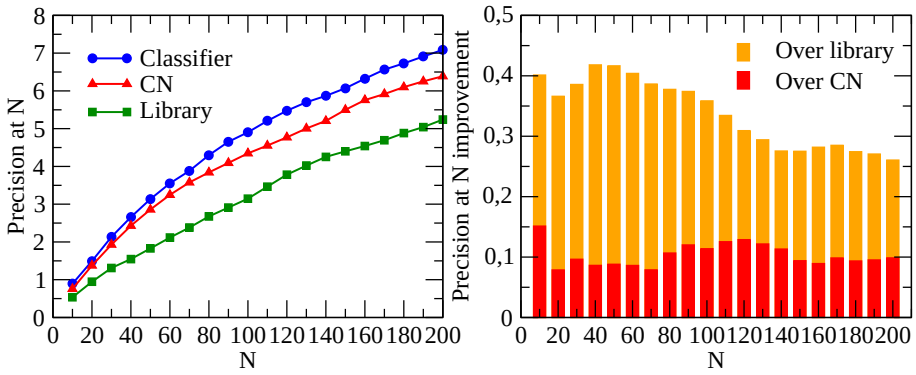


Figure 5.6: Recommendation results. (Left) Precision at N for the recommendation made with the classifier combining together all the relevant features and for two unsupervised baselines (common neighbors and library similarity). (Right) Relative improvement on the classifier-based approach over the baselines.

true positives over the total number of potential new contacts. Nevertheless, we underline that even in aNobii's more challenging setting the relative improvement of machine learning combination of different profile and structural features over the performance of common neighbors is comparable to the improvement obtained in the case of Facebook by previous work.

Chapter 6

Topics, profiling, and activity prediction in search systems

6.1 Mining information from the Web

Information overload is an issue that extends well beyond the boundaries of online social media. With the explosive growth of the World Wide Web [154] the number of multimedia resources that any user can browse became higher than any possible expectation. Effective information retrieval from the Web became one of the major issues in data mining [137, 131] and since the earlier stages of modern search engines [255] a lot of effort have been spent in enhancing retrieval procedures.

Methods and algorithms for improving web search have been extensively studied in the last two decades. In the vast majority of the cases, such methodologies are query-centric, i.e., they exploit only the query itself to understand a user's intent and to provide relevant results. Only recently user search activity has been studied by looking at the entire set of actions she performs in order to satisfy a need. Following this direction, previous work on query log mining has introduced general and widely used terms to define different structural features of the query log and behavioral aspects of the search engine users. Typically, sequences of query reformulations aimed to achieve the same atomic search need are referred as *query chains* [272]. More relaxed notions of coherence within the search session are represented by the concept of *logical session* [34] and *search mission* [162], namely a set of queries that express a complex search need, possibly articulated in smaller goals. Empirical studies have indeed shown that most of the needs are actually too complex to be satisfied by just one query [95]. Hence users are inclined to organize their search activity in missions.

By way of example, let's consider the activity of a user who wants to purchase a vacuum cleaner. She will probably organize her queries in a number of sequential steps. Initially she starts acquiring information about available brands and models. In a second phase she will look for reviews and comparisons between different models with similar features. Finally, she will search for sellers who offer the chosen model at an advantageous price or with an extended guarantee. Each single step is a different granular task and the totality of tasks are meant to fulfill the "vacuum-cleaner

purchase" need. In a similar scenario, an unsatisfied user who wants to return a vacuum cleaner is going to submit queries such as "returning shark navigator" or "dyson customer service." The two needs in this example concern particular aspects related to the same general topic "vacuum cleaner." We use the term *topic* in its acceptation of *mental object* or *cognitive content*, i.e., *the sum of what can be perceived, discovered or learned* about any real or abstract entity. In such a sense, topics naturally emerge from user search activity, since queries are issued to discover or learn facets (model, name, characteristics, pros and cons, price, seller's location, return policy) of a cognitive content (vacuum cleaner).

In this Chapter we present a contribution to reduce information overload in the Web search context by the means of the aggregation of the data that describe the behavior and the interests of the search engine users. We first describe a methodology to extract topics from query logs. Our objective is inherently different from previous attempts to classify queries according to a predefined set of categories, because our definition of topic encompasses and outstrips the definition of category. Categories like shopping, sport, news, and finance can indeed be seen as the perspective or focus of a search mission in the context of a particular topic.

Even if closely related to the document *topic extraction* task [293] or to multi-document summarization [36], where items from a textual corpus are summarized in one or more, possibly hierarchical [127] categories, our work gives a contribution to the *query clustering* area. Currently, query clustering is one of the hot topics in query log mining [44].

Grouping together queries with strong semantic relations is a task that is intrinsically harder than classic topic extraction or web document clustering [360], because of the short textual information contained into queries. Many approaches to query clustering rely on the computation of some notion of similarity between query pairs. When dealing with *query classification*, where the semantic categories are defined a-priori, it may be sufficient to compute the similarity based only on textual features to obtain good classification results [46]. Even classification based only on the so-called *clickthrough* data such as the information inside the result pages associated to the queries, can lead to good results [144]. However, if predefined concept categorizations or taxonomies are not available, lexical and content-based information taken separately are not sufficient to obtain good clusters. In this regard, an attempt to cluster queries from the Encarta user logs [342] showed that query-to-query similarity metrics that linearly combine textual features with click-through data can be used much more profitably in query clustering than single-attribute similarities. Similarly, hierarchical agglomeration of queries based on the similarity of their search result snippets (that mix words from the query and text from the page results) has also been profitably used [81].

Approaches focused only on the activity of single users, instead of the whole query log, are also interesting; in fact, it is clear that detecting the topics that can well shape the interests of the user is useful for personalization and recommendation. For instance, Song et al. [311] proposed a topic model able to extract most relevant themes from the user activity through probabilistic Latent Semantic Indexing [149]; once identified, such themes are used as a topical summarization of the user search history.

More recently, content-agnostic approaches based only on the relations between queries and

clicked URLs have been explored. The idea is that similarity graphs between queries obtained by proper projections on the multiple dimensions of the query log accurately model semantic relations between queries [35, 60]. Many approaches of this kind represent the query-URL relation as a bipartite graph [44] and pick the densely connected components, like bicliques [356], as representative sets of similar queries or pages. *Query graphs* where queries are connected by some lexical or semantic relation are also commonly used. In order to overcome the problem of query ambiguity in topic detection, query graphs that introduce a lightweight query contextualization using the pairs $(sessionId, query)$ as nodes, instead of single queries, have also been proposed [333].

Our primary objective is to define a methodology to aggregate different missions within the same cognitive content. This technique should be effective at different scales. In a individual perspective, it should be able to aggregate different but related missions of the same user, while in a wider context it can be used as a tool to cluster together all the missions that are topically coherent, across users. The method must not rely on predefined categorizations or taxonomies of user intent or topics.

For this purpose, we propose a mission-based clustering technique to aggregate missions that are topically-related. As a first step we train a classifier to predict if two different missions have a similar topical connotation. The learned function takes as input two sets of queries and computes the probability that they are topically-related. Such a function is used as a similarity metric by an agglomerative algorithm to merge missions into large topical clusters.

The resulting topics would be useful in a number of different scenarios where great advantage can be derived by performing query categorization in a more natural, intent-driven fashion, without any constraint imposed by artificial categories. As an application of the methodology, we show how to build user profiles over topics and we use such profiles to predict user behavior.

Before defining our methodology in detail, we introduce some basic terms that will be widely used in the following.

Query log. In a search engine context, search activity is typically recorded in a query log $\mathscr{L}$ defined as a set of tuples $\tau = \langle q, u, t, V, C \rangle$ containing the submitted query $q \in \mathscr{Q}$, an anonymized user identifier $u \in \mathscr{U}$, the time t when the action took place, the set of documents V returned by the search engine, and the set of clicked documents $C \subseteq V$ [55].

Missions. According to the original definition of Jones at al. [161], a *search mission is a related set of information needs, resulting in one or more goals.* In the example presented in the Introduction, "Purchasing a vacuum cleaner" is a mission that represents an intent that a user wants to satisfy. The three steps ("looking for models", "models comparison," and "sellers comparison") are the three sub-tasks (or goals) contained in the mission. All the queries in a missions have a strong topical coherence, which means that all of them are issued with a main common objective. It has been observed [95] that search activities that take place in complex domains like "travel" or "health" often require several queries before complex user intents are fully satisfied.

Topics. Missions are characterized by a main objective and one or more sub-tasks related to the objective itself. For example, a mission devoted to organize a trip, has the travel itself as main objective and a number of functional sub-tasks (like booking the flight, reserving the hotel, finding

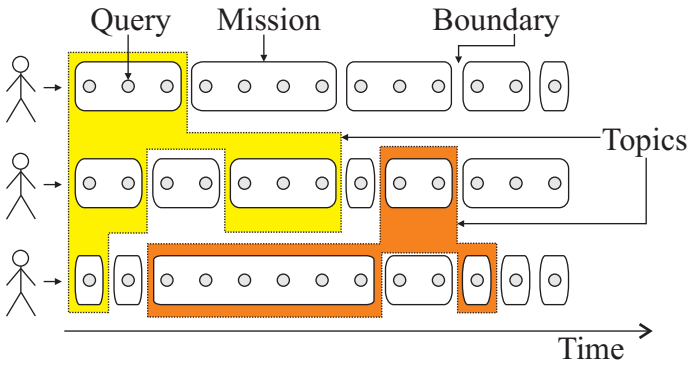


Figure 6.1: The activity of three search engine users partitioned at different levels. Every stream of user queries can be articulated in different missions that are aimed to satisfy a particular user intent. At a coarser level, topics can include missions from different users and also portions of activity originated by the same user at different times.

a guided tour). Travel missions generated by different users are all characterized by the same main objective regardless the destination, the temporal order in which the sub-tasks are issued or even the recreational activities booked. In such a sense all the missions devoted to organize a travel can be seen as part of a same topic or *cognitive content*.

Missions within the same cognitive content are meant to fulfill one or more intents related to such a content. Here on, we use the term *topic* to define a *cognitive content that includes the sum of what can be perceived, discovered or learned about any real or abstract entity.* From an operational point of view, this means that a topic can be seen as the aggregation of all missions with the same cognitive content generated over time across different users. A sketch depicting the relation between queries, missions and topics can be found in Figure 6.1.

6.2 Behavior-driven clustering of queries into topics

As we pointed out, the concepts of mission and topic are strictly related to each other. In fact, sequences of queries that express coherently a single and well defined user intent must have a high degree of topical coherence. This strong connection allows us to use missions as fundamental building blocks for topics: distinct missions can be merged together if their semantic connotation is very similar. In the following, we first summarize the state of the art technique that we use to detect missions from the query log; then we describe a method that is able to effectively combine together pairs of topically-related missions.

6.2.1 Detection of search missions

To partition the user activity into missions we use the machine learning approach proposed by Donato et al. [95]. This method is able to detect the boundaries of a mission by analyzing the live stream of actions performed by the user on the search engine. This approach relies on a module based on a gradient boosted decision tree classifier [347], called the *mission detector*, that works

at the level of query pairs. Given a set of features extracted from a pair of consecutive query log tuples τ_1, τ_2 generated by the same user, the mission detector indicates whether τ_2 is coherent with τ_1, from a topical perspective. When two queries are found to be incoherent, then a mission boundary is placed, so that the query log $\mathscr{L}$ is partitioned into missions containing one or more tuples.

The features used for the classifications come from three different domains: the *textual features*, that include different flavors of lexical similarity between the two queries, the *session features*, that measure several aspects of the click activity of the user in the time between the two queries and in the overall session, and the *time-related features* that take into account the inter-event time distance for some representative user actions. Using all of these feature together, the mission detector is able to reach a 95% accuracy in detecting boundaries on real user datastreams [95].

It has to be noted that missions identified by this method are semantically much narrower than topics, because queries in the same missions are not only constrained to be submitted by the same user, but they are also consecutive in time. Indeed, while a mission in theory can be fragmented in time, the mission detector by definition can only aggregate consecutive queries and, in practice, generates short-lived missions. Thus, the topical coherence constraints imposed on missions are much stronger than those that we require to be applied to topics.

6.2.2 Merging missions

Given the state of the art of mission boundary detection, it is possible to segment the user activity of every query log into missions. Furthermore, the strong topical coherence of queries inside the same mission can be exploited to generalize the approach used for mission boundary to a topic boundary detection. The idea is to use a new classifier, the *topic detector*, trained in semi-supervised fashion based on the data generated by the missions detector, to decide whether two query sets belong to the same topic. Its scheme is sketched in Figure 6.2.

Specifically, positive examples are automatically built by splitting missions in two consecutive query sequences and considering such two sequences as separate (sub)missions belonging to the same topic. Conversely, negative examples are formed by sets of queries belonging to consecutive missions of the same user, since we know they are topically unrelated because they are separated by the boundary placed by the mission detector. The topic detector is implemented with a Stochastic Gradient Boosted Decision Tree (GBDT) [353]. GBDT outputs the probability that the given sample is from the positive class; applied to our case, this is the probability that the two missions in input belong to the same topic. We can interpret this probability as a similarity score in $[0, 1]$ measuring the topical relatedness of the two sets and, consequently, the classifier can be seen as a *topical similarity function* $\mathscr{S}$.

The features given in input to the classifier are aggregated values over features computed from all the query pairs across two missions. Namely, given a (positive or negative) pair of missions m_1, m_2, all query pairs $q_1, q_2 | q_1 \in m_1 \wedge q_2 \in m_2$ are taken into account. Then, all the values of each feature are aggregated over all the pairs yielding four scores representing the average,

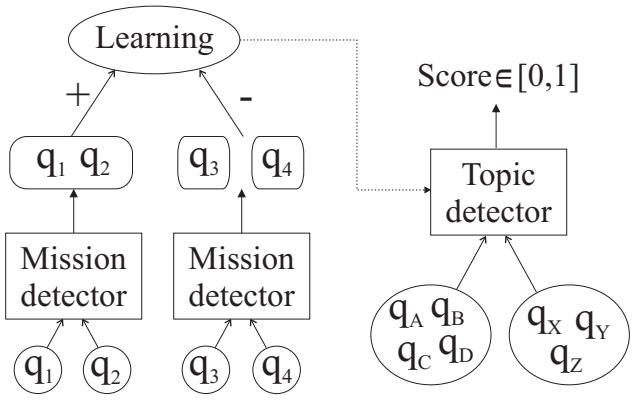


Figure 6.2: Training and application of the topic detector. Positive and negative examples for the learning phase are respectively missions and pairs of consecutive missions that are detected by the mission detector. Once trained, the topic detector can take in input any pair of query sets and compute a confidence score that can be interpreted as a topical similarity between the two sets.

standard deviation, minimum, and maximum values for that feature. For each query pair, features from three different categories are extracted[1]:

- **Lexical features**. Very often, similarity between the text of different queries denotes a strong semantic relation (e.g., "paris cheap travel" and "travelling to paris"). For this reason, we train the classifier using several lexical features such as length of common prefix and suffix, size of the intersection, edit distance, several similarity measures computed at word and character 3-grams level, and many others.

- **Behavioral features**. The behavior of users during the search activity gives much implicit information on the semantic relatedness of queries. For instance, if a user submits two queries in close succession, it is likely that the two queries are very related to each other, based on the assumption that the user activity is *bursty* [40] and events happening in the same burst are meant to accomplish the same task. However, since user behavior is very heterogeneous, it is necessary to aggregate behavioral information from several user sessions. We compute the average values of the behavioral features over a year of query log for each query pair q_1, q_2 such that q_2 has been observed at least once right after q_1 in the log.[2] The average time and the average number of clicks between two queries are examples of behavioral features.

- **Search result features**. Intuitively, the page result sets returned for a pair of topically-related queries will be topically related as well, to some extent. Thus, we consider a bunch of result-related features such as the intersection between result sets and the similarity between the vectors of the K most frequent words from a given content dictionary [24] appearing in the N top results.

[1] We do not report the complete list of features since they are commonly used in session analysis [162, 55].

[2] Behavioral features are defined only for successive query pairs observed at least 2 times. Else, a default value is used.

Algorithm 1: Iterative topic extraction

Require: Initial set of seed topics $\mathscr{T}_0$; similarity threshold $\theta \in (0,1)$; termination threshold $\alpha \in (0,1)$; topic similarity function $\mathscr{S} : \mathscr{T} \times \mathscr{T} \rightarrow [0,1]$

1: $\mathscr{T}_{i+1} = \mathscr{T}_0$
2: **repeat**
3: $\quad \mathscr{T}_i = \mathscr{T}_{i+1} - T_0$; $\mathscr{T}_{i+1} = T_0$
4: $\quad$ **for** $T_1 \in \mathscr{T}_i$ **do**
5: $\quad\quad T_x = T_2 \in \mathscr{T}_{i+1} | \mathscr{S}(T_1, T_2) \geq \theta \wedge \mathscr{S}(T_1, T_2) \geq \mathscr{S}(T_1, T), \forall T \in \mathscr{T}_{i+1}$
6: $\quad\quad$ **if** a valid T_x has been found **then**
7: $\quad\quad\quad \mathscr{T}_{i+1} = \mathscr{T}_{i+1} - T_x + (T_x \cup T_1)$
8: $\quad\quad$ **else**
9: $\quad\quad\quad \mathscr{T}_{i+1} = \mathscr{T}_{i+1} + T_1$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: **until** $\frac{|\mathscr{T}_i+1|}{|\mathscr{T}_i|} \leq \alpha$

We trained our topic detector using a balanced sample of 500K mission pairs extracted from random user sessions over the 2010 query log of the Yahoo! search engine. Results of 10-fold cross validation give an AUC value of 0.95, thus confirming that the topic detector is able to accurately discriminate between sets of queries that come from the same topic and those that do not.

One may think that the mission detector could have been used as-is to extract topics by means of detecting pairs of topically coherent queries among all the possible query pairs and iteratively clustering them. However, this approach has two main flaws. First, the computational effort to classify every possible query pair in $\mathscr{Q}$ is prohibitive given the dimension of real search query logs. Second, the mission detector is trained given query pairs that are adjacent realizations of the same user activity stream, while in topic extraction we are interested in comparing queries of different users, possibly in different times. On the other hand, classifying pairs of query sets allows us to leverage the mission data as an already available source. In any case, even if it were possible to classify every query pair, the clustering step would still require similarity computation for query sets.

6.2.3 Greedy agglomerative topic extraction

The topic similarity function $\mathscr{S}$ can be used iteratively to extract topics from the mission data. Consider a set $\mathscr{T}_0$ of *seed topics* where each topic contains only the set of queries submitted during a single user mission. This set is a very strict partition of the query corpus, where each partition is not only a coherent topic but it is the expression of an intent of a single user.

Our topic extraction algorithm is in the spirit of classic hierarchical agglomerative clustering [347]. As shown in the Algorithm 1 pseudo-code, at any iteration i, the two sets $\mathscr{T}_i$ and $\mathscr{T}_{i+1}$ are considered. Initially, $\mathscr{T}_{i+1} = \{T_0\}$ and $\mathscr{T}_i = \{\mathscr{T}_0 - T_0\}$.

Each topic $T \in \mathscr{T}_i$ is compared with all topics $T_x \in \mathscr{T}_{i+1}$ through the function $\mathscr{S}(T, T_x)$; if

topic similarity is below a certain threshold θ for all pairs, then T is moved from $\mathscr{T}_i$ to $\mathscr{T}_{i+1}$. Otherwise, the pair (T, T_x) with the highest score is greedily selected, T is removed from $\mathscr{T}_i$ and its elements are added to T_x, creating a broader topic. The algorithm is iterated as long as the relative decrease in the number of topics is large, or until $\frac{|\mathscr{T}_i+1|}{|\mathscr{T}_i|} > \alpha \in (0,1)$.

The complexity of a single iteration, in terms of number of computations of the $\mathscr{S}$ function, is $O(|\mathscr{T}_i|^2)$ and $\Omega(|\mathscr{T}_i|)$, since $\frac{|\mathscr{T}_i|^2-|\mathscr{T}_i|}{2}$ computations of the topic similarity are needed if no topics are merged, while just $|\mathscr{T}_i|$ comparisons take place if every topic merges into a single supertopic. The number of iterations needed depends on the stop condition α, but it is always bounded by $O(log(|\mathscr{T}_0|))$, thus leading to an overall algorithm complexity of $O(|\mathscr{T}_0|^2 \cdot log(|\mathscr{T}_0|))$.

Even if the number of iterations required is considerably smaller than the theoretical upper bound, in practice the quadratic complexity of computing $\mathscr{S}$ for every topic pair is still too costly for large query logs. However, the efficiency can be improved through a heuristic approach based on the observation that a very small portion of all possible topic pairs are actually merged in each iteration. To reduce the number of comparisons, the topic set $\mathscr{T}_i$ is partitioned into several smaller sets that are given in input to independent instances of Algorithm 1, so that the $\mathscr{S}$ function is applied only between elements inside the same partition and not across partitions. In addition to reducing the number of topic pairs considered, this approach enables a parallel implementation of the clustering algorithm, thus dramatically decreasing the actual computation time, even though the theoretical complexity remains the same. For brevity, in the following we will refer to our Greedy Agglomerative Topic Extraction algorithm using the acronym *GATE*.

The choice of a good *partitioning criterion* is crucial for the outcome of GATE. To maximize the number of topics merged at each iteration, partitions should contain topics that are more likely to be combined than randomly selected topics. This can be done by putting in the same partition topics that share some of the features given in input to the classifier used to compute topic similarity; for instance, topics can be partitioned on the most common character-level 3-gram that appears in their query sets, given that topics with some lexical similarity are more likely to be merged than random topics. The partition criterion can also possibly change at each iteration.

Aggregation on the basis of user identity is one of the most relevant to our study. If the first iteration of the algorithm is run keeping the missions of different users in different partitions, then the resulting agglomeration produces a minimal group of topically-coherent mission sets, called *supermissions*. These supermissions allow to define more compact profile of user activity on a topical basis.

6.3 Clustering experiments on Yahoo! data

We extracted the total activity of $40K$ users from 3 months of the anonymized Yahoo! query log. Statistics for the data set are reported in Table 6.1. The queries in the log were sequentially grouped into 3,005,724 missions using the mission detector described in Section 6.2.1. The number of missions per user and number of unique queries per mission are broadly distributed with average values as in Table 6.1. The first iteration of GATE is performed with a user-based aggregation

Table 6.1: Dataset statistics

Unique queries (uq)	2,198,815
Missions (m)	3,005,724
Unique missions (um)	1,606,733
Avg uq per mission	1.72
Avg m per user	75
Avg um per user	57

criterion, therefore the topics produced in the first iteration are supermissions. The average number of supermissions per user is 42, against the 57 missions per user, meaning that GATE can be used to compress the user description by nearly 30% on average.

In the rest of this section, we present OSLOM [180], a representative network-based clustering algorithm used for comparison with GATE. We then introduce the metrics used to compare such methodologies. The two methodologies are profoundly different, therefore a fair comparison is difficult. Nevertheless we show that our approach has a number of advantages when compared with OSLOM.

6.3.1 Baseline: Topic extraction through network clustering

We compare GATE with a content-agnostic query clustering algorithm based on query graphs. Graph-based clustering considers the queries as nodes and model relation between them with edges. Depending on the relation used, the query graph can assume different topologies and semantics [34]. The choice of comparing our method with a baseline from a different paradigm is motivated by the significant body of recent work and promising results using graph based approaches for query clustering. A thorough comparison between agglomerative clustering and graph based approaches is still missing in the context of query log mining.

The input to our baseline is a query graph based on the *click co-occurrence* relation, also known as *URL cover graph* [34]. Such graph models the topical relatedness of queries from the perspective of the common results to which they lead users: two queries are connected if users click on the same results. Edges are weighted depending on how many distinct clicked URLs are shared by the queries. In principle, several different query-query relations can be mapped onto a single query graph; for instance, it is quite common to mix lexical similarity and click co-occurrence (or other content-related similarity measures). In the present evaluation we consider a simple URL cover graph as input to our baseline.

To detect topically coherent clusters from the URL cover graph, we use a network *community detection* algorithm. Intuitively, the basic idea of community detection is to spot groups of nodes that have many connections with each other and few to the rest of the network, thus forming a dense cluster. Accordingly, a cluster on the URL cover graph would include a set of queries that have many more clicked URLs in common compared to all the other queries in the corpus.

Among the many community detection algorithms in the literature [112], we adopt the recently

proposed OSLOM (`www.oslom.org`). This choice is motivated by its good performance over other state of the art algorithms on both synthetic benchmarks [179] and real network datasets [180]. Furthermore, unlike the vast majority of other community detection techniques, OSLOM automatically detects overlapping communities and hierarchies of clusters. This allows a more direct comparison with our greedy algorithm, which also outputs overlapping topics through a hierarchical agglomeration process.

OSLOM performs clustering based on the optimization of a local fitness function that measures the statistical significance of the detected cluster compared to a global randomized null model known as configuration model [234]. Clusters are detected by selecting several random seed nodes in the graph and finding the locally optimal clusters that include the seed nodes. Similar clusters found over different realizations of random seeds are then merged together, thus originating a minimal set of clusters that may overlap. The procedure is iterated over higher hierarchical levels by collapsing clusters into nodes. Iterations stop when no higher level clusters are found. For further details about the algorithm we refer to the original paper [180].

For the sake of simplicity, here we focus on this single baseline. One could consider even more sophisticated baselines, for instance combining click co-occurrence with lexical similarity features, or even using query clustering algorithms based on alternative paradigms [46, 144, 70]. A more direct evaluation could be achieved with the golden standard of human judgments on the quality of the topics. A preliminary effort in this direction has pointed to the difficulty of human inspection of topic quality, as well as the challenge of identifying a suitable trade-off between topic specificity and broadness.

6.3.2 Metrics

In the following, we present the criteria used for a quantitative and qualitative comparison between the two methodologies. Although prior work has relied on human editors, manual evaluation for a huge corpus of topics is unfeasible, therefore here we focus on automatic evaluation metrics.

Clustering measures

The first group of metrics are meant to give a quantitative comparison of the two methods. We define three measures:

query set coverage: fraction of queries that the methodology is considering in the clustering phase;

singleton ratio: fraction of queries that remains isolated in singleton at the end of the iterative procedures;

aggregation ability: percent of topics that are aggregated in two consecutive iterations or in two consecutive hierarchical levels.

Algorithm 2: Algorithm to compute the concept vector

Require: Concept dictionary D, query q, parameters k, K

1: Retrieve set R of top-k results for q
2: T = Terms from D contained in R
3: Eliminate from T terms in q
4: **for** term $t \in T$ **do**
5: $d(t)$ = number of results in which t appears
6: $r(t)$ = sum of ranks of the results in which t appears
7: $R(t) = \frac{((k+1)\ d(t)) - r(t)}{d(t)\ k}$
8: $S(t) = \frac{d(t)\ R(t)}{k}$
9: **end for**
10: Return the K terms with highest score $S(t)$

Cluster purity

The quality of a topic depends on its *purity*, or semantic coherence. When no ground truth about the correct composition of a topic is available, an automatic way to assess purity is to consider the results for all pairs of queries in a topic. We compare the result sets of two queries to assess how *related* they are, and average the relatedness over all pairs.

Perhaps the simplest way to establish the relatedness of two queries is to compare the result sets returned for each query and use the intersection to derive a measure of similarity [273, 111]. However, a more fine-grained criterion that uses the information inside the clickthrough result provides a more accurate evaluation basis. We construct a bag-of-words vector for each query, consisting of the *concepts* in the documents returned for this query. Concepts are selected using a predefined dictionary [24]. For a given query q and a concept dictionary D, Algorithm 2 computes the vector of concepts with their scores. The parameters are set to $k = 10$ and $K = 20$. The score of a term $t \in D$ depends on the number of documents in which the term t appears, and on the sum of ranks of those documents. Table 6.2 shows examples of two pairs of topical related queries, and their top concept terms ordered by descending score. Given the concept vectors of the queries that it contains, the concept vector of a topic is obtained by marginalizing all concept vectors and keeping the top concept terms.

Let us define the purity of a topic by considering all pairs of terms in the corresponding concept vector and measure how much they are related to each other on average. One could achieve this using the well-known pointwise mutual information (PMI) given by:

$$PMI(t_1, t_2) = \frac{f(t_1, t_2)}{f(t_1) f(t_2)}$$

where $f(t_1, t_2)$ is the number of queries that have both terms t_1 and t_2 in their concept vectors, and $f(t_1)$, $f(t_2)$ are the numbers of queries that have t_1 and t_2 in their concept vectors, respectively. One weakness of PMI is that it may become very unstable for a pair of rare terms. For example, if both $f(t_1)$ and $f(t_2)$ are small, a few coincidental co-occurrences may lead to a superficially high PMI value. To take this into account, we use the log-likelihood ratio, which is the expected value

Table 6.2: Examples of the concept vector

Query	**Terms in aboutness vector (ordered)**
iphone	iphone, store, 4, camera, phone, apps, inc, facetime, mode, recording, software, 3gs, resolution, battery, shop, network, ios, 3g, broadband, ipod
iphone 4 reviews	phone, iphone, store, 4, camera, apple, reviews, apps, review, 3g, recording, model, inc, screen, resolution, ipad, coverage, network, photo, 3gs
toyota prius	hybrid, car, prius, toyota, mpg, photo, price, sales, specs, vehicle, yaris, cars, review, reviews, msrp, specification, milage, model, economy, research
toyota yaris	yaris, hatchback, car, price, hybrid, toyota, models, spec, mpg, liftback, dealer, model, vehicles, review, photo, reviews, city, prius, transmission,vehicle

of the PMI:

$$\begin{aligned} LLR(t_1, t_2) &= p(t_1, t_2)PMI(t_1, t_2) + p(t_1, \overline{t_2})PMI(t_1, \overline{t_2}) \\ &+ p(\overline{t_1}, t_2)PMI(\overline{t_1}, t_2) + p(\overline{t_1}, \overline{t_1})PMI(\overline{t_1}, \overline{t_1}) \end{aligned}$$

where $\overline{t}$ denotes the set of all terms except t. LLR fixes the unstability problem of PMI. Note that when the marginal query frequencies $f(t_1)$ and $f(t_2)$ are small, the other terms in the LLR equation will start to dominate. Averaging LLR across all the pairs of terms in the topic concept vector, we obtain the log-likelihood ratio for the topic.

URL coverage

The topic LLR only focuses on the purity of the topic, which by itself is not too meaningful for the evaluation. A trivial solution of considering each query as a topic would yield very high topic LLR values. Similarly, one can easily cluster only synonymous queries, such as the query "facebook" and its misspellings, again leading to a high purity measure. Similar to singletons, such topics consisting of queries with almost identical results are not useful. To generate meaningful abstractions of the query space, it is desirable to aggregate related queries with different result sets. We therefore need a measure of *coverage* to complement purity, as suggested by previous work on clustering [296].

We measure the coverage by the number of unique URLs in the result sets for the queries in the topic. Given the 2010 query logs of the Yahoo! search engine, we extract all distinct URLs clicked by users for each query. To remove the tail of the clickthrough distribution, URLs that received less than 0.01 clickthrough rate are discarded. We define the coverage of a topic as the aggregate number of distinct URLs across the queries in the topic. Note that the trivial solutions of singleton topics or synonymous queries have very low coverage. Overall, our goal is to extract topics with both high purity and high coverage.

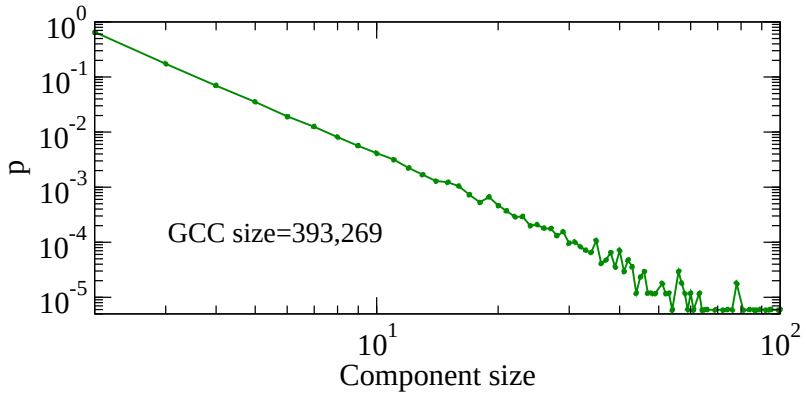


Figure 6.3: Distribution of connected components size in the query graph, except the giant connected component (GCC).

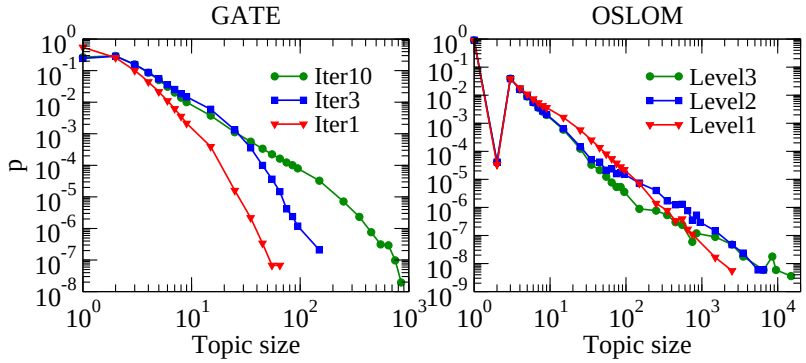


Figure 6.4: Distribution of topic size for GATE and OSLOM. Different curves in the same plot show the distributions at different hierarchical levels.

6.3.3 Experimental Results

In this section we compare the two clustering algorithms according to the metrics presented in Section 6.3.2.

The GATE algorithm has a full query coverage on the dataset, because, by definition, every query can be found in at least one mission. Conversely, when dealing with graph-based clustering algorithms, the sparsity of the graph can lead to the emergence of isolated components that directly affect query coverage. In fact, since the vast majority of queries share very few clicked results, it turns out that the URL-cover graph is composed by a galaxy of tiny disconnected components. These little islands cannot be merged with other components due to lack of connections and, since they are mainly singletons, they do not represent any meaningful cluster just by themselves. The size distribution of the components in the URL-cover graph generated by our dataset is shown in Figure 6.3. We note that less than 400K queries are in the Giant Connected Component (GCC), thus leading to query set coverage below 0.2.

Figure 6.4 shows the distribution of the topic size with increasing number of iterations in GATE and in the three different hierarchical levels detected by OSLOM. The first thing to observe is that

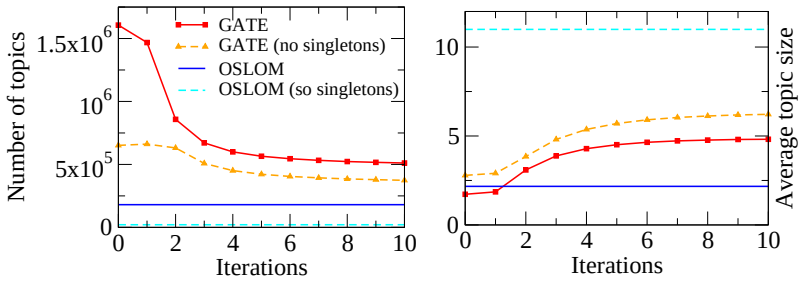


Figure 6.5: Number of topics and average topic size vs. GATE iterations. The values from the lowest hierarchical level of OSLOM are reported as a reference.

the singleton ratio in GATE decreases from 0.55 in the first iteration to 0.27 in the last one, while in OSLOM it remains stable around 0.88. Second, while in GATE the number of both medium-size (10-100) and big size (¿100) topics grows with the iterations, in OSLOM the medium-size topics that are detected in the first hierarchical level tend to be merged in very large comprehensive and heterogeneous topics, with up to some tens of thousands queries. Note that the low frequency of size-2 topics in OSLOM is probably due to the tendency of the algorithm to merge dyads in larger clusters to optimize the partition fitness function.

Lastly, the aggregation ability of the two algorithms is shown in Figure 6.5. In GATE the number of topics decays quickly in the early iterations and then stabilizes, until the stop condition is reached. The final number of topics is around 500K, against the 180K found by OSLOM; recall that OSLOM only deals with the fewer queries in the GCC. The number of topics in the different hierarchical levels in OSLOM varies very slightly. Furthermore, the average size of OSLOM topics is more than double the size of the topics generated by GATE, mainly because of the presence of very large topics that skew the mean value.

Results on topic purity are shown in Figure 6.6. To check how the purity of the topic decreases with its size we computed the correlation between the topic size and LLR by averaging the LLR values of the topics with the same size:

$$\langle LLR_s \rangle = \frac{1}{|T \in \mathscr{T} : size(T) = s|} \sum_{T \in \mathscr{T}: size(T)=s} LLR(T).$$

As expected, the larger the topic, the more heterogeneous the queries included, so we observe a negative correlation. OSLOM's topics have on average higher LLR than topics from GATE; this emerges clearly when computing the ratio between the average scores obtained by the two approaches for each topic size. This is mainly due to the fact that OSLOM generates many topics that include just concept *reformulations*, i.e., queries that are different from the lexical perspective but express exactly the same concept (e.g. "duran duran", "duan duran", "duran", etc.). As we remarked, such queries surely belong to the same cluster, but still they do not express a complex cognitive content. For this reason, it is necessary to complement the purity measure with the URL coverage. In Figure 6.7, the same analysis made for the URL coverage shows that the greedy algorithm has much higher coverage than OSLOM.

If considering purity and coverage metrics separately is useful to learn the peculiarities of the two

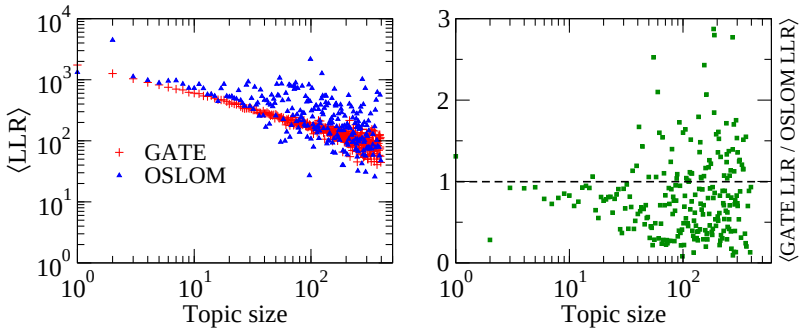


Figure 6.6: Left: Correlations between topics size and LLR values. Right: Ratio between the two quantities.

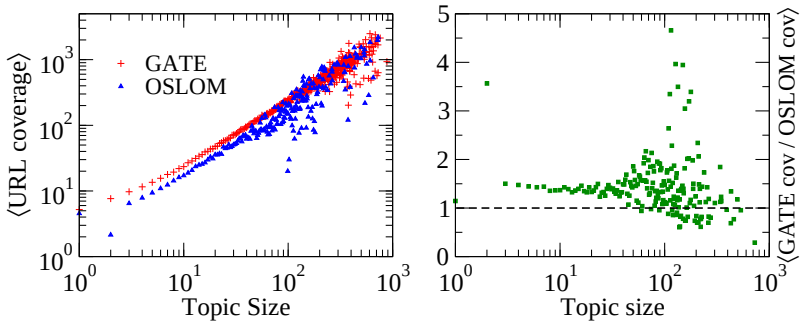


Figure 6.7: Left: Correlation between topic size and URL coverage. Right: Ratio between the two quantities.

approaches, a joint measure that captures the tradeoff between the two scores is useful to compare the overall quality of the query clusters. Such unified measure can be obtained by multiplying the purity by the coverage for all the size classes of the topics and then comparing the resulting curves. Results in Figure 6.8 show that GATE outperforms OSLOM for all the medium-small topic sizes, that represent the great majority of topics. For clusters containing about 100 queries the two approaches are comparable, and only for some bigger, more rare topics OSLOM achieves the best performance.

To summarize, if we consider all the quality measures together, we can conclude that GATE leads to a better topical query clusters because it is able to process all the queries in the corpus, and most of the topics it generates have a better tradeoff between purity and coverage compared to OSLOM.

As a final remark we note that a fair comparison of the computational time needed by the two techniques is hard, first because OSLOM's theoretical complexity is difficult to estimate [180] and last because GATE can run in parallel while the current OSLOM implementation does not enable parallelism. However, we underline that parallelization is a strong advantage of GATE, because when dealing with real-life data sizes, it is hard to regard non-parallel algorithms as practical solutions for clustering web search queries.

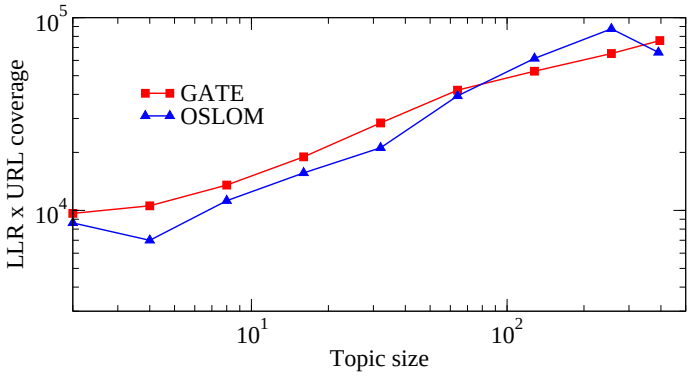


Figure 6.8: Trade-off between purity and URL coverage

6.4 User Profiling

A practical way to use the topics extracted from the query log is to profile users on a topical basis: each user can be described by the set of topics that match her queries. Since OSLOM has a small query coverage and allows only for an exact match between the user queries and the queries inside the clusters, it is not practical to employ it in profiling. Therefore, here we focus on user profiling based on our approach.

To build the profile of a user, we apply the topical similarity function $\mathscr{S}$ between the user missions and every topic that contain at least one query from that mission, then selecting the best match. Formally, let us define the topic that best matches mission m as:

$$T_m = \arg\max_{T \in \mathscr{T}} \mathscr{S}(m, T).$$

Given the best match scores, let us define the *topical profile* of a user u as a weighted vector over the topics matching her missions:

$$\mathscr{P}_u = \left\{ \left(T_m, \frac{\mathscr{S}(m, T_m)}{\sum_{m' \in \mathscr{M}_u} \mathscr{S}(m', T_{m'})} \right), \forall m \in \mathscr{M}_u \right\},$$

where $\mathscr{M}_u$ is the set of missions of user u. For a more compact user representation, supermissions can be used instead.

The topical profile can be used not only to detect the topics relevant to the user, but also to predict her future search goals. To check such a prediction potential, we perform an experiment to examine whether a user profile matches her future missions better than random missions from other users.

The match between a mission and a profile is performed by computing the $\mathscr{S}$ function between the mission and every topic in the profile, and scaling the resulting scores by the weights of the corresponding topics in the profile. This yields a vector of match scores over the profile topics. The match vector can be generalized to sequences of missions by averaging the elements of the vectors across the missions.

We produce the topical profile for the 40K users in the dataset starting from the 3 months of query log. For each user we select two sequences of missions from the 30 days right after the 3

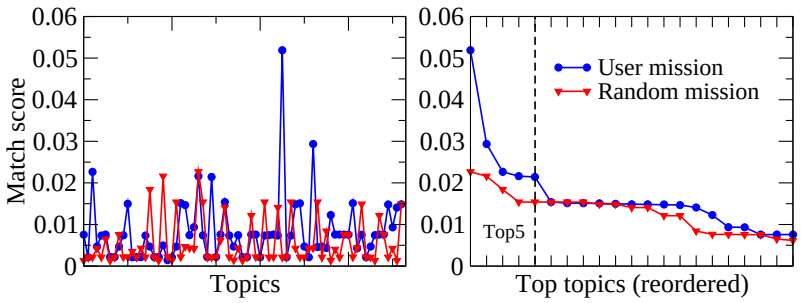


Figure 6.9: Example of match of missions sequences on a topical user profile. On the x axis are the topics in the profile; on the y axis are the match scores of the mission sequences for each topic. The sequence originated by the same user has more spikes compared to the sequence of the random user. When the values of the each curve are sorted, the best matching sequence is evident by looking at the top-N scores.

months considered. One sequence comprises of all the missions generated by the same user in the 30-day period. The other is a sequence of equal length generated by another user chosen at random among all other users.

Intuitively, a mission is likely to match at most a few of the several, possibly different topics in a user profile. Given this intuition, to decide which of the two sequences best matches the profile, we focus on the top-N ($N = 5$) elements of the match vectors between topic profile and mission sequences. We then apply a simple majority rule, i.e., which sequence has the most elements with higher match scores. This idea is exemplified by Figure 6.9.

Using this technique we are able to detect the user's own sequence in 65% of the cases. We stress that the mission sequences of randomly selected users are strongly biased toward high-frequency queries such as "facebook," "amazon," and so on. Since these are shared by a large number of users, any user profile is likely to match them, leading to a decay in detection performance. For this reason we divided the random sequences into three sets according to the average frequency of their queries. The accuracy rises to 72% when considering the sequences with lower frequency queries and drops to around 55% when considering the sequences with higher frequency queries. 65% success in prediction can be considered a good result. Even if the interest of a given user would presumably be quite stationary *within a particular domain*, in web search, where a much wider range of suitable topics is available, the user focus can be often inconstant in time, independent by past search sessions and made even more variable by bursty search activity triggered by external events (e.g., "Michael Jackson death"), hardly predictable by looking only at the user history. This issues make this prediction task much more difficult compared to domain-specific predicition or recommendation.

Chapter 7

Expanding information of collaborative systems

7.1 Tagging relations to achieve complex search goals

In previous Chapters we investigated how complex information can be handled and mined to provide data-driven services in both social and search contexts. A very interesting and widely used type of service that lies in the intersection between these two fields is the collaborative tagging system. As well as classic search engines, folksonomies are indeed an instrument to reduce information overload in search by means of a collaborative, social mechanism.

Since the scientific community has turned his attention on folksonomies, many aspects of collaborative tagging systems have been studied and developed. Recent research on social bookmarking includes the analysis of behavioral patterns, topical trend detection in tagging [295, 126], and studies about the relationship between folksonomies and taxonomical categorization of items [147]. Several efforts have also been spent in improving the tagging systems quality of service by structuring and properly ranking search results in tag-based search [151] or proposing services based on the information extracted from folksonomies, like personalized recommendation services [320, 218] or social link suggestion [299].

Some work has been done in extending tag-based search from a navigational point of view. Services like Yahoo! TagExplorer (`tagexplorer.sandbox.yahoo.com`) allow to specify narrower tags at each search step with the aim of quickly converging to a small set of resources, and other work has been done in realizing platforms for tag navigation, automatic tag clustering [45], and in improving the user experience in navigational search [227].

The problem of improving the quality of the relations between concepts has been somehow addressed in context different from collaborative tagging systems. The work by Völkel et al. [335], for instance, is aimed to extend the links between Wikipedia pages with semantic modifiers that could be used by editors to specify the nature of connections between Wikis.

Despites this relevant efforts, we believe that the search potential of such systems is greatly

underestimated and still unexploited. It has been observed that, among all the search intents, a considerable portion of traffic on traditional query-based engines is characterized by complex search tasks that can be satisfied only through the aggregation of information from multiple sources [95].

Recently, new Web services designed to meet this demand are successfully emerging under two different paradigms. The first uses automated techniques to assist the user during a complex query session. The latter, so-called *social search*, does not rely on algorithms to retrieve accurate responses to queries, but instead routes queries to other users that can provide their knowledge to answer them. Whether relying on the *"wisdom of the crowds"* —where the best reply is reached by a sort of consensus— or on a *"ask an expert"* principle, social search is expected to be more effective than automatic information retrieval in providing results that well satisfy complex search goals.

Originally, the idea of "social search" was referred to the task of searching a particular person through a path between users in a social network [3], but soon its meaning evolved to denote a process of search for knowledge *through* a social platform. In fact, complex queries that cannot be easily satisfied by conventional search engines can be efficiently answered if they are submitted to a social substrate of human computers, especially if they are experts in the query subject. It has been shown that users perceive many advantages in social search like getting personalized answers by trustworthy experts or the possibility of getting satisfactory responses to subjective questions [235]. Q&A bulletin boards like Yahoo! Answers have been the earliest examples of social search, but recently this paradigm evolved in services like Aardvark [150], where queries are automatically routed to users that declared their expertise in the subject of the submitted question.

The power of social search facilities is complementary with the services offered by traditional search engines. The synergy between the two paradigms have been already exploited by services like Google Confucius [306], where users are redirected to a social search platform for some query categories. Another social approach to deal with complex query activity has been adopted by Yahoo! SearchPad [95], a service that automatically recognizes the boundaries of a complex search mission in a query stream, allowing the user to save the session history and, subsequently, to share the steps of a successful search with other users.

In such a scenario, collaborative tagging stopped at an early stage of the Web 2.0 revolution and did not follow the trend outlined by social search. Despite folksonomies produce very high-quality categorization of items [148] and evidence of their effectiveness in improving Web search has been provided in the past [38], up to now tagging is used mainly to accomplish simple search tasks.

In this Chapter we propose a model to realize the unexplored potential of folksonomies in satisfying complex search intents. We present the *relational folksonomy*, an extension of the classical folksonomy that allows to compose complex user-defined *relations* between objects and tag them. While the most known tag-based search engines allow to tag atomic resources like single URLs or multimedia objects, the purpose of our model is to allow users to organize their knowledge or their search experience in meaningful relational structures that represent complex concepts. Resources produced by users or retrieved via classic query-based or tag-based search engines can be connected with relations that express the response to a complex search goal, thus creating a knowledge base

that provides solutions to complex search needs of other users.

The advantages of this model over existing systems are manifold. First, arbitrary relations between objects can shape complex knowledge more flexibly than simple set-like collections of items. Second, our model allows to tag not only relations between resources (e.g., URLs) but also between users, tags, and any combination of them. This implies that users can associate other users to items, thus easily triggering the possibility of social search mechanisms, and they can also share their search experience in terms of the relations between the tags used during a search session. Third, the composition of relations can be performed collaboratively by many users; this allows to solve search problems that are too complex or time consuming to be handled by a single user. Last, the relations between objects provide an easy and meaningful way to navigate *inside* the result. In a scenario where the result is a complex object, this kind of navigation could be reasonably more helpful to the user if compared to classic ranked-list visualizations of results.

7.2 User behavior in tagging item collections

Tagging complex relations between resources can be useful based on the assumption that a structured cluster of objects conveys richer semantics than a directory of unrelated resources. This assumption resembles the holistic philosophical current, effectively summarized by the notorious sentence: *"The whole is greater than the sum of its parts"*. We therefore expect that a sort of *Gestalt effect*, i.e. the human capability of understanding a wider, unifying meaning from a system of distinct but related items, can be observed in this context.

As partial evidence that this principle holds in social bookmarking, we perform an analysis on a popular collaborative tagging system to verify what is the relation between the tags assigned to user-generated clusters of resources and the tags assigned to the atomic items that compose the cluster.

We selected the Flickr photo sharing service for our analysis because, to the best of our knowledge, it is one of the few social media websites that allow users to group together tagged resources. In particular, user can compose *photosets* from single photos of their own; furthermore, photosets have a short title that describes the collection. Even if their structure is very simple and not flexible, since it does not allow a real tagging operation on the set, photosets are good candidate to verify the theory.

Using the Flickr API, we collected the information about 230,020 photosets corresponding to 13,317 users and containing an overall amount of 7,188,004 pictures. For every picture, we extracted the tags that its owner has attached to it. We then compared the photoset title —free from stop-words— with the tags inside the photoset.

The first result is that only the 43% of photoset titles overlap with the tags attached to the images inside the set. This means that in more than half of the cases, photo collections have a name that is completely different, from a lexical point of view, from the labels that define the atomic items inside it. Analyzing the overlap between title and tags at a finer grain, we can also observe that, when an overlap occurs, in roughly 50% of the cases the title contains at least one

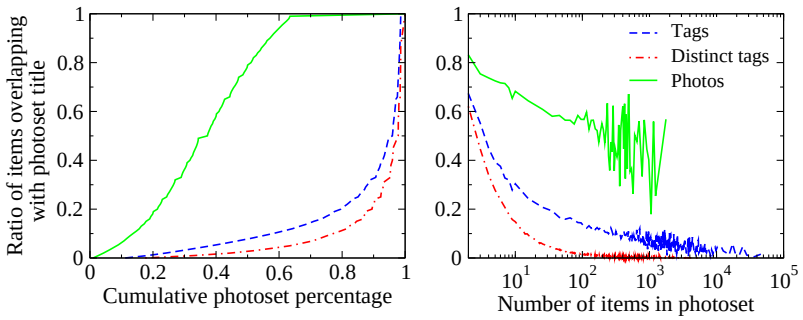


Figure 7.1: Overlap between photosets title and tags inside the set. The measures are restricted to the 43% of photosets whose title has some overlap with tags. Left plot shows the portion of tags and distinct tags that overlap with their photoset's title and the portion of photos that are labeled with at least one overlapping tag. Right plot shows the average values of same measures for photosets with the same item size.

word that does not appear in any of the tags of the photoset resources. This is a hint that very often the title add some semantics to the information specified by the tags of single pictures.

On the other hand, even when an overlap occurs, the tags specify different information than the photoset title. As shown in Figure 7.1, the 90% of photosets contains tags that overlap only 20% of the times with the photoset title and, as one can expect, the overlap ratio decreases as the photoset size grows. From the curves in Figure 7.1 we also learn that it is quite frequent that a picture in the photoset is labeled with a tag that appears also in the title. This is coherent with the observation that users tend to tag resources with words taken by the title of the resource or of the collection which contains it [203]. Moreover, In Flickr this behavior is somehow broaden by the possibility to tag pictures in batch. Nevertheless, it appears that this phenomenon is limited to a minority of cases.

This preliminary analysis is not meant to be a formal characterization of the user behavior in classification of aggregated items; however it clearly shows that when resources are aggregated even in a simple set structure, the description that users give at the set level considerably deviates from the labels assigned at atomic level. This observation validates the hypothesis that the meaning of relationship between resources goes beyond the mere aggregation of atoms.

7.3 Relational folksonomies

A common definition of the classic model of a folksonomy is the following [151]:

Definition 1. *A **folksonomy** is a tuple* $\mathbb{F} := \langle U, T, R, A \rangle$ *where* U*,* T*, and* R *are finite sets of users, tags, and resources respectively. The ternary relation* $A \subseteq U \times T \times R$ *represents the annotation of resources with tags performed by users. Instances* $(u, t, r) \in A$ *are called **triples**.*

We propose an new definition of folksonomy that is a consistent extension of the previous one.

Definition 2. *A **relational folksonomy** is a tuple* $\mathbb{F}_\Re = \langle U, T, R, A, \Re \rangle$ *where* U*,* T*,* R*, and* $\Re$

are finite sets of users, tags, resources, and relational ***bundles*** *respectively. The ternary relation* $A \subseteq U \times T \times \Re$ *represents the annotation of bundles with tags performed by users. The set of relational bundles* $\Re$ *is recursively defined as follows:*

$$\begin{cases} r \in R & \Rightarrow \quad r \in \Re \\ u \in U & \Rightarrow \quad u \in \Re \\ t \in T & \Rightarrow \quad t \in \Re \\ N \subseteq \Re \wedge \rho \subseteq N \times N & \Rightarrow \quad (N, \rho) \in \Re \end{cases} \tag{7.1}$$

Shortly, in a relational folksonomy users start creating bundles by specifying relations between atomic resources in $R \cup U \cup T$. The relation ρ can be picked among the (extensible) core set of relations specified in Table 7.1. In turn, once a bundle is created, it can be recursively included in a higher order relation together with other bundles or atomic resources, according to Formula 7.1. This model is very flexible and expressive, since any combination of simple and complex items through any relation can be created by the users. Of course, bundles can be tagged for indexing or description purposes.

Currently, some of the major social bookmarking services support some sort of grouping functions. Del.icio.us allows to create *tag bundles*, which are tagged sets of tags, Bibsonomy supports *relations* between tags in a hierarchical fashion, and, as seen above, Flickr allows the construction and tagging of *photosets*. Nevertheless, these solutions lack of generality since they are designed to fit their own single domain and hardly adaptable to more complex relations than the simple set. For instance, more general features like nesting of relational structures or composition of different types of relation are not explicitly supported by these models.

Perhaps, the only relevant attempt in changing the paradigm at the basis of folksonomies has been proposed in the GroupMe! project [2]. Similarly to our work, GroupMe! is based to an extension of the classic Folksonomy formulation that allows the creation of groups of multimedia resources. GroupMe! is able to capture just one of the many relations we introduce in our model and, even more important, it deals only with the resource dimension of the folksonomy, thus not being able to embed social search features like tagging and sharing of successful search paths or indexing users and associate them to particular resources.

The main goal of the model is to allow users to link together different resources with a relational structure that conveys complex semantics that would not emerge from a simple collection of the resources. This is particularly useful to organize and share the knowledge on complex topics or to collaboratively build a common relational structure on very articulated matters, in a Wikipedia fashion. Since also users can be inserted into bundles, social information like connections between people can be directly related to resources, for example to identify expertise in some knowledge areas, thus implicitly enabling social search mechanisms. Moreover, the possibility to tag relations between tags allow to index any search process seen as a relation of co-occurring tags in a search session; successful search paths can then be shared to provide other users a guidance to achieve a complex search goal.

Traditional folksonomies are clearly a subset of relational folksonomies that are obtained following the first branch of Formula 7.1. In this case, $\Re$ can contain only elements from R thus the

Relation	Description
Set ($\mathcal{S}$)	$\rho = \oslash$
List ($\mathcal{L}$)	$\rho =<$, antisymmetric and transitive partial order relation on N
Graph ($\mathcal{G}$)	If items are interpreted like nodes, then $\rho \subseteq N \times N$ determines the set of edges between them. It can be undirected if $(n_1, n_2) \in \rho \rightarrow (n_2, n_1) \in \rho$
Tree ($\mathcal{T}$)	A hierarchical structure obtainable as an acyclic graph $\mathcal{G}$
Hypergraph	Graph-like relation between elements in the power-set $\wp(\Re)$, obtainable by the $\mathcal{S}$ relation

Table 7.1: Core relations obtainable in relational folksonomies (Formula 7.1)

folksonomy reduces to the $\langle U, T, R \equiv \Re, A \rangle$ tuple. This means that our model still allows tagging of atomic resources like in classic folksonomies.

The structure of relational folksonomies allows to 1) effectively collect the resources or the knowledge gathered during a complex search activity on a traditional tag-based or query-based engine, 2) add a structured, semantic connection to the accumulated knowledge through *user defined* relations and 3) share the structured knowledge that comes out from this process with other users, using tags.

7.3.1 Use case

For illustrative purposes, we present an example of how the relational folksonomies can be used to solve a real-world task.

Suppose that Alice is willing to share her experience about a recent travel through Italy. She went to Venice, then Florence and, at the end, Rome. She visited the main touristic attractions of each city, taking pictures and videos, writing notes about the hotels she stayed in or the restaurants she went to, collecting online contacts of people she met during the trip.

At the end of her travel, Alice wants to organize and share her experience with friends. Using the proposed approach, she creates a *Set* bundle for each city, containing the media items related to that location; according to Definition 7.1 and Table 7.1, we refer to these as $(Venice, \mathcal{S})$, $(Florence, \mathcal{S})$, and $(Rome, \mathcal{S})$, respectively (supposing that city names are just macro to identify the collection of resources). To represent the temporal sequence of the trip, she arranges the stages of her journey using a *List* that represents the path covered: $Path = (\{(Venice, \mathcal{S}), (Florence, \mathcal{S}), (Rome, \mathcal{S})\}, \mathcal{L})$. Since the model allows to include also people in relations, Alice creates another *Set* bundle $(Participants, \mathcal{S})$ that contains the reference to her fellow travelers. At this point, she shares the bundle $(\{Path, (Participants, \mathcal{S})\}, \mathcal{S})$ labeled with tags *Italy*, *trip* and *vacation*.

Now assume that Bob is planning a vacation in Italy. He submits a query to the system with the tags *Italy* and *trip*, thus finding the bundle previously posted by Alice. He navigates through the bundle collecting information and possibly tagging or personalizing items. Since Bob has not much time for his vacation, he would like to gather opinions on which of the three cities is most worth seeing, in order to spend more time visiting it. To do so, he sends a request to the users in (Participants,$\mathcal{S}$) to ask for their opinion and suggestions.

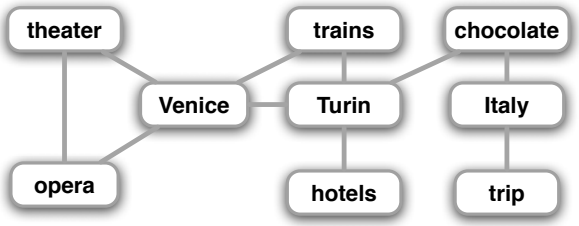


Figure 7.2: Query graph for a user that wants to visit Italy and enjoy opera and chocolate

Since Bob is keen on opera music, he wants to find an opera house in Venice. So, he submits a query with tags *Venice*, *opera*, and *theater*. At the end of the trip, Bob would also like to visit a place where he can taste good chocolate, so he searches for the tags *Italy* and *chocolate*. He finds that Turin is the Italian capital of chocolate and so he plans to go there by train and lodge in an hotel. Figure 7.2 summarizes the *Graph* bundle of tags used by Bob during his search, where tags co-occurring in the same query are linked. This bundle, automatically built by the system during the search session, represents an example of a complex search path performed by a user willing to visit Italy and enjoy opera and chocolate. Bob, who is interested in continuing its search session later, saves this *Graph* tagging it with *Italy*, *chocolate*, and *opera*. Bob can also choose to share its search experience by publishing the bundle. Doing so, other users can begin their own research starting from the knowledge gathered by Bob, possibly learning relations between concepts which are unknown to them (e.g., Turin and chocolate).

7.3.2 Portal

The relational folksonomy model has been implemented in an online Web service available at `http://mumb.di.unito.it`. Although, at this moment, the portal is an early-stage prototype that does not cover the complete set of intended functionalities, it can still provide to the reader a concrete idea of the scenarios that the relational folksonomy model enables. The current portal is structured in three main functional sections: *Search*, *Share*, and *Socialize*.

In the *Search* section (Figure 7.3) users can submit queries to the system obtaining a ranked list of atomic results or complex bundles that meet the search criteria. By selecting a bundle the user can navigate its content, tag it or publish a personalized version. On the left side, filters allow to organize the result set by type, source or ownership. On the right side, the upper box contains a weighted list of tags that are related to the words in the current query string. The similarity relation is derived from the semantic concept network built in the Great Minds Think Alike project [345, 344] (`http://greatminds.givealink.org`) that leverages the games with a purpose paradigm to collect high-quality social annotations on Web resources, tags, media content, people, and geographical locations. In the lower box, the current session is represented by the *Graph* bundle that contains the tags used during the search session, where edges between tags indicates their co-occurrence in a query. At this moment, the user is not able to publish the

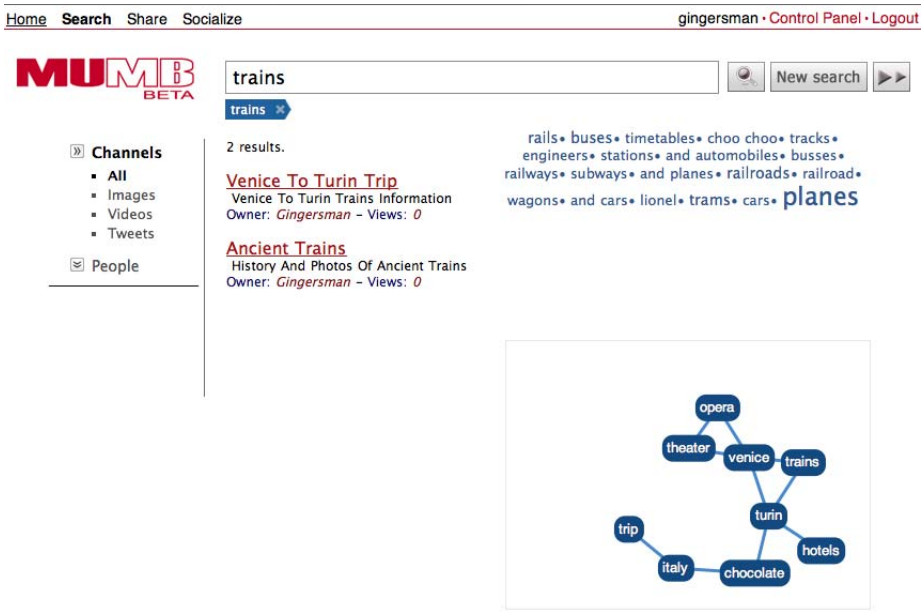


Figure 7.3: Search section

search path, however, this functionality is under development and it will be introduced in the next release since it represents a key functional aspect and a concrete shift from the current approaches.

In the *Share* section users can compose, share and tag bundles. Bundle creation is performed by aggregating content from existing bundles or from social media like Flickr, YouTube, Twitter or any other web services that exports a public API. Since the atomic elements are URIs, the system allows to append anything to a bundle, even users and locations, enabling a geo-social aspect to the engine. Figure 7.4 shows the process of adding a YouTube video to a bundle set. The current prototype provides the ability to create only bundle sets containing photos, videos or tweets. However, it is worth noting that the prototype architecture has been carefully designed to easily support new relations or media channels in future releases.

The *Socialize* section (not implemented yet) is intended to foster the creation of social links between users building a community around them. The system will suggest people interested in similar topics and will enable discussion panels to share the knowledge of experts.

The portal allows anonymous users to search, navigate and create a bundle; instead, sharing, tagging and connecting to people requires a login procedure. Users may login using an account from some of the most popular social media sites like Facebook, Yahoo!, Google and many others. Any service that implements the OpenID protocol is accepted. In the *Control Panel* the user can manage her multiple digital identities and connect them to the local id, enabling services like discovery, personalized recommendation, social networking, and so on.

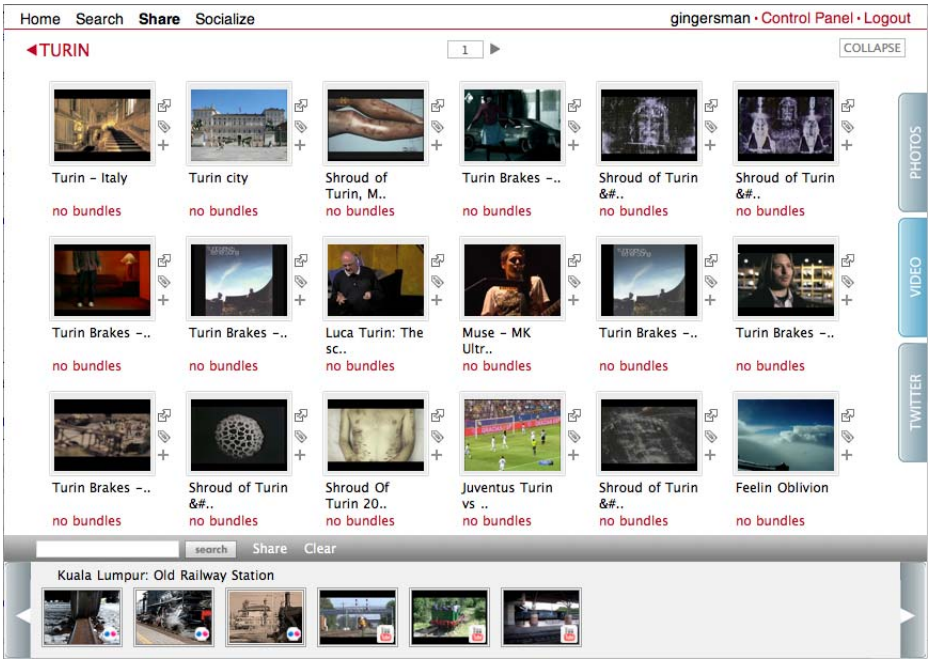


Figure 7.4: Adding a YouTube video to a bundle set

Summary on Part I

The exploration of the dynamics of complex social systems reveals interesting patterns about the macro-structure of the social substrates and the behavior of their human components. We studied the structure and the dynamics of three online social media (Flickr, Last.fm, and aNobii) to understand the underlying mechanisms of their **evolution**.

We first inspected the interplay between individual **user features** and **social interaction**. We observed a strong correlation between the **social connectivity** and the **intensity** of explicit user activities like tagging and participation in groups. Assortative mixing patterns between neighbors can be found for all the examined features as well. We showed a dependency between the **topical similarity** between pairs of users and their shortest-path distance on the social graph, using a null model built to discern purely statistical **assortativity** from actual **homophily**. We found a clear topical alignment trend between neighbors for all the examined features.

The observation of that topical alignment between individuals residing near in the social graph led us to investigate the **dynamics of link creation** in the network, thus revealing strong propensity to **homophily**, **reciprocity**, and **proximity-driven attachment**. We support with strong evidences the thesis that similarity patterns that are detected in the static network are also determined by **influence** that connected users exert on each other. In particular, we observe that link creation triggers a noticeable sudden increase in the similarity between the endpoints. Furthermore, information tend to **spread** along social ties rather than with more chaotic patterns. In this regard, we found that the fraction of neighboring users that may have influenced an adopter is very small on average, thus supporting the intuition that the "**information contagion**" is often triggered by a few of trusted, influential users. Accordingly, we observe that interaction ties, i.e., social connections over which some **communication** takes place, exhibit a stronger topical alignment and influencing potential.

Based on the previous findings on dynamics of interaction and of link creation, we studied the possibility of **predicting** the creation of social ties based on the similarity of users. We discuss different **similarity metrics** to be applied to structural and profile features. In particular, we explore a number of similarity metrics that can profitably model the user-to-user similarity in the three-dimensional space of **folksonomies** and we inspect many different topological similarity measures.

Results show that all the user profile features have substantial power in detecting links. Regarding the folksonomy-derived features, the similarity metric that we introduced to compare objects

in the folksonomy space, namely the MIP similarity, produced the most accurate results in link detection. Moreover, the aggregation technique used to extract folksonomic features from their original three-dimensional space is very relevant to the prediction results. Overall, the library feature, i.e., the collection of the focus items in the social media (e.g., books in aNobii and artists in Last.fm), turned out to be the most effective, and was very accurate even when users have a small number of library items.

Contextually to the study of link detection we also tested the effect that confounding aspects like spoken **language** and nationality have on the prediction task. We showed that the link detection task is easier in social networks that are strongly clustered by language, because users in different clusters tend to have very different topical interests and to establish very few social ties.

Focusing on a single language cluster, we observed that combining profile features with network-based features through a **machine-learning** approach leads to a considerable boost in link prediction accuracy. Using the Chi squared attribute selection method, we are also able to rank all the features considered according to their predictive power, thus being able to select a minimal set of them to train our classifier. Based on this, we present an efficient **contact recommendation** technique based on topological and profile features that reaches a good precision in conditions of extreme **sparsity**.

The possibility of mining user data to understand hidden information and, based on that, predicting the evolution on the system or the future behavior of its components, is not limited to social networks. In particular, we study the scenario of **Web search**, where the information overload in extremely high on both server and client sides. A possible way to synthesize the huge amount of information available in this context is to cluster together the information units depending on their relevance with respect to some more general **topic**. The **behavior** of the users in submitting queries to a search engine, including the implicit and explicit information that their actions reveal about their **search intent**, is a crucial element to determine what is the topic of a query or of a sequence of search actions.

We introduce a novel definition of topic in the context of **query log analysis** and propose a topic extraction algorithm based on agglomerative clustering of sequences of queries that exhibit a coherent user intent. Our algorithm relies on a semi-supervised classifier that can tell if two query sets are topically coherent with excellent accuracy (AUC 0.95). We compare our method with a graph-based clustering baseline, showing its advantages on query coverage and on the trade-off between purity and resource coverage of the clusters. We define the **topical profile** of a user in terms of a topic vector that best defines the user search history. With our classifier we are able to discriminate a query sequence submitted by the profiled user from a random query sequence 72% of the time in the best case scenario.

Finally, we give a contribution in improving the effectiveness of information retrieval in a systems that resides in the intersection between the macro-areas of **social** and **search**: folksonomies. We propose an **extension to the folksonomy model** that enables to tag complex user-defined relations between users, tags, and items. As a premise, we analyze the connection between tags associated to Flickr photosets and tags related to the single pictures composing the photoset. We

observe that even this simple form of aggregation conveys richer semantic than a directory of unrelated objects. Accordingly, we define the concept of **relational folksonomy** presenting a concrete example of how they can be used to solve real-word tasks. To the best of our knowledge, this is the first attempt to leverage the power of folksonomies to enable social search and the achievement of **complex search** missions in a **collaborative** way. An online Web portal is presented as a reference implementation of the model.

Part II

Privacy-aware online social platforms

Chapter 8

Privacy leaks of centrally managed systems

> *People have really gotten comfortable not only sharing more information and different kinds, but more openly and with more people and that social norm is just something that has evolved over time.*
>
> Mark Zuckerberg, creator of Facebook (2010)

In Part I we discussed how mining user generated data in social and search systems could help to extract useful information for the providers and to build better services for the users. However, the virtuous circle between the availability of user data and the quality of services has a cost in terms of *privacy*.

In the case of social systems[1] the privacy hazards on user information can be divided in three main categories, namely *direct data exploitation*, *information leakage* [170], and *information linkage* [169]. Direct exploitation of data is the simplest and most perceptible threat; it occurs when an undesired information mining is performed by the provider that stores the user data for purposes that may be also unrelated with the context in which the information has been defined.

A more subtle risk is given by the information leakage, i.e., the diffusion of sensible data out from the context they were originally defined. The leakage can happen along social ties and groups when some piece of information originated by some user reaches a wider audience than desired or expected; this is mainly due to the limits or to the opacity of the privacy policies provided. However, the leak can happen also on a larger scale when the data are brought outside the boundaries of their original context. In fact, very often, the content management policies adopted by online social network providers allow third parties to access the user information, which can then be used for any other purpose, for example faceted advertising.

[1] We focus our discussion on privacy on social systems because they represent a more extensive case study than search systems in terms of privacy issues. Also keep in mind that social and search systems are increasingly entangled (just think of the strict relation between Google's services *Search* and *Plus*) and they should be considered as a single ecosystems of user information plunged in a social context.

Last, information linkage is a data combination technique used by the data holder or even by unauthorized third parties to aggregate data from different data repositories to infer non-expressed information about the identity or the behavior of a user. Facebook's instant-personalization service [104] and the access policies to private resources for third-party applications [110] are evident examples of such possibility.

Therefore, the actual risk in this setting is that users may lose the control over the diffusion of at least part of their own information. Even if most of existent SNSs offer the possibility to change some privacy settings or profile visibility options, a full customization of the settings is often not allowed. Furthermore, the recent trend of providers policies has been to relax more and more the privacy constraints in the default user settings [251].

Although, until the recent past, users could be misinformed about who actually has the permission to access their online information [136], the awareness on SNSs privacy leaks is rapidly growing and spreading among OSN users, thanks also to the relevance attributed by the media to this topic. Recently, also many voluntary protest initiatives have bloomed on the Web [25, 232, 135], reflecting the fact that the user demand for privacy is becoming stronger.

For these reasons, to restore the control of the users on their own information has become a major challenge for the research community today [26].

To mitigate the information leakage, some efforts have been made to improve the privacy settings in OSNs. Felt et al. [110] address the privacy problems related to the disclosure of personal information through APIs provided by online social services like Facebook and OpenSocial. They propose an alternative social platform design that prevents third party applications from obtaining real user data. They introduce a *privacy-by-proxy* anonymization technique that do not reveal any non-public information to external applications or to the application users that are not granted with the proper privileges. This goal is achieved by customizing the interaction protocol between the social service provider and the applications.

To limit the data leak along the social connections, Fang et al. [107] propose a machine learning methodology to infer resources access control policies to be applied to the set of social contacts. Based on a small learning set of representative friends for which access control settings are actively specified by the user, their method infers the proper privacy settings for all the other users in an automated way. The features used in the training phase include the position of a friend on a specific network cluster in the friendship graph and all the profile information specified directly by the user.

The definition of smarter techniques to control the diffusion of user data is useful but may also collide with the reticence of OSN service providers to change the privacy features they offer to their clients. For this reason, another line of work proposed a set of workarounds to avoid undesirable information mining or defining guidelines to enhance the privacy services offered by SNS providers, but without changing any feature of the original services. This choice is motivated by the advantage in preserving all the existent services and by the observation that many OSN service providers may be unwilling to change the privacy policies.

One of the most evident examples of how a centralized social media service could be "patched"

with some additional privacy features is the NOYB system [139] (`adresearch.mpi-sws.org/noyb.html`). It aims to solve the problem of user information exploitation with a direct encryption approach. NOYB is a tool based on a substitution cipher that allows to post encrypted textual information on the social network. A secret key shared between NOYB groups, together with a public dictionary, stored by a Trusted Third Party, are used for encryption and decryption and to produce ciphered text that is hard to be discerned from real text with automatic techniques. Doing so, the provider is prevented to tell fake encrypted data from valid information. Its feasibility is supported by a proof-of-concept sketched on the Facebook case.

Even if this approach applies only to textual fields and it is not suitable for any other kind of sensitive information, like social connections, group affiliations, or pictures, it reveals a general need to find simple and effective solutions to the mentioned privacy issues. On the same line, similar solutions have been presented [210].

To avoid direct data exploitation, some solutions to build privacy-preserving services over untrusted storages has been proposed. *Persona* [33] is a privacy-aware OSN where user data are kept into a centralized untrusted storage. Information privacy is accomplished using encryption. In particular, Attribute Based Encryption is used to flexibly manage group membership and revocation of privileges. A possible integration with existing OSNs is drawn. A similar approach is described by Anderson et al. [23], who propose a client-server model with untrusted server as a privacy-aware OSN architecture. In this work, the server is used as a public storage without any access control policy (it offers only simple PUT and GET primitives). The task of preserving the privacy of user information is delegated to clients, that upload encrypted and equally-sized data blocks to hide both resource content and size. Only users with proper keys can access to the items. Identity verification and key distribution are managed in a P2P fashion.

More sophisticated approaches has been also proposed. As an example we mention *Lockr* [326] (`www.lockr.org`), a discretionary access control system adaptable to centralized OSNs as well as to the BitTorrent streaming system. The idea is to decouple the user social information (i.e., the contact list) from the resources the users share with others. In this approach, the list of friends is not explicitly stored anywhere; instead, every user distributes a signed social attestation to each of her social contacts; only users that have a proper social attestation are allowed to access the resources. In order to avoid replay attacks, the attestation is verified through a zero-knowledge protocol. In a centralized setting, a dedicated server is responsible for the attestation validity check. In BitTorrent, thanks to the introduction of a signed social torrent, attestation verification can be managed in a decentralized manner.

All the solutions mentioned above are useful both to explore the different facets of the problem and to raise the awareness on the privacy flaws of OSNs. However, they cannot be enforced easily. First, applying a blurring mask to the user data layer in order to hide sensitive information stored in a centralized environment is a solution that would be not accepted by the service providers themselves, whose terms of service often impose very strict conditions on information inserted by users [105]. Furthermore, the implementation of privacy-aware solutions that imply to build up a centralized infrastructure for data management would be too costly if a very large audience must

be supported.

For this reasons, independently from the above proposals, an important line of research focused on solutions based on the replacement of centralized platforms supporting OSNs with peer-to-peer (P2P) systems.

The most commonly understood meaning of P2P systems is a class of distributed systems where all the nodes cooperate to accomplish some task by acting as both providers and consumers of some resource. A single P2P system, understood as the substrate of the peers and the logical connections between them, together with a communication protocol, can potentially support a wide range of different applications simultaneously. File sharing applications like Napster (`www.napster.com`), and eMule (`www.emule-project.net`), real-time communication services like Skype (`www.skype.com`) and ICQ (`www.icq.com`), or content distribution networks like BitTorrent (`www.bittorrent.com`) and Joost (`www.joost.com`) are noticeable examples of applications that follow (at least partially) the P2P paradigm, even if relying on very different protocols and on different structure of the network of logical connections between peers. Nevertheless, despite their differences, all P2P applications are united under the same philosophy that is well summarized by Shirky's definition [305]:

> *P2P is a class of applications that takes advantage of resources — storage, cycles, content, human presence — available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have* ***significant or total autonomy from central servers****.*

The autonomy from central coordination and control is an excellent basis for the construction of privacy-aware services. Replacing centralized SNS providers with peer-to-peer services [26, 67], where no central authority can claim the right to exploit the data, is a proposal that has achieved widespread success among the scientific community. Furthermore, a P2P solution could be realized with a negligible infrastructural cost if compared to centralized architectures.

In particular, considerable attention was paid to structured P2P systems, or Distributed Hash Tables (DHTs). In fact, DHTs are the most natural choice to implement a decentralized storage for user information, since they transparently provide content availability, caching and dissemination, assuring a very high scalability of the system. However, implementing complex SNSs on such purely decentralized layers lead to several practical problems, that can be summarized with the keywords *security*, *access control* and *services*.

First, DHTs suffer from many security issues [330] that can considerably undermine the stability of the network, and of course of the applications layered above. This point is even more important for social applications, that usually have high robustness requirements motivated by the high-quality user experience they must offer. Second, in a open environment like a DHT-based storage system, data can be potentially accessed by everyone. For this reason, proper access control policies should be implemented in order to offer a full customizability of content privacy levels and to support a fine-grained and highly-dynamic group membership. Last, P2P overlays offer a

very restricted and low-level API. Since social applications usually require high level primitives like inter-application notifications or identity management functions, the distributed social layer should expose a suite of higher-level services that prevent an excessive overhead in the implementation of applications. Of course, while realizing such services in a centralized environment can be trivial, it can be very challenging to do the same in a fully decentralized environment.

Given this context, the main contribution of this part is the following. First, we describe the general model of structured P2P layers and we survey their security weaknesses (Chapter 9). For a specific and very dangerous kind of attack we prove through a simulation study that the network can be subverted with a very low effort by an attacker. Accordingly, we propose a new Distributed Hash Table system that strongly mitigates the effect of all the most known attacks that can be directed to structured overlays (Chapter 10). The idea at the basis of this new DHT is the embedding of a strong user identity management at overlay level; we show that this feature has very good implications on DHT properties and potentialities, at a reasonable cost in terms of network latency and required computing power. On the top of our security-enhanced DHT, we define a distributed SNSs that effectively tackles the mentioned privacy problems (Chapter 11). The identity-based primitives provided by our architecture allow us to define a set of basic core social services for custom OSN applications. In particular, our service suite includes fine-grained discretionary access control, reputation management, synchronous notifications and high level search primitives. We will focus in particular on how an efficient and privacy-preserving tag-based search engine can be build in such decentralized environment (Chapter 12). We show that our approach allows an easy application mash up and relieves the applications from many service implementation details, thus making their development quick and extremely easy.

Chapter 9

P2P overlays and their security issues

9.1 Structured P2P networks

Peer-to-peer systems have been studied and categorized according to several dimensions like their purity (i.e., presence of super-peers or centralized components) [351] or the structure of the logical connections between nodes [155]. Nevertheless, one of the most accepted dichotomy in taxonomies of P2P systems is the separation between *unstructured* and *structured* systems [209].

The main distinction between the two categories is the way in which messages are routed along the logical connections between nodes. In unstructured networks, messages are sent in broadcast from the originator to a set of known neighbors; in turn, each receiver broadcasts the message to its own neighbors, originating a message cascade that can, in principle, flood all the network before the recipient of the message is received. The Gnutella network [280] is the most known example of such kind of network.

Conversely, the message routing in structured overlays is constrained by a strict management of the logical connections between nodes. For each message, the set of peers to be contacted to continue the routing procedure is deterministically given by tight routing protocols and the message can reach the target node(s) with a very limited overhead in term of number of exchanged messages. For this reason, structured networks has been considered more convenient to use in a wide range of applications than unstructured layers [287, 231, 171]. Structured overlays are often referred as Distributed Hash Tables (DHTs).

More in detail, DHTs [318, 285] are a class of fully distributed systems which provide an exact-match lookup functionality: given a certain *key* from a flat identifier space, they retrieve the value associated with such key. From an application point of view, at a high level of abstraction, a generic DHT system could be defined with a 6-tuple:

$$DHT = \langle K, N, C, \kappa_N, \kappa_C, \lambda \rangle$$
$$\kappa_N : N \rightarrow K$$
$$\kappa_C : C \rightarrow K$$
$$\lambda : K \rightarrow \{N\}^*$$

K is the DHT *keyspace*, a large (usually 2^{128} or 2^{160}) set of numeric keys, N is the set of online nodes and C is the set of all the resources (also referred as "content") owned by the users. An identifier chosen from the keyspace is assigned to every node (function κ_N) and content (function κ_C). Usually, nodes generate randomly their ID (the *NodeId*) while the content ID is calculated from the cryptographic hash (for which a collision is unlikely, for large values of $|K|$) of the content payload or from its metadata; however the definition of the function κ_C could be delegated to each specific application, depending on the structure of the resources. λ is a function of *responsibility* that associates the task of storing all the content marked with the same key to a set of replica nodes; these nodes are called *indexes* for the key. A node interaction protocol can perform a lookup procedure that, in $O(log|N|)$ steps, is able to locate the index nodes for any key.

Various DHT specifications differ in the routing table structure and updating procedures and in the nature of the lookup procedure (which can be either iterative or recursive). A real DHT implementation must also provide several other features like techniques to maintain content over time even with a high node *churn rate* (i.e., the frequency of join or leave of new nodes to the P2P network), or caching strategies that avoid *hot spots* for popular keys.

We talk about *structural components* of a DHT referring to four elements:

- The mechanism for *identifiers assignment*
- The *routing table*, containing the contacts of the known nodes
- The *storage* of resources
- The *interaction protocol* between nodes which determines the lookup procedure and the *bootstrap*, that is the join process of a new node to a existing network.

Several DHTs have been defined in the past. They mainly differ in the structure of the keyspace and in the routing procedure but they all share the general principles we described in this Section. In particular, we mention Chord [319], Pastry [285] and Tapestry [361] among the others [277, 214]. However, in the following we focus on the Kademlia DHT, that is the basis of our secure framework.

9.1.1 Kademlia

Kademlia is the reference protocol on which we base our secure DHT. Here we give an overview on its core features. Some details (such as caching strategies) that are not functional for the following sections are not presented here; for further details, please refer to the original paper [221].

Route table and state maintenance

In Kademlia, each node of the network has an identifier (ID) consisting of a 160 bit vector generated using the SHA-1 hash function on a random value. The nodes of the network can then be

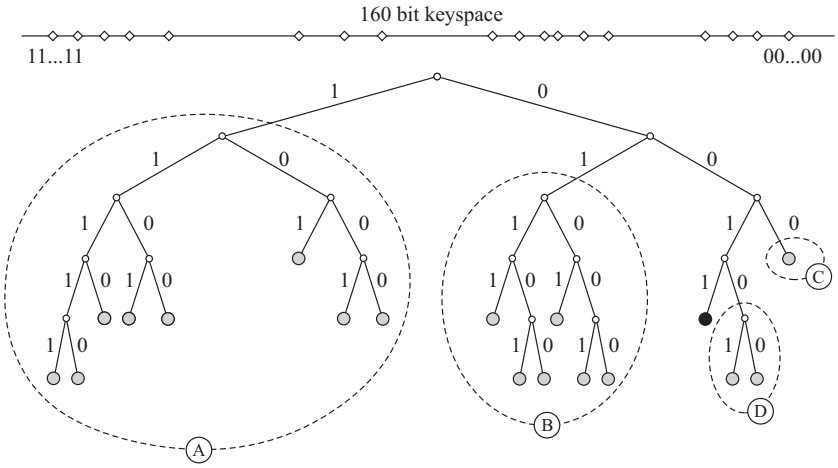


Figure 9.1: Kademlia tree. The black node is the local peer whose ID has the unique prefix "0011"and the gray nodes are the other peers known by the local node. Four subtrees not containing the local nodes are marked. These are obtained by iteratively splitting the largest tree containing the local node. Each subtree contains nodes with some common prefix. Subtree A contains all the IDs starting with "1", B with "01", C with "000" and D with "0010".

represented as leaves of a binary tree with 160 levels. Clearly, in any real case the number of nodes is much lower than 2^{160}, so it turns out that the majority of the leaves correspond to free IDs. For this reason, each node can be identified by the initial portion of its ID that is distinct from any other ID present in the network.

A distance between two IDs is defined as the XOR between them. The distance intuitively represents the number of levels that separate two leaves when navigating from one node to another in the binary tree. The order of magnitude (base 2) of the distance expresses the number of levels that must be navigated up in the tree to go from one node to another.

Figure 9.1 shows how the contacts known by a peer can be partitioned in sets where all the IDs share the same prefix. These sets are called *k-buckets* and form the route table of Kademlia peers. Each bucket contains at most k IDs[1] that have the same distance (in order of magnitude) to the ID of the local node. k-buckets are updated during the interaction with other peers. When receiving any kind of message, the sender ID is put in the corresponding bucket; if the bucket is full, the least recently seen contact is pinged and, if no reply is received, it is replaced by the new contact. This strategy is used to keep the route table more stable as possible and to replace only the inactive nodes.

When entering the network, a node starts with a minimal route table with a single bucket containing the contact of a bootstrap node. During its lifetime, the node expands its route table by adding new k-buckets through a *bucket splitting* procedure. When the k-bucket in which the ID of the local node is contained becomes full, the next contact to be added triggers the splitting of the bucket in two sub-buckets that cover exactly the half of the keyspace covered by the original bucket. The splitting process is exemplified in Figure 9.2. Additionally, to avoid a too unbalanced

[1]The default value of k in the original Kademlia protocol is 20

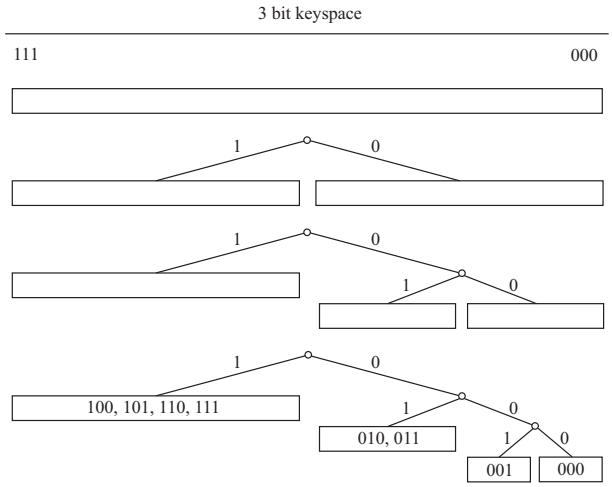


Figure 9.2: k-bucket split in a toy keyspace of 3 bits. In the last split state the IDs that can be contained by the buckets are shown.

expansion of the route table, the Kademlia protocol allows the split also for all the buckets residing at depth d in the tree for at most $b-1$ times[2], when the condition $d\%b = 0$ is satisfied.

Routing protocol

Kademlia defines four Remote Procedure Calls (RPCs) that any node can call on any other peer. They are used namely to ping a contact (PING RPC), ask for the k known contacts nearest to a certain ID x (FIND_NODE(x)), ask for resources marked with the key x (FIND_VALUE(x)) or store a resource y under a given key x (STORE(x,y)). All messages are sent using UDP transport, for efficiency reasons. Except PING, mostly used in the route table maintenance, all the RPCs are used during the routing and content storing procedures.

To find the responsible nodes for a key an *iterative lookup* is performed. The initiator of the lookup selects from the route table the α contacts nearest to the lookup key x and sends a FIND_NODE(x) message to each of them. In response, the initiator receives at most $k \cdot \alpha$ new contacts. Among those contacts she iteratively sends new FIND_NODE(x) messages to the α closest to the lookup key. The algorithm terminates when the k nodes closest to x whose contacts have been learned during the process have been all contacted with a FIND_NODE. These nodes are the ones responsible for the lookup key. Once they have been detected, FIND_VALUE or STORE RPCs can be used to ask them to return or to store a resource.

It is easy to see that if each node has at least one active node in each of its buckets and each node knows all its closest neighbors (which are quite realistic assumptions), for any hop the distance is reduced by half, so that the routing algorithm will converge in $log_2(distance)$ steps to the ID closer to the target.

[2]The default value of b in the original Kademlia protocol is 5

9.1.2 Kad

The most successful implementation of the Kademlia DHT is the Kad network, that has been used in the eMule file sharing software [172]. In 2008, the estimated number of users in Kad was between 3 and 5 millions, sharing more than 80 millions unique resources [315]. The most relevant difference that Kad adds to the Kademlia protocol is the *Kad tolerance zone*.

Whenever a node A wants to share a resource x, it issues a routing request (actually multiple routing requests in parallel) to find the node responsible for ID_x. During the iterative request, node A collects IDs of other nodes and performs the STORE() operation on the first 10 nodes it encounters that have at least the 8 most significant bits in common with ID_x. This 8-bit zone is called the tolerance zone for ID_x. As a consequence, the search operation is different. If node B wants to retrieve resource x, it issues an iterative search for ID_x; whenever it receives the IP of a node that falls into the tolerance zone, it issues a FIND_VALUE(ID_x) for ID_x to that node; in response, a list of possible resources is returned (for example, a list of IP addresses that contain the resource). The iterative search for ID_x and the FIND_VALUE(ID_x) for ID_x keep going in parallel until: 1) a timeout is fired, 2) the routing to ID_x expires because no new nodes are found, or 3) a maximum number of resources are returned by the FIND_VALUE primitives. This value depends on the kind of resource being searched.

Kad contains many distinct kinds of resources, among which there are keywords and files. Keywords are used to reconstruct the ID of a file: when a client wants to share a file x, it calculates the hash of the file ID_x, as well as the hash of each of the words that compose the file-name, and stores the association (ID_{word}, (ID_x, "complete title")) in the network. When searching for a file, the peer issues separate RPCs for the hashes of all the words that are in the filename. From the intersection of the responses the the most appropriate one is selected by the user herself. Then a look-up for ID_x is issued to retrieve the list of IP addresses that contain that file. Then, the peer can directly contact the given IP addresses to download the file.

Whenever a node shares a file x with ID_x and a certain file-name, it is responsible for periodic republication of the IDs of the file and of the title-words, each republication being performed on 10 nodes in the corresponding tolerance zone. This means that if a file is shared by many users, its ID is republished (refreshed) frequently, and a higher number of nodes in the corresponding tolerance zone stores the IP of the owners. When a node receives a store message, it starts a timeout, and after a fixed amount of time, the resource is erased if no update is received. The expiration time and the content refreshing time depend on the kind of resource, varying from 5 hours for keywords to 24 hours for files. Lastly, Kad network uses an hash space of 128 bits instead of 160.

Each node periodically sends PING packets to the nodes in its buckets and purges the ones that do not respond, so that nodes with longer lifetime are the last to be removed. Since a new contact is added only to a non-full bucket, it is not trivial to perform bucket poisoning, i.e., send unsolicited pings to a certain node in order to be present in its bucket.

9.2 Attacks on DHTs

Full decentralization and self-adaptation to high churn conditions makes DHTs extremely resistant to random failure of nodes. However, their structure is very vulnerable to targeted attacks. In the following we discuss a simple attacker model and the most known attacks that can be directed under this model. This discussion applies to the general definition of DHT given in Section 9.1 and is independent from the particular implementations.

Attacker model

Opponents that we take into account are users that aim to break off or degrade the DHT service or to exploit the potential of the network to attack another peer or a target service outside the DHT. We suppose that an attacker is able to perform the following operations with minimum computational effort:

- run a large number of node instances on the same computer
- spoof its nodes' $NodeId$s and their network addresses
- intercept and alter the communication flow between any two nodes
- conspire together with other malicious peers in order to accomplish coordinate attacks

This broad freedom of action allows an attacker to effectively put off a large spectrum of attacks against every structural element of the network. Next, we classify and inspect the attack categories that such adversary can put off. Several classifications of DHT attacks can be found in literature (e.g., [308, 72, 330]), and many of them focus on the exploitation of arbitrary $NodeId$ assignment and on the routing procedure compromise. Our purpose is to consider a wider range of attacks, including also those against DHT storage functionality and Man In The Middle attack.

Sybil attack

In a structured P2P network, $NodeId$s are generated locally from each node instance, arbitrarily. As we stated before, we suppose that a user can generate many node instances on the same machine, with as many different $NodeId$s. Multiple identities belonging to a single user are called *Sybils* [96]. Such behavior is in itself harmful because it undermines the redundancy property of the DHT system: if many different nodes are instantiated on a single machine, the disconnection of the computer would lead to failure a large number of nodes and, consequently, to the unavailability of replicas of the content kept in their storages.

However, Sybils can be used by an attacker to put off massive and organized attacks. Assigning identifiers near to a target key to a sufficient number of Sybil nodes, an attacker could be able to intercept and discard most of the lookup requests for that key, thus censoring the content stored in the DHT for that key. More in general, Sybil entities are used to avoid involving multiple users or using many computers, thereby reducing the attack cost.

To avoid Sybil attacks, the system must therefore ensure that different identities refer to separate entities (intended as users or computers). Douceur [96] noticed that a honest entity can recognize a Sybil only gathering information from three different sources: a trusted authority, other untrusted peers (*indirect validation*), or itself (*direct validation*).

In the case of direct validation, a honest peer can make sure that n different identities refer to n distinct entities by challenging them with a computational puzzle that could not be solved by a number of entities less than n in a time shorter than a given threshold [284]. This method results very unpractical, both for the heterogeneity of the computational capacity of the nodes, both for the challenge simultaneousness requirement. Exploiting assumptions on the underlying physical network in order to identify nodes which are instantiated at the same physical position have also been proposed ad a possible countermeasure [338].

In indirect validation the correct node may accept only identities that have been validated by a sufficient number of introducer peers. The obvious flaw of this approach is that a group of deceptive nodes can introduce Sybils as valid peers.

Several ad-hoc protocols have been proposed in more recent years; for example, Jyothi et al. [30] propose a scheme in which each node is dynamically associated to a monitor node that moderates transactions involving its twin node, thus making ineffective any Sybil attack attempt. Another recent approach leverages the information of the social network of users to detect the untrusted contacts established by Sybils [358].

The introduction of an access control service which assigns certified identifiers only to authenticated entities seems to be the most effective solution. This service could be implemented through a centralized authority as well as with a distributed system. Steiner et. al [316] propose the use of a centralized agent to bind nodeIds to IP addresses after having received a proper user request sent via SMS; nevertheless, since a user of a node with dynamic IP address have to send an SMS at every IP change, the central agent is an undesirable single point of failure because its disconnection would prevent bootstrapping nodes to join the network.

Routing attacks

By appropriately altering the correct message routing procedure of a set of target nodes, an attacker could be able to make the DHT unserviceable to them, or she may even cause all victim's outgoing messages to be redirected to a parallel, corrupted network. These effects are usually obtained through a practice known as **routing poisoning** which consists in injecting *ah-hoc* entries in the victim's route table. Route table poisoning comes out easily in a DHT environment, because of the *push-based* approach in routing information updating: since the route table is built and updated on the basis of unsolicited messages received from other peers (such as the periodic publishing of neighbors route tables), an attacker could easily replace most of the entry victim's route table entries with fake information.

In particular, a node is very vulnerable to poisoning attacks during its bootstrap phase, because the initial information to initialize the route table entries is supplied only by the bootstrap node, that can easily fill the joining node's route table with fake entries.

Another particular kind of routing attack is the **lookup misdirection attack**). A lookup can be simply deviated by a malicious node that replies to a lookup request with contacts that are not nearer to the responsibility area for the lookup key. In a worse scenario the malicious node can even claim that she is responsible for a keyspace subset she is not.

To limit the effectiveness of poisoning-based attacks, Castro et al. [72] propose to assign random certified nodeIds to peers and to establish strong constraints on the set of values that a specific route table entry could assume. If a malicious node cannot choose arbitrarily its identifier and it can insert only its own contact into the victim's route table, then the index poisoning would be unfeasible in practice.

A different, unusual approach to defeat this threat is proposed in the design of the *Maelstrom* DHT [87]. In Maelstrom, P2P network activities are divided into time periods called *epochs*, determined by an external time authority. At the end of an epoch, route tables are flushed and nodeId are recomputed on the basis of the epoch identifier. Route table resets, combined with a given limit to the routing information update rate, make ineffective any routing poisoning attempt, because the adversary cannot effectively complete a routing poisoning before the end of the epoch. However, the DHT activity is completely dependent on the time authority.

Eclipse attack

The eclipse attack [307] is a form of routing poisoning which aims to separate a set of victim nodes from the rest of the overlay network. A set of attacker nodes reaches its purpose by fooling correct nodes into adopting malicious nodes as their peers, with the goal of entirely dominating their neighbor set. If carried out successfully, the eclipse attack allows an attacker to mediate most overlay traffic, thus effectively eclipsing correct nodes from each other's view. Briefly, the eclipse attack is performed by an attacker that tries to intercept all the requests directed to a specific resource and redirect them to a fake resource. It has already been shown that this attack is possible in Kad if the attacker controls a set of IDs close to the target ID (i.e. ID_e) before the publication of ID_e [316].

The eclipse attack can be also intended as an attack against the storage; when it is targeted to overshadow content stored on DHT, making them inaccessible to lookup requests, it takes the name of *node insertion attack*. It is carried into effect by initiating a substantial number of nodes marked with identifiers numerically close to the ID_e, so that all lookup requests for ID_e are routed to these nodes. Once received a lookup request they can answer with a fake content or they can send no reply message at all, effectively hiding the content to the DHT.

Singh et al. [307] propose the *anonymous auditing* heuristic to contrast eclipse attack. This technique is based on the definition of the *backpointer set* of a generic node X, that is the set of all the nodes that keep a reference to X in their neighbor sets; in order to eclipse a huge portion of the network, the attacker aims to become the neighbor of a large number of nodes, hence the size of its backpointer set is much larger than that of a correct node. A peculiar protocol allows a honest node to query anonymously its neighbors for their backpointer sets and then to recognize an attacker if its set does not contain a reference to the querier or if its cardinality is higher than

a certain threshold.

It should be noted that all the variants of eclipse attack can effectively take place only if the attacker nodes may select their nodeId with an "ad-hoc" strategy, so they can populate the target nodes' route table entries or the keyspace depending on whichever is the key associated to the node or to the content they aim to eclipse.

Storage and DDoS attacks

A node is free to insert into the DHT any content bound to arbitrary lookup keys, which are chosen at application level. Attackers can disseminate fake resources or harmful information. We talk about **index poisoning attack** when bogus resources are deliberately spread to the nodes responsible for those lookup keys (the *index* nodes); this attack is particularly effective in P2P file sharing systems. If index poisoning is massively carried into effect, the ratio between the number of fake and true resources can soar, hiding the original content from the lookup process [283]. When resources contain references to other resources that are intentionally corrupted or fake we talk about **pollution attack** [195], a index poisoning closely related attack.

Index poisoning is the main mean to perform DDoS attacks [239]. In content sharing applications, for example, nodes publish the network addresses of content providers in the DHT. If an attacker spreads references to a very popular item, specifying a target service as the source of that item, she will cause the redirection of all the item requests to the victim, easily realizing a TCP flooding. For this reason, solving the index poisoning problem in DHTs provides also a robust shelter against DDoS attacks.

Lu et al. [205], proposed *RIEP*, a secure content publish protocol to limit the risk of a DDoS attack in a DHT-based file sharing network. RIEP is based on IBS and assumes that every peer has a private key bound to its public identifier; every published content includes the publisher identifier signed by the publisher's private key. Before starting the file download, a node that receives a signed content verifies that it is signed by the same identity that appears within the content. A mismatch between the identity of the signer and the identity of the source reveals a DDoS attack attempt. However, this solution is strictly bounded to the file sharing applications.

More in general, rating systems could be used to evaluate the quality of a content to detect any attempt of poisoning. Ross et al. [196] point out that applying rating systems to resources, thus allowing users to mark content depending on its quality, are often ineffective and could be easily deceived by a polluter, while rating systems applied to users, that is adopt a reputation systems that allows users to evaluate peers' behavior and to blacklist bad users, could be more valid. Ideally, one would like to have a distributed reputation scheme that responds to attempts to evade detection by changing identity.

Man In The Middle attack

In our adversary model an attacker is able to overhear and modify the content of messages flowing between any two endpoints. This is a tangible occurrence in real overlay networks, because of the use of *buddies* to manage the communication to nodes who reside behind a firewall or a NAT

service. Basically, a peer hidden by the NAT establishes a TCP connection with a chosen buddy node who acts as an application gateway for incoming requests; as a consequence, the buddy can easily alter the content of the responses addressed to the hidden node. A detailed description of how buddy system works in Kad is given by Brunner [66], while Steiner et al. [317] report a crawling analysis of Kad showing that the portion of nodes that relies on a buddy is very significant.

To avoid MITM an authenticated channel between the two communication endpoints must be established and the integrity of exchanged data must be assured. MITM is related not only to P2P systems but to every distributed service; for this reason, many studies about MITM resistant two-ways authentication protocols are available in literature [54].

9.2.1 Simulations of Eclipse attack

To fully understand the security issues that affect the Kademlia and Kad DHTs, we perform a simulation-based analysis focusing on the evaluation of the impact of the eclipse attack on the network. The simulations are realized with the Omnet++ simulator (`http://www.omnetpp.org`) on a single-layer Kad network. By means of simulations can be better evaluated the effectiveness and the consequences if such an attack, to design adequate countermeasures. We analyze to kinds of attacks, i.e., the pure eclipse attack and a variation with randomly chosen nodes. It will be shown that the impact of the attack depends mainly on three factors: the number of attacking nodes, the possibility of the attacker to choose IDs arbitrarily and the possibility to start the attack before the resource is published.

Simulations have been performed observing the behavior of a single tolerance zone containing 4,000 nodes, corresponding to a network composed of approximately 1,000,000 nodes. To lower computational load, not all the network was equally populated. Lifetime of the nodes and other system parameters were chosen consistently with the measurements reported in previous work [66, 315]. The deviation of measured results between two simulations with distinct random seeds is extremely low (below 2%) so in the figures are reported the profiles given by an average one. After a set-up phase, the attacks start at second 10,000.

Accordingly to work by Steiner et al. [317] the simulations reveal that if the attacker is able to choose the IDs arbitrarily and to place those IDs in the network before ID_e is published, the attack is 100% successful. This means that almost all the requests for ID_e are captured by the attacker; to achieve this, the attacker needs to control just 8 IDs. In Kad, users choose their own IDs. If IDe is an ID of a keyword, then it is predictable and can be eclipsed prior to its publication.

Through simulations it is possible to estimate the impact of eclipse attacks even in the case when ID_e is a file hash. In this case, ID_e it is not known before its publication and the attack starts after it has been published. This means that there is a race condition between the attacker spreading fake resources and the other nodes spreading correct ones. In this scenario the attack does not have a high impact if not supported by a bucket poisoning strategy. With such a strategy, the attacker IDs send unsolicited pings to the nodes in the tolerance zone of ID_e so that such IDs are added to the buckets of correctly behaving nodes. In Figure 9.3 (right) the eclipse attack is started by 8 attacking nodes after the publication of ID_e together with bucket poisoning. As time

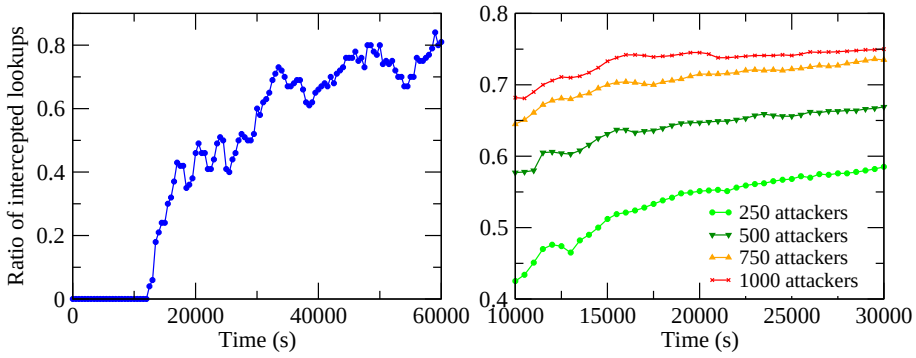


Figure 9.3: (Left) The ratio between the number of search requests for IDe and the number of search requests that are intercepted by an attacker node. Measures are sampled every 500s and averaged over a sliding window of 10 samples. (Right) The ratio between total number of lookups in the tolerance zone and lookups intercepted by the attacker IDs

passes, the attack becomes more effective, but even on a long time span it does not reach a total eclipse. This is explained by the fact that for every obtained request, the resource is republished by the receiver on ten nodes in the tolerance zone, as it would happen for a downloaded file in Kad. In a nutshell, there is a concurrency between the republication of correct resources and fake ones, that prevents the attack from being completely successful.

A variation of the eclipse attack has been considered when the attacker is not able to determine its own ID. Previous work [76, 37] suggests that IDs could be chosen as a hash of network parameters (IP address and UDP port). Our simulations show that even in that case, if the attacker owns a large number of IDs, the eclipse can still happen with high probability. With a single IP address, a node can obtain 216 different IDs uniformly spread. This yields about 250 IDs per tolerance zone. In Figure 9.3 it can be seen that even when the attack starts after the publication of ID_e, with bucket poisoning, the percentage of eclipsed requests is still high. Moreover, in this scenario there is no concentration over a single ID so that attacker could eclipse any resource in the tolerance zone. This means that the attacker could perform a monitoring activity, substitution of resources or DoS against any resource in the tolerance zone with the success ratio shown in Figure 9.3 (left).

The last attack that has been evaluated through simulations, is a DDoS attack against a third party. This last scenario can be used to produce a different attack. As pointed out previously, if all the attacking IDs answer to FIND_VALUE RPCs with the IP of a victim node (even outside of Kad), this node will be flooded by requests coming from multiple sources [322]. This will produce a Distributed Denial of Service that is very hard to stop. From our simulations we were able to estimate that with FIND_VALUE RPCs the attacker receives an average of 4.5 frames for each frame sent by the attacker. This means that the upload bandwidth resources of the attacker are multiplied by 4.5.

In short, Kad vulnerability to eclipse is given by two main factors: the possibility for the attacker to choose its own IDs and the possibility to have multiple IDs. In the following part of the paper it will be shown how the introduction of a certification service can help Kademlia (and

consequently Kad) to be more resistant against the attacks that exploit these factors.

9.2.2 Applying countermeasures

The previous overview highlights the main instruments that can be taken to develop a comprehensive defense against all the mentioned attack categories. Specifically:

1. $NodeId$s must be generated randomly. The possibility of arbitrary $NodeId$ selection should not not be given to any node.

2. The possibility for a user to generate many nodes on a single machine must be severely restricted or made as expensive as possible.

3. The procedure for routing tables updating should provide appropriate constraints. A peer should be able to insert into a second peer's routing table only its own contact. The routing table in which its contact is added should not be determined by the message sender.

4. During the bootstrap, the node must acquire routing information from trusted sources.

5. A unique, strong user identity must be associated to each node. This identity must be certified and verifiable by other peers so that a system for the evaluation of user behavior can be realized.

6. The communication protocol between nodes must be two-way authenticated and must ensure the integrity of messages.

As we stated before, our attack analysis is quite general because it does not take in consideration any detail of a specific DHT. So, Kademlia protocol is included in the previous security considerations. The following Chapter describes the model of a DHT-based system that is able to fulfill all the mentioned points.

Chapter 10

Likir, an identity-based DHT

10.1 Toward a secure DHT

Recent research on reliability of P2P overlay networks has focused mainly on three aspects: scalability of fully decentralized architectures [79], incentive mechanisms against free-riding [138] and Distributed Hash Tables (DHTs) security problems. As we remarked, security is very relevant, especially when proposed architectures are addressed to the implementation of applications that are more critical than file-sharing and have strict requirements of stability. This need becomes even stronger if applications use the DHT to store sensitive user data.

Besides proposing ad-hoc solutions for specific attacks, research on DHT security came up with several secure DHT designs that tried to address a broader spectrum of security issues [330]. Some attempts focused on the customization of the routing protocol to patch security flaws. S/Kademlia [43], for instance, is a secure Kademlia-based routing protocol where some key features of the original protocol are adapted with custom security devices. S/Kademlia limits free $NodeId$ generation by using crypto puzzles in combination with public key cryptography. It extends the Kademlia routing table by a sibling list and it reduces the complexity of the bucket splitting procedure. Its lookup algorithm uses multiple disjoint paths to increase the lookup success ratio. It also allows the DHT to store data in a safe replicated way to reduce impact of attacks against the storage.

Other solutions provide the enforcement of the notion of identity or of the responsibility function used to map resources on nodes. Chord-based DHT Myrmic [339], for instance, adopts a "root verification protocol" that allows to check that the responsibility function is correctly applied. This is accomplished by the combined action of a trusted authority, which issues certificates specifying the responsibility keyspace area for a node, with a set of designed witness nodes, that checks that the responsibility area is respected. A very similar approach is adopted also in NeighborhoodWatch DHT [48], where a third party authority issues signed tokens to certify the responsibility of a node on a keyspace subset. Both of them do not offer any specific defense against storage attacks and they have the evident drawback of the single point of failure: if the central authority is down, nodes cannot join the network.

Another identity enforcement approach has been proposed by Ryu et al. [289] in their secure DHT. They leverage the node identity assignment procedure to reduce the impact of some dangerous attacks. An ID assignment protocol based on identity-based cryptography is presented, showing that the id-based cryptography is a suitable and affordable technique that preserves scalability by introducing a slight overhead. The described bootstrap procedure is accomplished through a weak authentication method (i.e., based on a callback to the presented IP address) that has to be executed at each join.

Nevertheless, despite all the efforts spent in securing such systems, currently there is no widely accepted and general countermeasure to all the attacks surveyed in Section 9.2. Our contribution to this research area is the definition of *Likir*, a secure extension of the Kademlia protocol that strongly mitigates the most detrimental attacks to the DHT through the embedding of a strong identity notion at overlay level. In fact, the most evident structural flaw that originates most of the DHTs security issues is the uncontrolled assignment of node identifiers.

In Likir, a strong node identity allows to build a secure, authenticated communication protocol. By exploiting a Certification Service, peers are given verifiable and certificated identifiers, which are tightly coupled with the user identities. Doing so, most of the security issues are overwhelmed, or at least strongly mitigated, under our very general adversary model. Likir security services are transparent, enabling developers to consider decentralized implementations of distributed applications, without having to cope with the security devices that are used in our architecture. Furthermore, the Certification Service is not a single point of failure like in previous work, because nodes can keep joining the overlay also during its downtime. In addition to the improvement of the DHT security features, embedding a strong identity management into the overlay allows to provide *services* to the application level.

The advantage of including identity in the overlay is then twofold. First of all, the same identity-based services provided at lower level can be reused by several applications. Furthermore, *integration* between different applications is made easier because of the sharing of the same identity. This has a significant impact on the design of new applications, because it allows mash-ups based on the explicit link that resources have with their owner.

Last, the overhead of the security layer on the computational cost and on the network delay of the DHT is affordable and does not impact sensibly the overall network performance.

10.2 Architectural model

Likir, Layered Identity-based Kademlia-like InfRastructure, is the architectural model of a new DHT system that offers both a very high protection level from all the most common attacks against structured P2P networks and a simple framework supporting identity-based services. The means by which Likir reaches these goals is the delegation of user identity management to the overlay network layer. Likir architecture is structured in two main modules, shown in Figure 10.1.

The first is a user registration service, accessible via Web, which returns to the user the certified identity that will be used to mark its node on the overlay network. The service is composed of a

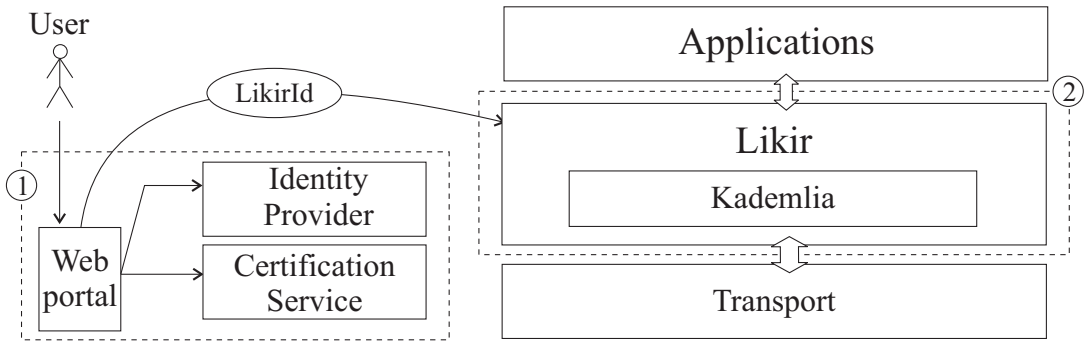


Figure 10.1: Likir architectural model. The components of the user registration service and the extended DHT protocol are marked with numbers 1 and 2 respectively. The user registration procedure produces a *LikirId* that is used by the DHT node for authenticated interactions with its peers.

web portal, which interacts with an Identity Provider for user identity verification purposes and with a Certification Service for the creation of the certified identity. The latter is a DHT protocol which extends the Kademlia protocol, encapsulating it. This layer provides an essential set of simple and general purpose API for developing any kind of application.

Next, we inspect such two components, defining in detail purposes, functions and interactions between them. We made only a pair of assumptions, useful for the following discussion; we suppose that each user has a pair of RSA keys and an *OpenId* account. Finally, we adopt the following notation.

A, B	:	*Likir* users
$NodeId_A$	:	node A's DHT identifier
$UserId_A$	:	A's OpenId
K_A^+, K_A^-	:	A's public and private key
$E_{K_A^-}(msg)$	:	message signed with the key K_A^-
$H(o)$	:	hash code of the object o
ts	:	timestamp
$a\|\|b$	:	concatenation of strings a and b

10.2.1 User registration service

In order to use Likir services, the user must fulfill a registration procedure; Likir architecture provides a web portal for this purpose.

A generic user A is authenticated by the registration website using the OpenId protocol. This implies that A sends its OpenId (the $UserId$) to the registration service, which in turn contacts a third-party OpenId provider, where A has a valid account, to validate the user identity (a detailed description of the OpenId 2.0 framework is given in [279]). Once the *UserId* is validated, A sends her public key to the registration portal through a simple submission form supplied on the website. The OpenId and the public key are then forwarded to a trusted entity, the *Certification Service*

(CS). We pass over the specific structure of the CS and we handle it as a black box service, which peculiar function is creating signed identifiers for newly joining users; we only suppose that it owns an RSA key pair $\langle K^+_{CS}, K^-_{CS} \rangle$.

Upon A's request, the CS binds A's *UserId* to A's public key and to a random 160bit string that represents the DHT identifier of the Likir node. The binding is made through the production of a cryptographic token which is then sent to A through a secure channel:

$$LikirId_A = E_{K^-_{CS}}(NodeId_A || UserId_A || K^+_A || ts_{exp})$$

CS keeps track of the association between $UserId_A$ and $LikirId_A$, to prevent subsequent requests from causing the production of unnecessary signatures. Only when $LikirId$'s validity is near to its expiration (determined by ts_{exp}) the CS must create a new $LikirId$.

When user registration procedure is successfully terminated, A can instantiate its own Likir node just supplying the $LikirId_A$, its key pair and the CS's public key, that we suppose to be publicly available on the registration portal.

It is very important to notice that, once a user has obtained her $LikirId$, she does not need to contact CS until her signed ID validity expires. If the CS fails, the user registration service becomes unavailable but the network activities are not affected, because the users that previously obtained their $LikirId$ can join the overlay without querying any central service. Since the ts_{exp} can be chosen to last even many years, we can state that the CS is not a real single point of failure of the system.

Of course, the CS infrastructure and the maintenance of the registration portal have a cost that have to be sustained by some project promoter. Such infrastructural cost could be low enough to be set up also by no-profit organizations like universities (realizing a single-server PKI and a OpenId-compliant web service is relatively cheap). However, also commercial promoters could be interested into supporting the project because of the potential revenues from advertisements. Since Likir can be exploited also as a platform for the creation of a decentralized, privacy-aware online social network, we believe that its potential attractiveness to consumers could be high. An alternative way to finance the registration service cost could be to ask for a micropayment for each new $LikirId$ issued.

10.2.2 Node interaction protocol

To join the Likir network, a node must execute the Kademlia bootstrap procedure, namely performing a lookup for its own $NodeId$ starting querying an online bootstrap node. If the node is not aware of any alive contact (e.g,. it is performing its first bootstrap), it can send a proper request directly to the CS, which responds with a signed *bootstrap list*. The CS controls at least one active Likir node which executes a periodical probing task (simply performing lookups for random $NodeId$s) in order to learn of fresh contacts. However, it is important that every node keeps track of a substantial number of previously known contacts to use as bootstrap nodes, to avoid the CS to be flooded by bootstrap list requests.

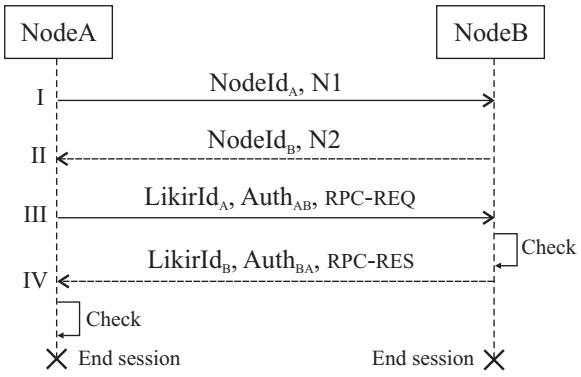


Figure 10.2: Likir node session between two peers. The four-ways interaction is shown as a sequence diagram. The token exchanged are listed on the arrows.

A node A can successfully send a Kademlia RPC to a node B only if both A and B follow this four way *session*.

I $A \rightarrow B : NodeId_A, N1$

II $B \rightarrow A : NodeId_B, N2$

III $A \rightarrow B : LikirId_A, Auth_{AB}, \text{RPC-REQ}$

IV $B \rightarrow A : LikirId_B, Auth_{BA}, \text{RPC-RES}$

$N1$ and $N2$ are randomly generated nonces; RPC-REQ and RPC-RES fields are respectively the request and response RPC defined in Kademlia. Messages sent at steps I and II must be somehow marked differently (e.g., distinct opcodes), to differentiate the request from the response.

$Auth_{AB}$ and $Auth_{BA}$ are two authentication tokens structured as follows:

$$Auth_{AB} = E_{K_A^-}(NodeId_B||N2||H(\text{RPC-REQ}))$$
$$Auth_{BA} = E_{K_B^-}(NodeId_A||N1||H(\text{RPC-RES}))$$

Figure 10.2 shows the session message exchange. Steps I and II just accomplish a preliminary nonce exchange. Messages III and IV are thoroughly symmetric; the Kademlia RPC is sent with the $LikirId$ of the sender and with a signed authentication token. The $Auth$ contains the addressee $NodeId$ (to avoid replay attacks), the previously received nonce (to assure the freshness of the token) and the RPC hash (to protect RPC message from modifications). We observe that freshness of authenticators can be granted also using timestamps instead of nonces thus avoiding the preliminary message exchange. However this would require to assume at least a loosely synchronization of nodes' clocks. Later, in Section 10.3, we show how this protocol assures authenticity of messages.

The RPCs exchanged during the session follows exactly the Kademlia RPCs specification, except for the STORE RPC. In Likir, Kademlia STORE request message is enhanced to prevent the

disownment of the insertion operation. It is structured as follows.

$$\text{STORE RPC} = k||content$$
$$content = Obj||Cred$$
$$Cred = E_{K_A^-}(UserId_A||k||H(Obj)||ts||ts_{exp})$$

The STORE RPC embeds a *content* and its lookup key. The content is composed by the actual object, which is application-specific, and by a signed credential $Cred$. $Cred$ token includes all the useful information on the published object: its ownership ($UserId_A$), its lookup key (k), its SHA-1 hash to grant its unalterability, the publish time (ts) and its validity period (ts_{exp}).

Content store and retrieve operations on the DHT can easily built upon the defined primitives, according to the Kademlia lookup protocol; we call those PUT and GET, respectively.

10.2.3 Replacing RSA with IBS

Likir protocol exploits conventional RSA cryptography for token signatures, however, in agreement with the identity-centered Likir design, an Identity Based Signature (IBS) approach could be adopted as well.

IBS is a cryptographic technique that allows to compute a key pair whose public counterpart could be obtained from an ASCII string. This cryptographic paradigm allows a user to verify a signature of another user just knowing her user ID. The original IBS idea, based on the RSA function, was presented by Shamir [304], but it was subsequently revisited by Boneh and Franklyn [59] and Cocks [84], who used bilinear pairings [212] for efficient Identity Based Cryptosystems (IBCs) design. The presentation of IBS mathematical background goes beyond the goals of this work, however we give a brief overview on how such paradigm works.

In IBS, when a user A wants to send a signed a message to a user B, the following steps must be executed.

1. **Setup**: a trusted third party, the Private Key Generator (PKG), creates a pair of *master keys*: a public key MK^+ and a private counterpart MK^-.

2. **Extraction**: A presents her identity (Id_A) to the PKG, who produces a private key K_A^- from MK^- and Id_A; the new key is then sent to A through a secure channel.

3. **Generation**: using its private key K_A^-, A creates a signature s on message m and sends (m, s) to B.

4. **Verification**: B checks whether s is a genuine signature on m using Id_A and MK^+.

Likir can profitably take advantage of an IBS scheme. Since the $UserId$ must be sent in every communication session, the recipient of a request or response RPC always knows the sender's user name, so, using IBS, the Likir protocol overhead could be noticeably streamlined because the information of the user public key in the $LikirId$ could be omitted, and the $UserId$ could be used directly to verify tokens signatures.

Nevertheless, IBS has two main drawbacks. The first concerns efficiency: the known IBS algorithms [118] in current implementations (e.g., Stanford PBC Library (`http://crypto.stanford.edu/pbc`)) are slower than RSA, both in signature and in verification phases. The second (the most severe) is the *key escrow* property: the PKG is a genuine single point of failure, because if an attacker takes possession of the private master key, she could generate the private keys related to every $UserId$, thus violating the whole cryptosystem.

For these reasons, the current Likir implementation adopts a traditional public key scheme. However, the scientific community is still very active in IBC research, therefore the possibility of replacing RSA with IBS should be taken into account.

10.3 Security analysis

Likir contrasts the well known security threats against DHTs with its two main architectural elements: the enhanced node interaction protocol and the CS service. Next we discuss more formally how such features impact the security level of Likir protocol. To show the Likir protocol effectiveness against poisoning, Sybil, MITM and storage attacks we must proof two properties.

Property 1. ***Node authentication.*** *A node can communicate with others only providing its own* $LikirId$.

Proof. Suppose an attacker node X who is pretending to be A during a session with node B. Clearly, since the node session protocol requires to provide a valid proof of the $LikirId$ ownership (the $Auth$ token), X cannot simply reuse the intercepted A's $LikirId$ to spoof its identity, but a valid couple of $LikirId$ and $Auth$ is needed.

X cannot produce a valid $Auth_{AB}$ because the $Auth_{AB}$ token must be signed with the private counterpart of the RSA key included in the $LikirId_A$, that is unforgeable because it is signed by the CS; so, the entity that is able to produce a valid $Auth_{AB}$ is A only. So, X has only two ways to counterfeit its identity: to intercept and reply a valid $Auth_{AB}$ or to trick A into produce a valid $Auth_{AB}$.

In the first scenario, X must intercept an $Auth_{AB}$ that contains the same nonce received by B (at session step I or II, depending if X is the session initiator or not); but if nonce's size is big enough and a good pseudo-random generator is used by Likir clients, the probability that X can find such $Auth_{AB}$ is negligible.

In the second scenario, X must convince A to build an $Auth$ containing the nonce received by B. This can be easily achieved by establishing a proper session with A, but the token thus obtained would be an $Auth_{AX}$, and not an $Auth_{AB}$. To get a proper $Auth_{AB}$, X must pretend to be B during the session with A, but this creates a cyclic dependency: to pretend to be A in a session with B you must pretend to be B in a session with A. □

Property 2. ***Message integrity.*** *Every message flow alteration attempt causes the abort of the session.*

Proof. If an attacker modifies the data at session steps I or II, the *Auth* tokens sent in the following steps will be no more valid, because they include all the fields of the preliminary nonce exchange messages. *LikirId*s and *AuthId*s are unalterable because they are signed; the same is for the RPC message, because its cryptographic hash is included into the *Auth*. So, the message flow between two nodes is unalterable by a third party. □

Property 3. ***Content authenticity***. *Any node retrieving a resource from the DHT can determine its original publisher, its original lookup key and whether the resource has been modified after the publishing phase.*

Proof. The only valid resources that can be considered during the retrieval phase are those bound to a valid certificate, signed by the owner. The unforgeable certificate includes the original lookup key, the hash of the original resource and the publisher's $UserId$, which is sufficient to check the authenticity of the resource. □

From Property 1 follows straight that $NodeId$ cannot be generated arbitrarily and cannot be spoofed. In Kademlia a new routing table contact is added at the end of a communication session, if there is enough room in the proper bucket. Since the overlay communication is authenticated, the $NodeId$ are randomly chosen and cannot be spoofed, and the position of the new contact in the routing table is determined locally on the basis of the sender's $NodeId$, an attacker cannot insert an arbitrary contact in an arbitrary position of the routing table. Therefore, any routing table poisoning based attack is unfeasible.

Property 1, combined with Kademlia design, prevents also lookup misdirections. The nodes considered during the lookup process must be directly probed; since the node responsibility function depends only on the $NodeId$ and since the $NodeId$s cannot be spoofed, the initiator knows for sure what key responsibility area she is addressing to. Of course, denial of service (e.g., a node replies to a lookup query with valid contacts whose $NodeId$s are not close to the target) is always possible, but this is an inherent problem of the distributed lookup mechanism. Nevertheless, the Kademlia lookup procedure is pretty resistant to such attack because favors the retrieved contacts that are nearest to the target. According to the Kadmelia protocol, at every lookup step, the α nodes whose $NodeId$s are the closest to the target are probed. If, during the process, a honest node is queried, the k contacts returned will be likely closer to the target than the contacts returned by other non-collaborative nodes, so the next α nodes to be queried will be chosen from this set (since, usually, $\alpha < k$).

Protocol authentication and $NodeId$ randomness limit also the sybil attack impact. To run many different nodes, as many *LikirId*s are needed, but if the user subscription service provides valid techniques to avoid the registration phase automation, the effort needed to produce many sybils can be arbitrarily increased. For example, during user subscription phase, a credit card number can be required. Different identities could be issued for the same user, but the CS can limit the number of different *LikirId*s issued for the same credit card number, thus making very expensive the hoard of a huge number of identities. Moreover, even if an attacker has many nodes

under her control, she cannot position them in specific key space areas, because the $NodeId$s randomness.

Property 1, together with Property 2, shows the protocol resistance to MITM; furthermore, the bootstrap list provided by the CS for the first bootstrap is signed, thus partition attacks during the join phase are avoided.

Finally, Property 3 grants the ability to discern polluted content from genuine entries. Combined with Property 1, it allows to avoid any future overlay interaction with all the polluters detected, thus preventing attackers to keep injecting bogus resources in the DHT. Further details on the strategies of exclusion of the attackers from the network are given in Sections 10.5 and 11.4.2.

10.4 Performance evaluation

Compared to Kademlia, the Likir protocol introduces an overhead that affects both the number and the size of messages exchanged between nodes. Besides, the cryptographic effort spent during a node session due to the signature operations increases the computational load on every single peer.

In order to quantitatively evaluate the performance decay due to additional messages, enlarged message size and cryptographic overhead, we opted for a test in a real, large-scale distributed environment. We run small Likir and Kademlia overlay nets on PlanetLab network (`www.planet-lab.org`), in order to compare the time effort needed for PUT and GET primitives in both protocols and to measure the average impact of cryptographic operations on the whole Likir session time.

The reader should note that a *scalability* test is not needed here, for three reasons mainly. First, we do not modify the Kademlia routing protocol neither its routing table management policy; thus, the number of hops for a lookup operation in Likir is exactly the same as in Kademlia. Second, the number of messages sent during a Likir session is incremented by a *constant* number, compared to a Kademlia session; this implies that the number of messages per lookup still grows logarithmically with the network size, like in Kademlia. Last, the cryptographic operations impact on the nodes that perform RSA checks and signatures but clearly do not burden the network with any additional traffic.

For these reasons, Likir has *by design* the same scalability properties that have been shown for Kademlia. Of course, the time needed for a lookup operation is greater in Likir if compared to Kademlia, so we want to quantify this gap in a real network environment. Prior to this, we present also a static analysis on the message size overhead and on the cryptographic primitives cost.

10.4.1 Spatial and cryptographic overhead

The size of a Likir message is greater than the size of ordinary Kademlia RPC due to the addition of $LikirId$, $Auth$ and $Cred$ tokens. In Table 10.1 the whole set of elements that composes these tokens is shown, together with their size; furthermore, the specific composition of each signed token is given, together with their total size. We suppose that 1024 RSA keys are used.

Element	Size	LikirId	Auth	Cred
$NodeId$	20	•	•	
$DHTkey$	20			•
$UserId$	128	•		•
K^+	128	•		
$Signature$	128	•	•	•
$Nonce$	16		•	
$Hash$	20		•	•
ts	8	•		••
Total size:		412	184	312

Table 10.1: Cryptographic token size (bytes)

RPC	Request	Response
PING	596	596
FIND-NODE	596	596
FIND-VALUE	596	$596 + 312 \cdot n$
STORE	908	596

Table 10.2: Spatial overhead (bytes)

RPC	Sender	Receiver
PING	$gen + 2 \cdot check$	$gen + 2 \cdot check$
FIND-NODE	$gen + 2 \cdot check$	$gen + 2 \cdot check$
FIND-VALUE	$gen + (n + 2) \cdot check$	$gen + 2 \cdot check$
STORE	$2 \cdot gen + 2 \cdot check$	$gen + 2 \cdot check$

Table 10.3: Cryptographic primitives used in each RPC

Once crypto token size is assessed, we can easily calculate the size overhead on each Kademlia RPC. Every RPC contains at least a *LikirId* and and *Auth*, which form the message header. In addition to this, the STORE RPC request contains also a *Cred* bound to the content to be stored, and the FIND-VALUE RPC response payload contains a *Cred* attached to every content returned to the querier. Of course, the number of resources per FIND-VALUE response is variable due to the availability of objects bound to the requested key stored in the queried replica node, so the overhead for such RPC can change; to seize this variability, we define n as a variable representing the number of *Cred* per FIND-VALUE response. Table 10.2 summarizes the given considerations, showing the overhead for every RPC. It is worth noting that the additional header size is smaller than 1KB in the worst case, and the FIND-VALUE payload dimension overhead is linear with the number of retrieved resources.

The node interaction protocol requires also that both sender and receiver generate and verify signatures; for the sake of brevity, we refer to *gen* and *check* respectively for a signature generation and a signature verification. Table 10.3 summarizes the number of cryptographic primitives to be performed by a node during a whole session, for every RPC. We deliberately ignore the SHA-1 hashing operations due to the non-influential cost. The n additional *checks* reported to FIND-VALUE RPC client side represent the *Cred* verification of all retrieved content. The impact of such primitives on the RPC session time is discussed in the next Section.

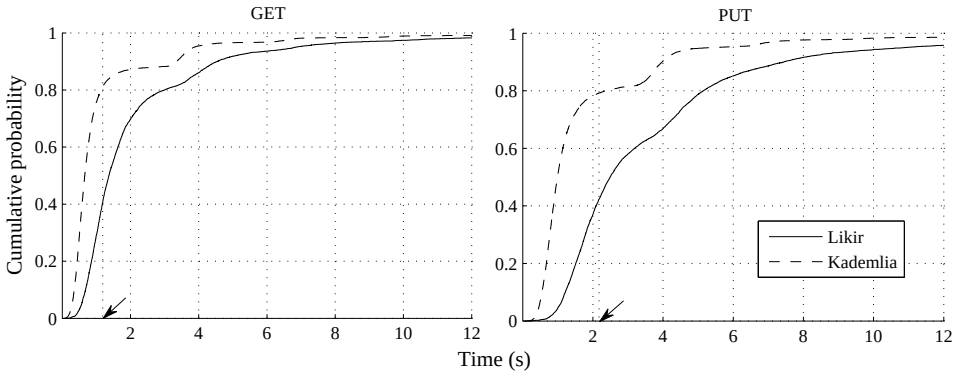


Figure 10.3: CDF of GET and PUT times in the PlanetLab experiment

Operation	Likir			Kademlia		
	E	$\mu_{\frac{1}{2}}$	σ	**E**	$\mu_{\frac{1}{2}}$	σ
GET	2,291	1,366	2,930	1,276	659	2,402
PUT	3,877	2,408	4,123	1,844	1,091	2,626

Table 10.4: Likir session duration (milliseconds) for both PUT and GET in the PlanetLab experiment

10.4.2 Network emulation

We built a Kademlia implementation simply by replacing the Likir node interaction protocol with the classic Kademlia protocol on our Likir Java implementation; the Kademlia parameters k and α we chose are respectively equal to 4 and 2.

We bootstrapped 250 overlay nodes on as many PlanetLab nodes; we used the support of a centralized server for bootstrap lists distribution. Then, each Likir node executed 25 PUT and 25 GET, randomly interleaved, on random keys. The sequence of called primitives followed a Poisson process; the temporal distance between two events was determined by an exponential distributed random variable. The same experiment was made for the Kademlia configuration.

We measured the whole execution time of each PUT and GET. The cumulative distribution function of these times is depicted in Figure 10.3. We observe that, in both plots, the relationship between the two curves is different depending on the time range taken into account.

In a first interval, from 0 to the value highlighted with the arrow, the Kademlia curve assumes values that are more than double than the Likir curve; in the second interval, up to infinity, the curves get asymptotically closer. This happens because in short lookup procedures the cost of Likir cryptographic operations assumes a non-negligible weight respect to the overall PUT/GET time, while in the second range the network delay prevails on the time spent in signatures generations and checks. As expected, PUT requests are slower than GET because they require an additional hop to command the found index nodes to store the content. The mean lookup hop number is 2, so the number of step required is 2 for the GET and 3 for the PUT.

To give a more precise estimation on the overhead introduced by Likir we need some statistics (presented in Table 10.4). We notice that the PUT and GET mean and median time in Likir are

roughly double respect to the same primitives executed with the Kademlia protocol. The standard deviation assumes always high values because the huge network latency variability and the different number of hops of the lookup processes. This is the result we expected, since in a Likir session four messages must be exchanged, compared with the two messages of a Kademlia session. This suggest that, on average case, the cryptographic overhead has a little impact on the overall time.

To give a more precise estimation of the impact of checks and signatures on the session time, we measured the mean time for *gen* and *check* operations on a PlanetLab node. Since we know the lookup hops mean, the average number n of retrieved resources in a GET operation (we observed $n = 4$) the number of primitives needed in PUT and GET for each hop (Table 10.3), and the mean time needed for cryptographic operations, an estimation of the mean time spent on the local node for cryptographic primitives can be easily done. We calculated an overhead of about 172 ms for GET and 347 ms for PUT. These values are less then one tenth of the total operation time, and, however, they even do not impact in full on the total PUT and GET time because of the parallel nature of the lookup process.

In conclusion, the network emulation results shows that the predominant Likir overhead is given by the additional message exchange, that necessarily doubles, on average, the basic DHT operations execution time.

10.5 Likir API

The Likir's interface to the application offers a simple and essential set of primitives. Since in this Chapter we focus on the *security* of the overlay layer, for the sake of a smooth presentation, here we report a restricted Likir API that does not include some parameters that enables access control policies over resources stored on the DHT. We will discuss the complete API (see Table 11.2) contextually to the additional privacy services presented in Chapter 11.

In the following, we denote the node N of a user $userId$ as N_{userId} and we suppose that key is an identifier of the keyspace.

1. BOOTSTRAP*()*. It joins the local node to an existing Likir network by contacting previously known peers as bootstrap nodes. If no peer is known, send a proper request to the CS to gain a fresh bootstrap list

2. PUT*(key, obj, type, ttl)*. It is the basic insertion primitive. It looks up the DHT nodes responsible for storing objects marked with *key* and puts the binding between *key* and *obj* in their storages. *type* is a string denoting the application-specific type of the object and *ttl* determines the expiration time of the content.

3. GET*(key, type, userId, recent)*. It queries the DHT for resources marked with *key*. Only objects marked with a certain *type*, or belonging to a user identified by $userId$ could be retrieved. Please note that even if an identity-based resource filtering could be implemented also in classical DHT by attaching proper identity tags to published items, in Likir the ownership of resources is verifiable in a *secure* way thanks to certificates. The *recent* boolean

parameter allows to retrieve only the last published version of the objects. This is useful since a common strategy to release an update of a content on a DHT is to publish a new version of the object with same key and type.

4. BLACKLIST*(userId)*. It adds the specified user to a local blacklist. Every new incoming message from $userId$ will be discarded; this is possible because overlay interactions are authenticated.

In the following, we refer to PUT and GET also to denote the messages (RPCs) originated by the corresponding API call.

Very sharp resource retrieval can be made through the index side filtering facility. If all GET parameters are set, at most one resource is returned (i.e., the last resource inserted by the specified user, under the specified key and type). Obviously, identity-based resource filtering could be achieved simply tagging the stored resource with a label that specifies the owner identifier. However, such method is vulnerable to storage poisoning attacks, therefore it cannot grant the resource ownership. On the contrary, Likir protocol assures verifiability through certificates.

A Java implementation of Likir is available online (`likir.di.unito.it`). It follows faithfully the Kademlia specification except for the node interaction protocol, for the addition of a nested-map data structure which implements the content storage and for the management of the blacklist.

10.5.1 Applications interaction and integration

In classical DHT-based applications, like file-sharing (e.g., eMule), despite the preliminary index-side filtering that storage nodes perform based on the keywords specified in the search query, the content retrieval process usually returns a big quantity of results that often contains several almost-equivalent versions of the same resource. This is perceived as an acceptable output by the user, who simply selects the resource that best fits her needs among the returned set in order to start the download.

The same situation leads to a completely different scenario for identity-based applications that might be used in OSNs. In this case users often search for *specific* resources that belong to *people they know yet* (their "friends" or "contacts"). For this reason, they expect to receive a very precise response to their queries; for example, a search for the last wall post on my friend's blog is not supposed to return the whole set of posts of that person or the last wall updates of the whole list of the friends of mine.

Instead, a very sharp resource retrieval can be made with Likir thanks to its filtering facilities. As pointed out before, if all the filtering parameters are set in GET, at most *one* resource is returned. This possibility allows to manage the access to frequently updated information (e.g., user status modification). Two relevant, unique aspects characterize this kind of filtering. First, the identity-based filtering is performed in a fully verifiable way thanks to certificates paired to the resources. Second, the whole filtering is made by index nodes, thus relieving the local application from *any* filtering responsibility.

Algorithm 3: LiCal events management

```
1 Node n = new Node (UserId_A)
2 n.bootstrap ()
3 Object Calendar_A = createUsrEvents (...)
4 String weekID = getCurrentWeekID ()
5 Int replicas = put (UserId_A|| weekID, Calendar_A, "Cal", defaultTS)
6 List <Object > res = get (UserId_B|| weekID, "Cal", UserId_B, true)
```

It is worth noticing that widgets are not forced to communicate exclusively through the DHT and can establish direct connections if needed. In this case, the distributed storage can be used for preliminary Diffie-Hellman exchange in order to establish a secure out-of-band connection. Since the key agreement protocol is performed on a fully authenticated layer, the new secure channel will be also authenticated as well as encrypted.

Identity-based resource retrieval has a very good implication also on integration between different widgets. Since identity is managed at overlay level, all the data published by the same node are marked with the same user identity, regardless of the nature of the widget that generated the content. Furthermore, suppose that widgets publish their API, describing the internal structure of the items they manage together with the lookup keys and types associated to them (this is a realistic assumption given the fact that publishing an application API is a practice that is already adopted by the vast majority of Web 2.0 services). Doing so, integration becomes easy, because just by the invocation of a simple method every widget can gather and aggregate content from other (possibly different) widgets owned by the known social contacts.

To give a practical demonstration on how Likir API allows a quick development of applications and their integration on an identity basis we consider two simple demonstrative applications.

First, we consider **LiCal** (**Li**kir **Cal**endar), a client that allows a user to publish her commitments and to consult the public events of her friends. Algorithm 3 shows that few code lines must be executed to startup a node (lines 1,2), publish user A's weekly arranged events (lines 3–5) and to consult her friend B's public events (line 6). We omit details about the event structure and we suppose to deal with a time interval of one week, even if of course different time granularities can be chosen.

Quite differently to DHT-based file sharing clients (e.g. eMule), when the calendar client queries the DHT it is not interested in receiving a huge amount of results. If the weekly events of a known friend are looked up, only a specific entry, inserted by a definite identity is wanted. Moreover, only the last calendar update is sought. Setting all the GET filtering parameters each index node return only one result (the most recent one), so the DHT data retrieval can be realized with an accuracy that is uncommon for classic DHT services and the application is relieved from *any* filtering task.

Even if quite embryonic, LiCal represents an example of a straightforward identity-based application. It shows how, in principle, synchronization problems afflicting ordinary calendar manager systems can be solved using a reliable DHT approach. In fact, a LiCal client can be interfaced with a commonly used calendar client (e.g., Apple iCal, Microsoft Eudora), in order to access our

Algorithm 4: LiCha bootstrap

```
1 Object Options_A = get (UserId_A||"options", "LiCha", UserId_A, true)
2 Int replicas = put (UserId_A||"contact", contact, "LiCha", defaultTS)
3 Object friend
4 for contact_k in options.buddylist do
5     friend = get (UserId_K||"contact", "LiCha", UserId_K, true)
6     friendEvents = get (UserId_K|| weekID, "Cal", UserId_K, true)
```

data from different machines without a centralized provider (e.g., Google, Plaxo).

As a second demonstrative example, we introduce **LiCha** (**Li**kir **Cha**t) that is a more mature social networking application we developed (`likir.di.unito.it/applications`) using the Likir API. LiCha is an instant messaging client whose architecture is fully decentralized. The conventional central server that, in classic chat clients, retains all user information is replaced with the DHT.

Two kind of resources are managed: the user *options*, containing the buddylist and other local user preferences, and the client *contact*, that is trivially a TCP socket address of the chat service and a status specification (offline/online). Algorithm 4 shows how local options are retrieved from the DHT (line 1) and how the client network contact is published (line 2). Contacts are then retrieved (lines 3–5). Finally, the LiCha client pings each online friend to inform them of its status. When LiCha clients exchange contacts each others, they can simply start chat sessions without involving the Likir layer.

LiCal and LiCha can be profitably integrated. For example, the LiCha buddylist can be enriched displaying the daily events of those users that are also LiCal users. Such feature can be achieved with a single code line (line 6), supposing that the rules to build the correct LiCal lookup key are known. This basic example shows how any cross-application integration can be implemented in Likir; knowing the $UserId$ of a friend and the correct lookup keys production rules, a generic module can easily retrieve public information related to any other Likir application used by that friend.

The extreme openness of this scheme, where every application can potentially cooperate with any other module, enables an implementation of a OSN free from the so-called *information silos problem*. Paradoxically, in fact, even if the user information leaks from centralized OSNs, trampling on the user privacy rights, it is often difficult to share data between different OSNs or to reuse social applications in a profitable way, due to the heterogeneity of the platforms or to narrow content management policies [355]. Some centralized solutions, like the OpenId-compliant Global Social Platform [236], have been proposed to overcome this problem, but without receiving sufficient consensus so far. Instead, full decentralization and modularity allow to compose any kind of SNS as an arbitrary combination of cooperating applications and even many different OSNs can be linked together in the same way.

In the next Chapter we provide the details on how this idea is realized in practice.

Chapter 11

LotusNet, a privacy aware P2P OSN

11.1 Building P2P social applications

The high practical relevance of the privacy issues in OSNs has recently strongly attracted the interest of the research community and gave origin to a line of research focused on the realization of decentralized privacy-preserving frameworks suitable for SNSs. As well as to deal with information confidentiality and access control, one of the challenges in this context is to provide an efficient support for all those services that are easily provided by classic server-based architectures (e.g., data availability) but that are not trivial to be implement on a distributed layer.

Some research about the implementation of social application on P2P layers has been made even before the viral spreading of OSNs [211], but the idea of using a P2P framework to solve ONSs privacy issues is relatively recent [67, 26].

The *PeerSon* [68] (`www.peerson.net`) system is one of the first P2P design for a OSN. The goal of user information privacy is achieved through symmetric encryption of resources stored in a DHT. The P2P network serves mainly as a lookup service: once the two endpoints' contacts have been retrieved from the DHT, direct connections are established. When a friend is offline, update notifications are managed asynchronously through the DHT using a pull approach. Full decentralization and encryption prevent, respectively, the "Big Brother" effect and network crawling activities aimed to data collection. However, advanced access control features like highly dynamic group membership are not taken into account.

The *Safebook* [93] (`www.safebook.us`) P2P OSN focuses on resource availability as well as content privacy and end-to-end communications confidentiality. It combines a DHT with another P2P network called *Matryoshka*. Peers in the Matryoshka are connected by trust bonds: user items are stored at highly trusted neighbors. The DHT is used to store the contacts of the Matryoshka members and encryption is used to preserve content privacy. The Safebook design provides also the presence of a trusted, offline identification service to avoid Sybil attacks.

A similar approach is adopted by Narendula et al. [240]. Here, the problem of content availability on fully decentralized social networks is faced with a P2P storage model based on the concept of trust. Items replicas are stored at a set of nodes trusted by the content owner, called Trusted Proxy Set (TPS). The initial selection of TPS nodes and their churn dynamics are handled accordingly to the nodes' geographical location, to place the data as close as possible to nodes that often access the content. References to TPS nodes are published on a DHT layer. Data on the DHT are indexed with a k-anonymization technique in order to grant both owner and content privacy.

Graffi et al. [134] propose a DHT-based storage with access control capabilities for social shared resources. Encrypted items are published together with several copies of the secret key; each copy is encrypted with the key of a users who have access permission for that resource. A drawback of this solution is that when the access control list of a specific content must be changed, a new updated item has to be built and stored again. User registration phase and secure P2P communications are inspected as well.

Participants discovery in fully decentralized OSN is discussed by Abbas et al. [1]. Authors define a gossip protocol, implemented on the Tribler network [264], for user discovery. When the target friend is unavailable and the search initiator goes offline, a set of online *helper* peers is delegated to perform a periodic probing activity aimed at the retrieval of the searched peer's contact.

Vis-a-Vis [302] is a scheme for decentralized OSN that aims to high content availability. Each use stores her personal data in a Virtual Individual Server (VIS) that is kept on the local user machine and it is replicated on a cloud infrastructure. When the desktop is offline, the cloud service is activated and the availability is not interrupted. Further replicas of the resources are hosted among the VISes of the social contacts, that are connected together through a P2P overlay network. However, the rental cost of the cloud service makes this proposal not very practical.

All these solutions focus mainly on the problem of access control or on single social services like contacts discovery. A real distributed OSN needs a wider design that considers *security*, *privacy* and *services* as a whole, providing a complete and coherent solution without the imposition of any constraint or assumption on the nature of applications or on the structure of the social network.

In this context, Likir is an ideal platform for SNSs mainly for two reasons. First, the security level offered by its protocol grants a very high robustness to the modules above, which is very important for this kind of applications. Second, Likir identity support better matches the SNS requirements rather than any other DHT. In fact, since services in OSNs are strictly coupled with user identity, a low-level management of user identifiers is useful for an identity-based integration of data from different applications.

In the following, we extend the Likir API and, on the top of that, we define a core set of services that add a privacy preserving layer over the secure Likir substrate and satisfy several critical requirements that any OSN should provide to its clients. Finally, we show how basic services typical of centralized OSNs can be easily and efficiently realized in our framework.

11.2 OSN requirements

A SNS can be defined, in its most general meaning, as a *customizable suite of inter-operable, identity-based applications*. In this context, every user composes its own combination of applicative modules, or *widgets*, into a customized application suite, where every widget can share data with other possibly heterogeneous, widgets running locally or remotely. Note that this is a very general model because no assumption is made about the nature of widgets or the structure of exchanged data, and both synchronous and asynchronous communications are allowed. Given this general definition, we inspect a set of desired privacy, security, and service requirements that are common to a very wide range of social widgets. Such requirements are at the basis of the architectural design of our distributed social framework.

11.2.1 Privacy requirements

Confidentiality. Any kind of content created by any widget should be accessible only to those authorized to have access. This property has a wide meaning. First, it implies the definition of access control policies that allow a flexible and fine-grained specification of grants. Furthermore, in a OSN context, *fully-customizable* confidentiality is the most effective patch to stop information leakage.

Ownership privacy. The creator or the owner of a content should be enabled to not disclose the ownership information to other users. The relational tie between a user and a content of a certain type could be in fact a very valuable information, whose disclosure could be potentially detrimental to the user privacy.

Social interactions privacy. Social ties and the data flowing on them can tell much information on the behavior of individuals and on the dynamics of groups in a networked environment [120]. For this reason, a user should be able to arbitrarily hide the interaction between local and remote widgets.

Activity privacy. The type and number of widgets that compose a user application suite must be considered sensitive user information, because they partly reveal the nature of user activity on the network. Flexible privacy settings should be provided to tune the exposure of the application suite composition to the public.

11.2.2 Security requirements

Channel authentication. Communication channels should be two-way authenticated, so that both the initiator and the recipient can check the identity of the partner. Unauthenticated channels are potentially vulnerable to social engineering attacks. *Phishing* aimed to sensitive data collection [159], for example, is mainly based on persuading the target user that the attacker is a known and trusted entity. Strong authentication, combined with an educated behavior of users, greatly reduces also the risk of identity theft, that is one of the main issues in today's Web-based OSNs [52].

Data integrity and authenticity. Information that is published by widgets and exchanged with remote entities must not be modifiable by any non-allowed user. The genuineness of content

should also be assured through ownership verifiability, to avoid making wrong associations between a content and a user that is not its owner.

Non-repudiation. Users are fully responsible of their actions on the network. In many distributed services, the agreement on social or micro-economical transactions passes through messages that are exchanged by the parties involved. The property of non-repudiation of the content ownership lays the foundations for traceability of user actions and is therefore a deterrent to frauds, to the spreading of false information on the network, and to *spamming* activities.

11.2.3 Service requirements

Content availability. Access to data should not be conditioned by the connection status of the owner. Even when the owner is offline, users with a proper permission should be able to access the data.

Flexible communications. End-to-end communication can be synchronous or asynchronous. A widget should be able to listen for live notifications and to recover messages that were sent during its offline time.

Easy integration. The *mash up*, namely the composition of existing building-block services into more complex applications, is at the basis of the Web 2.0 paradigm. For this reason, the interoperability between social applications should be facilitated as much as possible.

Search facilities. Users are interested in acquiring new contacts and in exploring the resources published in the OSN. Proper search engines should allow to find the desired items, but in compliance with privacy requirements stated above, if possible.

Reputation management. The collaborative environment of OSNs can often rely on reputation and trust notions to balance social interactions or negotiations. For this reason, widgets should be provided with common tools to quantitatively express their perceived reputation of other participants and to convey their reputation beliefs to remote widgets.

11.2.4 The wall, the fence, the garden

Privacy, security, and service requirements (summarized in Table 11.1) lay on three orthogonal axes; an ideal OSN platform should preferably meet at the same time all the requirements that are listed in the previous sections, thus maximizing the level of fulfillment for each of the three properties. However, it is clear that such peak cannot be reached due to the partial conflict between some requirements that lays on different axes. For instance, security properties like traceability and authentication may often conflict with ownership and social interaction privacy, which can in turn affect the effectiveness of search facilities or the significance of a reputation system.

To effectively depict this trade-off we can use a *walled garden* metaphor. Users in OSNs are like gardeners who work for their plants and flowers to thrive. As long as the garden is open to the external environment it receives the benefit of sun and rain and new seeds can root on its ground, carried by the wind or by other gardeners visiting. But, without any shelter, the garden also suffers from the bad weather and can fall prey to vandals and thieves. Gardeners can rise

Privacy *(Wall)*	**Security** *(Fence)*	**Services** *(Garden)*
Confidentiality	Channel authentication	Content availability
Ownership privacy	Integrity and authenticity	Flexible communications
Interaction privacy	Non-repudiation	Easy integration
Activity privacy		Search facilities
		Reputation management

Table 11.1: Summary of the social network service requirements, partitioned in the three main categories.

fences to turn away malicious visitors and walls to protect the garden from the weather, but high **fences** (*security policies*) prevent good visitors to easily access to the **garden** (*services* and user *resources*) and thick **walls** (*privacy settings*) can deprive the flowers of light and water; this is the *gardener's dilemma*.

Next we describe *LotusNet*, a framework for a privacy-aware implementation of SNSs. LotusNet is designed to reach a good trade-off between the three aspects. It is based on secure interaction protocols and enables a set of powerful services, but its architecture is not bound to a single system-defined privacy setting. On the contrary, LoutsNet provides the users with the ability to tune their own privacy configuration with fine granularity over a range of possible privacy policies, thus giving wide freedom to the user to open doors and windows in their own privacy wall.

11.3 Architecture

In LotusNet, widgets are layered on the Likir P2P network and can interact by exchanging objects through the DHT. The goal of LotusNet is to build higher-level functions upon Likir and realize the requirement-driven OSN model defined in Section 11.2.

The API offered by the Likir middleware represents the first step toward the goal, however we need to build higher-level functions upon it. The resulting idea is depicted in Figure 11.1: a custom suite of widgets relying on a layer of social network services. In our design such services are an extension of the Likir primitives and offer higher-level functionalities useful for OSNs. Such services are directly based on the API of the overlay node, which encapsulates the identity management and authentication features. The DHT cloud is the primary medium for P2P communications.

In the following, we inspect the three main aspects that realize this idea. First we present a general framework for widgets interaction and integration, then we discuss in detail the access control facilities of our architecture and finally we define a set of service modules that compose, together with the Likir interface, the complete API for widgets. During the dissertation, we highlight the architectural aspects that satisfy the requirements we defined in Section 11.2. Contextually to each service described, we will also define the extended and final version of the Likir API,

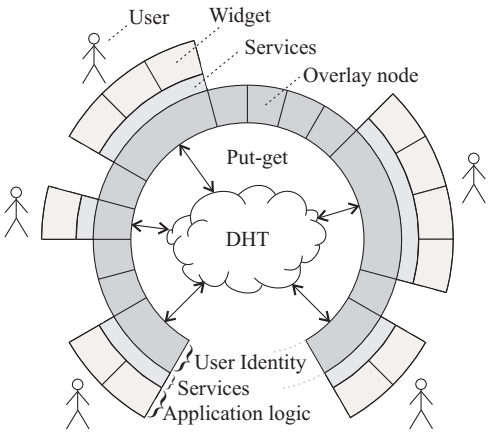


Figure 11.1: Conceptual scheme of the LotusNet social network service

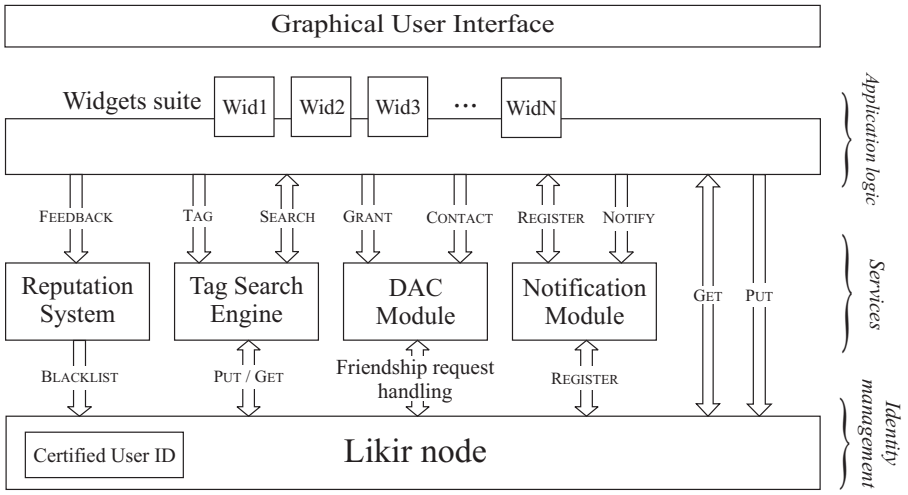


Figure 11.2: Architectural scheme of the LotusNet platform

with new operations and parameters explicitly supporting the access control at overlay level and demultiplexing services for inter-widget communications.

The detailed architecture of a LotusNet client is depicted in Figure 11.2; every element of this scheme will be expounded in the following.

11.3.1 Building the social graph: access control and contact discovery

In Section 10.5.1 we outlined a scenario of maximum interoperability, where every application can potentially interact with any other and access the whole information stored in the DHT. Even if this flexibility grants a very high level of customizability of the OSN structure, it totally lacks privacy. In order to preserve privacy, the shared information which is potentially available to everyone must be channeled into the communication pipes that participants create by establishing social ties. In other words, we must model the social acquaintance graph that links together OSN users and then we must limit the resource sharing only to the pairs of linked users. Furthermore, users should be

able to define with a fine granularity the portion of the personal information that is shared with arbitrarily-defined groups of neighbors.

Cryptographic access control techniques suitable for groups with dynamic membership has been extensively studied in the literature [116, 117]. In particular, solutions based on *key regression* schemes manage the eviction of a group member by redistributing to remaining members a new key that is used to encrypt new items but that can be used as well to get the previous group key. Under the assumption that a former member can have access to all the data published before her exclusion from the group, a *lazy revocation* [31] policy can be adopted: an old item is re-encrypted with the new key only when an authorized member modifies it, thus avoiding the re-encryption of all the items at every membership change.

Even if this strategy may be suitable in UNIX-like shared file systems, it is too less flexible in a OSN context, for several reasons. First, since the vast majority of items (e.g., posts, photos) which are shared in social network are written once by their owner and never modified, lazy revocation is often not applicable. Furthermore, even if, in principle, users formerly granted to access the content could have saved it locally, we argue that preventing a user whose grant is revoked to download the old data would be even a better guarantee for privacy that would be appreciated by many users. Finally, the most important point is that the mentioned key management techniques are not suitable for *overlapping* groups, commonly used in OSNs: if an item is accessible by several different groups it must be encrypted several times, thus greatly increasing the complexity of keys and groups management.

To provide a more flexible access control we recur to signed *grants* to specify permissions. Grants are associated with social contacts and not with shared resources, so their number does not grow with the quantity of resources owned or with the number of rules in the privacy policy. A grant certificate, produced by a user A for a user B is composed as follows:

$$Grant_A(B) = \{A||B||regExp||expireTime\}_{Sig_A} \tag{11.1}$$

It contains the identities of the owner and of the granted user, an expiration time and a regular expression that is a compressed list of all the allowed content types. The token is signed by the issuer.

In practice, the use of grants is made in the Likir API. When user A wants to publish any resource, it calls the Likir PUT with an additional binary parameter *public* that indicates whether the published content is accessible by anyone or just by granted users:

$$\text{PUT}(key, obj, type, public, ttl). \tag{11.2}$$

On the other hand, when user B wants to access a protected resource submitted from A, it adds the grant she received from A to the parameters of the GET primitive:

$$\text{GET}(key, type, userId, recent, Grant_A(B), sizeonly). \tag{11.3}$$

We also add the *sizeonly* option. If it is set, the response message will include only a list of pairs (t, c), where t is a content type stored at *key* and c is the counter of the number of *public* objects

of type t stored at the index node under key. This may be useful in some applications where the number of people submitting some content is more relevant than the content itself (we show an example in Section 12.3).

Of course, grant certificates can be used if the entities that have the duty to store the user data are able to verify the validity of the grant. The key idea that allow grants in our setting is the use of authenticated channels for the overlay communications. In fact, when an overlay index-node receives a request for a protected resource, it can ask for a valid grant before returning it, being able to securely verify if the identity of the querying peer is the same identity specified inside the grant. Note that the index-node can check the signature validity because, during the content insertion phase, the publisher's $LikirId$ which contains her public key is sent along with the object. Finally, the regular expression allows to specify permissions for an arbitrary subset of resource types.

Revocation of grants is implemented through expiration. The problem of choosing a proper life span to balance best the grant renewal cost with the maximum period during which a user with a revoked permission can still access to a protected resource has not a single optimal solution. In a fully-customizable framework, such duration should be chosen by the applications or even by the user itself, depending on the trust she places on the granted friends. Anyway, since the number of grant is limited by the number of contacts (that is reasonably small in the vast majority of cases), we suggest that short durations (e.g., one week) can be used.

Grants are very flexible and powerful, but they do not hide published content from the eyes of the index-nodes, that can mine their local storages as they wish, even without a proper certificate. To shelter user information from this potential privacy breach, content are encrypted. Note that in this case encryption does not imply a complex key management system. In fact, each participant can use a *single* encryption key to protect the full set of its data; the key is shared with the known contacts and it does not need to be replaced when the access control policies change.

An access control mechanism based on grants associated to users, instead on permissions attached to resources, draws implicitly the edges of the social network. With grants, the social network topology should not be explicitly mapped anywhere: the existence of a $Grant_A B$ means that a social tie has been established between A and B. Of course, the nature of ties can differ depending on the set of capabilities that the corresponding grant specifies. Besides, the asymmetrical structure of grants allows to build both *directed* and *undirected* social networks, depending on whether grants are reciprocated or not.

Implementation

In LotusNet, the *Discretionary Access Control Module* (DACM) is responsible for the management of the individual social connections and to set privacy policies by assigning grants. The DACM is layered directly on the Likir node and has a very simple behavior. At its startup, it creates a daemon listening on a TCP port and puts its address on the DHT, using a lookup key extracted from the user identifier. Then it enters in a passive state, waiting for incoming TCP connections or

for calls performed by local widgets. In particular, DACM's API to applications is the following:

$$\textsc{contact}(userId) \tag{11.4}$$

$$\textsc{grant}(userId, regExp) \tag{11.5}$$

$$\textsc{getGrant}(userId) \tag{11.6}$$

$$\textsc{getSelfGrant}() \tag{11.7}$$

When a widget W, in the user A's application suite, wants to find a new friend B, it calls $\textsc{contact}(B)$[1]. This method triggers a DHT lookup for B's DACM contact. If the contact is found, it means that B is on LotusNet. However, at this stage, nothing else is known about B, not even the widgets she has installed on its client. In order to determine if B is using widget W, the $\textsc{grant}(B,\hat{}\{W\backslash.\})$ function is called[2]. Doing so, a direct message containing a grant certificate for B is sent to B's DACM; this is interpreted by B as a new incoming friendship request. If the request is accepted, B reciprocates it by sending a proper grant for A. An analogous interaction occurs between widgets for the periodical renewal of grants.

For the sake of brevity, here we omit many less relevant implementation details. For example, here A directly sends a grant to ask for B's friendship, but more complex interactions could be implemented as well. For example, A may want to release the grant only if it will be reciprocated, or even more complex transactions can occur. A very general framework for resources negotiations which can be possibly applied to this case is described in [312].

The call of GRANT(userId,regExp) is used also to update the regular expression for a formerly granted user, in order to extend or reduce its permissions; in this case, a new signed grant is produced to replace the old one. The DACM stores in a local database both received and released grants. Methods $\textsc{getGrant}(userId)$ and $\textsc{getSelfGrant}()$ are used, respectively, to get the grant received by a social contact from a local database and to obtain a self-signed grant to access to the information stored by the local widget on the DHT.

Just for completeness, we note that this API can be profitably extended to manage grant distribution also to local widgets. If every widget is supplied with a grant that contains only the minimal permissions that allow its correct activity, *trojan horse* widgets are prevented to fetch private information stored on the DHT by other widgets in the same application suite and spread it publicly on the network without any permission.

11.3.2 Tuning the privacy level: lift up the walls

To show to what extent the privacy requirements listed in Section 11.2.1 are satisfied by the LotusNet design, we discuss two attack scenarios whose main actors are LotusNet users with two different roles.

[1] We suppose that B's identifier is known. A specific indexing application for user searching could be built, however this implies the disclosure of some personal information that are revealed for indexing purposes (e.g., hometown, schools attended, etc.)

[2] We suppose that content types begin with the main widget's name, that we assume to be unique, followed by a dot. The regular expression is written in POSIX notation, and it means "any string beginning with *W*."

In the first scenario, the attacker is a generic node that aims to disclose the private information of a target user, from which she has not received any grant. The attacker can query the index nodes that are supposed to store the target's data, trying to find a breach in the access control mechanism. Supposing that index nodes behave correctly and properly follow access control protocol, confidentiality trivially holds because peers without a proper grant cannot access protected resources. Furthermore, supposing that index nodes return a generic "content unavailable" message in response to unauthorized requests for protected resources, an attacker cannot infer the type nor the index key of the content stored by the target user; so, owner and activity privacy are satisfied. Last, social interaction privacy is not breakable because the attacker cannot learn what is the set of users that are granted to access the private content.

The attacker can also try to infer some information on the activity of the target user by analyzing the incoming overlay network traffic. However, the messages that the attacker can possibly receive from the target or from peers interested in retrieving the victim's data are just Likir lookup requests. Lookup messages are used to locate the index nodes for a given item and their payload contains only the DHT lookup key. This little information is not enough to learn if the lookup requests are aimed to store or to retrieve a content, what kind of content is looked up and who is the owner of that content.

In the latter scenario, the attacker is an index node and aims to disclose private information from the content she stores and the incoming requests she receives. Since stored information is encrypted, confidentiality holds, but the other privacy requirements cannot be fully satisfied because they are in conflict with the authenticated P2P communication that the Likir layer provides. In fact, the index node knows the identity of the owner of every stored item because of signed certificates (owner privacy), it is aware of the item types because they are specified by the publisher in the PUT primitive for indexing purposes (activity privacy), and she can log all the identities of the peers who make a GET request for a certain item, thus inferring social relations between the owner of the requested content and every querier of that content (social interactions privacy).

The second scenario confirms the intuition of conflicting OSN requirements presented in Section 11.2.4: full privacy is not reachable if all the security and service requirements are maintained. However, even in this case, LotusNet grants a good level of privacy. In fact, since the location on the overlay is determined by the Kademlia identifier inside the LikirId, a node cannot arbitrarily position itself on the keyspace because the identifier is generated randomly by the trusted CS. Being placed in a random overlay position, a malicious index node cannot intercept the data belonging to a specific target user. Moreover, since different widgets presumably use several different lookup keys to remotely store their data, user data are randomly scattered on the DHT; therefore, recovering the full information about a user is practically unfeasible for an attacker. In a nutshell, the information held by a index node is little and fragmentary, therefore the risk of privacy infringement is very low and we believe that it can be considered acceptable in most of the cases.

However, to reduce further the risk of privacy violation for information that is particularly sensitive, LotusNet allows the *tuning* of the privacy level required by the widgets. This can be done simply storing the sensitive data among a set of trusted contacts specified by the user [93].

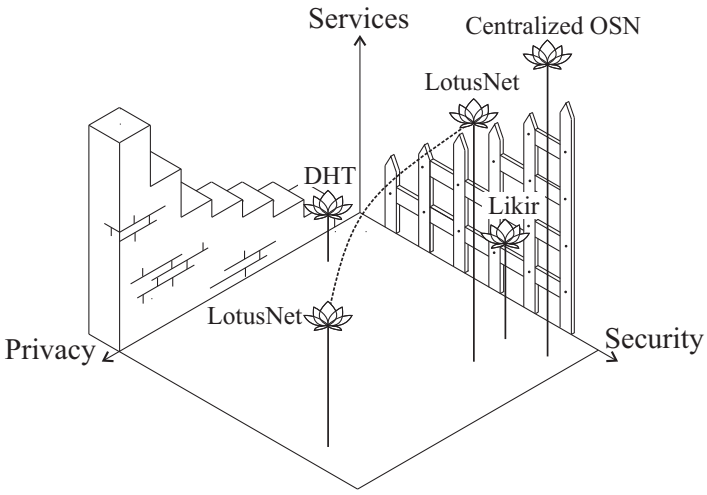


Figure 11.3: Trade off between security, privacy and services in OSNs depicted as a walled lotus garden. The Position of lotus flowers respect to the wall and the fence represent respectively the privacy and security levels, while the height of the flowers represent the potential number (and quality) of service that the user can benefit by sharing her data with others.

This approach is easy to follow because in Likir it is easy to get the overlay ID from a known user ID because they are permanently tied together by the signed *LikirId*. Practically, a trivial customization of the Likir PUT and GET primitives (PUT$^+$ and GET$^+$ in the following) which allow to explicitly specify the list of index nodes that are required to store/return the data is enough to implement this strategy. Of course, in order to not disclose the sensitive information to untrusted peers, index nodes must not apply the usual content dissemination policies for data published with this customized PUT operation.

Exploiting this opportunity increases the privacy level while preserving all the security properties of the Likir network, but at the cost of decreasing the quality of service. First, trusted nodes may not be enough in number to grant a good redundancy of the data and if uptime distributions are not very heterogeneous (which is likely if their geographical locations are very close), the availability of data is partly compromised. Second, since trusted nodes will be located in different regions of the overlay keyspace, the PUT$^+$ operation should perform a different lookup for every selected index node, thus slowing down the data save. Symmetrically, the GET$^+$ operation executed by data consumers should probably perform several lookups to locate at least one online trusted index node, thus slowing down the search procedure. However, it is worth notice that these drawbacks can be reasonably limited if the set of trusted peers is selected among the users who are potential consumers of the data and if this maximum privacy policy is applied only for resources with critical privacy requirements.

Figure 11.3 depicts the trade-off between security, privacy and service requirements recurring to the metaphor of the garden recalled in Section 11.2.4; we symbolically map different frameworks for OSN development on the three-dimensional space, in order to compare them. Specifically, OSNs based on classic DHTs can satisfy few privacy and security requirements and offer a minimal set of

services; the security level is considerably higher if the Likir DHT is used. LoutsNet benefits from the Likir security property and offers the opportunity to arbitrarily increase the privacy level from a minimum threshold (all published items are public) to a level where everything is protected by access control policies and content is stored at trusted peers; as stated before, as privacy level is increased, the quality of service (in terms of both efficiency of basic insertion/retrieval operation and of remote widgets that could benefit from that hidden information) decreases. In any case, privacy in LotusNet is always higher if compared to centralized social networks, since centralized SNS providers hold the complete information about every user in the network.

11.4 Services: grow the garden

A OSN does not live of privacy alone. The success of a SNS platform is mainly determined by the services it offers to users and applications. From our point of view, ongoing projects on P2P OSNs have not yet focused enough on the realization of complex social tools. In this Section we give a contribution in this direction by defining three core services of the LotusNet architecture: notifications, reputation management and high-level content indexing. The defined services extend the basic Likir API with operations 3-11, thus offering a high-level set of social primitives at the basis of the construction of complex SNSs. The set of methods offered by the LotusNet interface is summarized in Table 11.2.

11.4.1 Notifications

Social widgets can adopt synchronous and/or asynchronous protocols to communicate, depending on the application logic. If synchronous interaction between peers is needed (e.g., instant messaging application), it is reasonable to think that the two endpoints establish a direct connection to manage the data stream flowing in both directions. Conversely, asynchronous communications (i.e., *notifications*) fit better applications that are quiescent most of the time (e.g., wall posting service in OSNs); in this case, usually an always-online entity stores the message and delivers it as soon as the target user is online.

When using an overlay for message exchange, notifications can be stored on the DHT, but the *pull-based* approach at the basis of any DHT would force applications to continuously probe the network in search of new incoming messages. Several publish-subscribe schemes for DHTs aimed at equipping the overlay network layer with a *push-based* notification service has been proposed in the past [49]. Solutions proposed in literature are very articulated and include features like multicast communication, dynamic groups management, continuous complex queries, and so on [287, 164].

Here we use a simpler and lightweight unicast notification service that is suitable for end-to-end notifications between social widgets. The idea is rooted in the extreme precision that DHT routing algorithms have in locating the overlay node whose identifier is the nearest to the lookup key, even in presence of a high churn rate. In particular, it has been shown that the Kademlia lookup procedure is characterized by a very high precision [80]: regardless of the distance between the

querier identifier and the target key, the set of replica nodes located by the lookup contains the peer whose identifier is the nearest to the target key with probability near to 1.

Since in Likir every peer has a *fixed* position on the keyspace, every user can determine the exact overlay position of any of its friends. Notifications can thus be easily implemented by sending PUT massages with a lookup key which is equal to the Kademlia identifier of the user to be notified. Notification messages can be easily distinguished from all the other overlay message received because they are marked with a key which is *exactly* equal to the local node ID; since the Kademlia keyspace has a cardinality of 2^{160}, it is quite unlikely that random messages are mistaken for proper notifications.

We extend the Likir API with the operation REGISTER(*handle*), that simply specifies an application handle to which all the notification messages are passed. Furthermore, we introduce a *Notification Module* (NM) as a middleware between DHT notifications and applications. The NM registers on the Likir node as notification handler and offers two main methods to the widgets above:

$$\text{REGISTER}(applicationName) \tag{11.8}$$

$$\text{NOTIFY}(userId, applicationName, obj) \tag{11.9}$$

Using REGISTER, widgets are notified with all the incoming messages marked by the specified application type, while the NOTIFY operation sends an arbitrary notification object, marked with the application type of the notifying widget, to the target $userId$. The NM manages transparently the binding between the $userId$ and its corresponding Kademlia ID and probes the DHT at every startup to search for missed notifications.

11.4.2 Reputation management

Reputation is at the basis of many social dynamics, in real world as well as in OSNs. Online market places, question and answer bulletin boards and fora are just some examples in which reputation and trust play a key role in social transactions and in the selection of reliable partners. Very often, reputation is strongly *context-dependent*. For instance, a good photographer in a picture sharing system could be a bad movie reviewer and vice versa. For this reason, reputation systems are designed to operate within the boundaries of single applications and, typically, users are not represented with a single "karma" but with many, possibly conflicting, reputation*s* depending on the application they are plunged in.

However, introducing a *context-independent* notion of reputation could play a very important role in improving the service of collaborative P2P systems and of social networks in particular. Spamming, phishing, frauds and trolling are among the today's most serious threats for the security and usability of SNSs. Although these attacks are often made through the channels provided by single applications, their particular gravity has repercussions not only on the context of the application itself, but also on the quality of service of the underlying OSN: a social platform which does not effectively filters spam or does not provide any tool to detect fraudulent participants is indeed unreliable and, ultimately, not attractive to the public.

In centralized OSNs, malicious users like spammers could be traced and banned by the provider, which is able to detect anomalous behavior since it can monitor all the activity happening on the social network. Conversely, decentralized OSNs should be complemented with a handle to spread the information about misbehaving users that are detected by honest participants. In LotusNet, even if independent services can be accessed through different widgets installed in the application suite, the actions performed by a user in different social contexts are always referable to its *single* identity, determined uniquely by the $LikirId$. This feature allows to build a cross-application reputation system, where a reputation is associated directly to the misbehaving peer and not simply to a widget user.

The *blacklist* operation, offered by the Likir API, is meant to realize this idea. Interactions with blacklisted users will be avoided *at overlay level*. This means that any message or content, coming from any widgets belonging to the blacklisted user, will be automatically discarded by Likir. When a appreciable consensus is reached among the network, the misbehaving node will be not able to interact with the majority of honest peers, thus being confined to a dead network partition.

Relying on such low-level primitive, we can define a Reputation Module (RM) in our architecture's service layer. Its interface offers a single operation:

$$\text{FEEDBACK}(userId, score, evidence) \tag{11.10}$$

The idea is that an application can specify an evaluation on the behavior of a known user with a numeric score. Together with that, a proof of the good/bad behavior of the evaluated user can be stored by the RM. The proof is simply the content for which its behavior is being evaluated; for example, in a generic question and answers system, a post with inappropriate language can be an evidence of user misbehavior. Note that, since the ownership of content published on the DHT is verifiable, malicious users cannot forge fake proofs against honest peers, thus avoiding *social mobbing* phenomena. When the reputation score falls below a certain threshold, the bad user is locally banned using the BLACKLIST primitive.

The adopted approach has several advantages. First, our scheme strongly contrasts the *white-washing* phenomenon, thus limiting the risk that a user whose reputation is stained in a social context can rejoin with a different identity or can damage other systems where its identity is still unknown. Second, global reputation makes very high the cost of any bad behavior, thus being a powerful deterrent against malicious peers. Last, since in pure P2P OSNs single peers take upon themselves the critical responsibility of storing data from other participants, it is reasonable that malicious peers should be not trusted anymore to store user data; information spreading techniques that are implemented at overlay level are able to transparently replicate the data stored by the blacklisted user to another more reliable index node.

Banishment of polluters

We are not interested into define the details of a specific reputation module (i.e., how scores are managed), also because different schemes can fit better different types of OSNs (remind that several complex OSNs can coexist in LotusNet). Several suitable reputation schemes that potentially suits

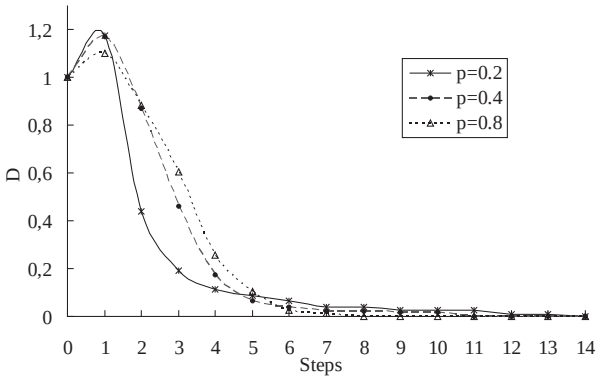


Figure 11.4: Reputation test results

our context can be found in literature [313]. Instead, the crucial point we want to underline is that the reputation information can be securely spread across the network thanks to proofs associated to users. Even a simple gossiping protocol could be effective: when the *RM* blacklists a user, it sends the proofs of its bad behavior to neighbors in the social networks that, in turn, can blacklist the same identity too and forward the information to their neighbors. Since, due to the *small-world effect* [341], the typical diameter of social networks is quite small, the information is diffused in few hops across a great portion of the network, thus rapidly excluding the malicious user from social activities.

We show this effect through an emulative experiment. Each peer runs a LoutsNet test widget with a very simple behavior: broadly, the peer periodically stores and retrieves resources from the DHT on random lookup keys; we suppose that the application is able to verify if a resource is polluted in a fully automated way, without interacting with a human user, so when a fake resource is retrieved, its publisher's $UserId$ is immediately inserted into the blacklist.

Each widget interacts with a local *RM*, just notifying it when a new $UserId$ is blacklisted and providing the related *evidence*. The *RC* stores a *evidence list* in a local database.

The *RM* behavior follows a simple, zero-tolerance gossip-based approach to spread local reputation information. Periodically, the *evidence* list is published in the DHT using a lookup key which is easily obtainable from the publisher's $NodeId$. Before the publishing phase, the *RC* retrieves the lists of other users in order to learn of new polluters' $UserId$s and, possibly, to increase its own list. To do so, the lists of the k known $NodeId$s nearest to the local $NodeId$ (the closest overlay neighbors) are retrieved from the DHT and the local list is possible updated with new $UserId$s.

To make our experiment easier, and to get time-scale independent results, we organize the emulation into time *steps*. At each step, every peer on the network performs a variable number of DHT operations; we refer to N_{put} and N_{get} respectively as the random variables of the number of store and retrieve operations executed and we suppose that these variables are normally distributed. The lookup keys specified as PUT and GET parameters are selected from a set of 10^5 randomly generated 160-bit keys; accordingly to previous studies on the distribution of popularity of the resources on P2P file sharing networks [155], we suppose that the frequencies of such keys are

distributed accordingly to a Zipf's law; we choose an exponent equal to 1. When a step is over, the *RC* gossip service is activated and new blacklisted contacts are possibly learned.

The network is, of course, partitioned into two subsets: the *Good* and the *Bad* peers. The bad nodes are distinguished from others because they publish only fake objects and they do not take part in the *RS* activities. At each step, the *bad out degree* (D) of good nodes is measured. D is simply defined as follows:

$$D = |\{(x, y) | x \in Good \wedge y \in Bad\}|$$

A link between a good and a bad node (the (x, y) edge) is determined by the presence of the bad node's contact into the good node's routing table; the percentage of bad nodes is given by the p parameter.

We instantiate a 150 nodes network and then we run our emulation for different values of p. We set Kademlia routing parameters k and α to tiny values (respectively to 4 and 2) because the network size is relatively small. Our experiment is aimed to model a coordinated pollution attack, so we suppose that attacker nodes start polluting the DHT at step 0, and no new node is instantiated during the emulation period. Of course, during the whole overlay network life, several attacks like this can happen, however we focus only on a snapshot of a single attack which anyway easily allows to understand the effectiveness of Likir blacklist and RS.

The results are shown in Figure 11.4; the plotted values of D are normalized on the initial D value. In an initial phase, before fake resources are widely spread across the network, the number of bad contacts in the good peer's routing tables increases, because new contacts are learned due to the lookup procedures executed by the nodes of each partition. But the diagram shows that this trend is reversed after the first step; the value of D is reduced to about one fifth in just three or four synchronization steps, for every value of p, and then decreases asymptotically to zero, thus cutting off the cluster of bad nodes from the healthy part of the network.

The blacklisting method results effective also for very high values of p (e.g., $p = 0.8$) because, even there is a slight probability that a non-polluter node has the contact of another honest node in its closest overlay neighbors set, a great portion of resources on the DHT results corrupt, so many evil nodes are discovered at each step.

11.4.3 Folksonomic content search

A resource search engine is a central component in modern SNSs. Search tools are used by OSN participants to expand their knowledge to thematic contexts that reside beyond their local social cluster. For this reason, we believe that an effective instrument for content search cannot be left out of consideration in P2P designs of OSNs if they are intended to be competitive with their centralized counterparts.

In our framework, like in many other decentralized OSN designs, content encryption is one of the means to obtain content confidentiality. Obfuscating data enhances privacy but has also a negative side-effect on searching procedures. In fact, any content-based resource indexing is not applicable if the structure of shared objects is completely obscured by ciphers.

Certainly, encrypted objects can be found specifying their type and DHT key in classic DHT lookups. This approach is effective as long as the entity that initiates a search procedures is an automated application, that knows by protocol the set of keys and types associated to the objects it has to insert or retrieve from the overlay. But this system becomes too rigid when the search query is submitted directly by a human user. In fact, DHT lookups are limited to the well-known *exact-match problem* [141]: the knowledge of the *exact* lookup key is required in order to retrieve the content from the distributed storage.

The straightforward solution adopted in most of the DHT applications, like file sharing, is to compute the lookup key hashing the words of a representative content name. Nevertheless, the name assigned to a resource by its publisher may be insignificant to people that are interested in its retrieval and, in practice, this approach hides many objects from their potential consumers.

A more suitable alternative comes from the collaborative tagging paradigm brought into vogue by the Web 2.0 philosophy. In tagging systems, resources are marked with arbitrary labels assigned directly by the users; the result is folksonomic categorization of resources. Recurring to folksonomies is often necessary in very populated collaborative systems because the extremely high and continuously growing number of objects would make unfeasible for a team of experts to perform a classification. Furthermore, it has also been shown that the overall quality of the folksonomic indexing structure is comparable with the classic taxonomies, in terms of accuracy, completeness and consistency [148].

For this reasons, we define a folksonomy-based search engine for our social architecture. Given the distributed setting in which we are positioned, the bindings between tags and resources are directly stored by the users in the DHT. The idea is that a OSN user can mark a resource with a tag just publishing a reference pointer to that resource (i.e., its DHT key and its type) using the hash code of the assigned tag as lookup key. Subsequent searches for that tag will return the set of resources labeled with it. It is worth noting that if indexed resources are not public, a proper grant is anyway needed to get the content referenced by the pointer.

In addition to the basic indexing feature, folksonomies can be exploited to realize also a *navigational* paradigm of content search, where a user can drift from a selected category to another based on the semantic correlation between them. Examples of such paradigm can be found in classic query-based search engines like Google Wonder Wheel [130] as well as in tag-based search engines like Yahoo! Tag Explorer [350].

In folksonomic navigation, the most similar tags to the selected one are returned to the querier at each step. Anytime, the user can choose to continue the navigation or to retrieve the resources marked with the current (and with all the previously selected) tag. Implementing this possibility in our architecture implies to explicitly map the similarity relations between tags on the DHT. In particular, in order to consistently maintain the information on the tag-tag similarity graph over time and to efficiently manage the frequent addition of both tags and indexed resources, we rely on the $DHARMA$ approximated algorithm defined in Chapter 12. In DHARMA, tag-tag similarity is based on the notion of co-occurrence [216] (i.e., the similarity score between two tags is the number of resources that the two tags label in common) and the update of similarity arcs is managed with

an approximated strategy that limit the cost of the both tagging and navigation operations to a constant number of lookups.

As a result of these considerations, we can define a folksonomy-based search engine layered on Likir together with its high-level API:

$$\text{TAG}(label, obj) \tag{11.11}$$

$$\text{SEARCH}(tag) \tag{11.12}$$

The semantic of the operations is self explanatory. The SEARCH method returns both the set of tags similar to the specified label and, of course, the set of tagged resources.

11.4.4 On storage service

Assuring content availability is one of the most important challenges in the implementation of distributed SNSs. In our system, the DHT is intended to be the main storage layer for items generated by LotusNet widgets. Availability, lookup accuracy and efficient content dissemination are *intrinsic* properties of DHT storage platforms. Well-known incentives mechanisms for DHT peers to stay online [357, 21] can be applied to Likir as well. Moreover, note that Likir is a general-purpose DHT which can simultaneously support many other services beside LotusNet (e.g., classical file sharing). Therefore, LotusNet clients could profitably advantage also of the storage provided by DHT nodes that are not OSN participants.

Many DHT-based utilities that complement the basic storage services with more complex functionalities like durability or unlimited content size have been presented in literature [286]. We do not want to stuck on a specific storage scheme, because different applications could have very different needs regarding the storage. Instead, our aim is to define a flexible framework that offers a basic, privacy-aware storage facility and that can be used as fundamental building block to create more complex storage services without losing the property of privacy-awareness.

This is accomplished with the use of grants. If the items published on the DHT are used as pointers to external services, the same privacy mechanism that combines grant-based access control with content encryption can be used. In a nutshell, grants are self-contained entities that can be reused also in services that are external to the DHT. The possibility of reusing grants preserves the content confidentiality across different storage systems.

Of course, if the DHT is used as main mean of storage, non-stop availability of data comes at the price of relying on untrusted storage servers, i.e., the index nodes. Usually, this is not considered to be a good policy [222] unless basic security/privacy requirements like confidentiality, data integrity and authenticity are satisfied [123]. We have shown that the LotusNet architecture transparently satisfies such requirements by design and, additionally, DHT redundancy property is a good shelter against denials of service like the *smashing attack* [222]. Moreover, like pointed out in Section 11.3.2, thanks to the strong binding between overlay ID and user identity, LotusNet is flexible enough to allow widgets to store their data at trusted index nodes, like the known friends, at cost of potentially losing full-time availability.

Module	Operation	Description
Likir	PUT($key, obj, type, public, ttl$)	Publish an object on the DHT
	GET($key, type, userId, recent, grant, sizeonly$)	Retrieve an object from the DHT
	BLACKLIST($userId$)	Put $userId$ in a local blacklist
	REGISTER()	Register a handle for incoming notifications
DAC	CONTACT($userId$)	Search for $userId$ in LotusNet
	GRANT($userId, regExp$)	Issues a grant for $userId$ for types specified by $regExp$
	GETGRANT($userId$)	Gets the grant received from $userId$
	GETSELFGRANT($userId$)	Gets a self-signed grant
Notification	REGISTER($applicationName$)	Register an application-specific handle for incoming notifications
	NOTIFY($userId, applicationName, obj$)	Send a notification to a specific user and application
Tag Search	TAG($label, obj$)	Tag the specified object
	SEARCH(tag)	Tag-based search. Return a set of tags and a set of resources
Reputation	FEEDBACK($userId, score, proof$)	Rate a user's behavior

Table 11.2: Summary of the LotusNet services to applications

11.5 Crawling attack

Previously in Section 11.3.2 we showed that a malicious entity cannot crawl the DHT searching for private information and cannot even position probe nodes on the overlay to intercept sensitive data. However, a weakness of this scheme reside exactly in the delicate phase of creation of new contacts. Indeed, the attacker can leverage the incautious behavior of some users inducing them to accept the attacker's contact request and, consequently, to issue grants for it. In this Section we want to show to what extent malicious peers can extract user information from the network using this technique. Additionally, we propose further security expedients that could be adopted to improve the privacy preservation also in this scenario.

We suppose that the attacker is a common peer that runs a *crawling* application whose behavior is sketched by the pseudo-code in Algorithm 1. Basically, the crawler starts from a seed user (lines 1-2), asking for its friendship and offering in exchange a grant for the full set of its resources (lines 5-6). If the request is accepted (line 7), the set of widgets specified in the received grant is extracted (line 8). For each widget, all the accessible information is retrieved from the DHT and possibly added to a local database (lines 9-10). The crawler can also try to extract informations about target user's friends directly from the widgets' data (line 11). The whole procedure is then repeated for the new discovered contacts in a breadth-first fashion (lines 12,3,4).

Of course, extensive crawls must be executed by *automatic* procedures like the crawling application described. For this reason, a first counterattack to limit the effectiveness of crawls is recurring

Algorithm 5: Crawler widget

```
Input : seed, a user identifier
List contacts = new List ()
contacts.add (seed)
while contacts not empty do
    String userId = contacts.pop ()
    contact (userId)
    grant (userId,".*")
    if grant received from userId then
        Set widgets = getWidgets (userId)
        for w in widgets do
            retrieveInformation (w)
            List newContacts = retrieveContacts (w)
            contacts.add (newContacts)
```

to a contact establishment procedure that is resistant to robots. Requiring that both parties solve a puzzle which implies human interaction (e.g., captchas [336]) before they can connect to each other with a social tie is a very lightweight and inexpensive solution that can severely limit this kind of attacks.

Apart from this consideration, if the target user is somehow tricked to issue a grant for the attacker, a portion of its private information is inevitably disclosed. The extent to which this happens is proportional to the level of trust that the target user places in the new friend. Here, two cases may occur.

In the first case, the attacker introduces herself with a user identifier that recalls to the target node a person she knows already in real life (or in other online social contexts). The belief of being interacting with a highly trusted peer can lead the victim to disclose a high quantity of its personal information. This problem can be mitigated by adopting some basic *web-of-trust* features borrowed from the PGP setting. Users can produce special tokens for their social contacts in order to certify the binding between the user and its identity. When contacting a peer, a user exhibits its own identity certifications; since link formation process in social networks often involves *triadic closure* [282] (i.e., people becoming friends have often at least one friend in common), it is likely that the recipient of the friendship request directly knows some of the certifiers. The absence of known certifiers should warn the user about a potential privacy risk.

In the latter case, the attacker's identity is completely new to the target user. In this situation, the grants should be released gradually. To this end, a profitable collaboration between the Reputation Module and the DAC Module could be exploited, posing an upper bound to the permissions that can be issued for a social contact according to its the reputation level. Afterward, the growth of reputation in time determined by positive feedbacks received from other known peers or from the local user may cause the expansion of the grant.

If all these countermeasures fail, can the attacker seize also the identifiers of the victim's social

neighbors (line 11 in Algorithm 1)? Or, worse, can the attacker retrieve also data *belonging* to its neighbors? We previously noticed that the information about the user's personal contact network is not explicitly mapped anywhere, so the attacker may learn it only analyzing the resources published by the widgets it has been granted the access to. Moreover, widgets may often not require to store user identifiers explicitly. For example, in publish-subscribe applications, where a user publishes updates for a group of *followers*, there is no need to store the list of recipients together with the published data, simply because the access to the content is managed using grants only.

However, even if the attacker succeeds in learning a part of the social network topology, the most important thing is that the personal information of that contacts cannot be unveiled, since the grants gained by the attacker are useful to retrieve only the victim's protected content. In order to access the information of discovered friends, the attacker must reiterate the whole crawling procedure.

In this setting, considering all the countermeasures we presented, a crawl of the P2P network aimed to data collection would be very costly and limited to very inattentive users. Of course, the greatest defense ever against privacy leakages is a continual user awareness.

Chapter 12

DHARMA a DHT-based collaborative tagging system

12.1 Decentralizing collaborative search

Social applications are rapidly popularizing collaborative tools for indexing, retrieval, access and distribution of content over the Internet. Multimedia resources are made available through websites and P2P systems, together with annotations, metadata, tags, and other kind of information about the owner and/or the content itself. Such information is often used to fill the semantic gap between the personal user experience, and a more general description of a given resource. Nevertheless, such huge volume of information is often hidden to traditional search engines, since a common query infrastructure and language is missing.

During the years, the Web community has been supported with many retrieval techniques, that can be categorized in two main paradigms: *navigational search* and *direct search*. The first family of strategies assumes the existence of a taxonomy, usually predefined by a group of experts, that can be iteratively browsed by a user from general categories to more specific subclasses of information (e.g., Yahoo! Directory). Direct search let the user query the engine by means of a (set of) keyword(s) (e.g., Google). Even if the latter has gained a vast amount of success during the last years, very recently navigational paradigm has emerged again due to the diffusion of folksonomies within popular tagging systems as Flickr, del.icio.us, and so on. In fact, folksonomies have been showed to overperform monolithic hierarchical classifications in social domains where many users with different mental attitudes and vocabularies are active.

In the recent past, much effort has been devoted to direct search strategies, very common in unstructured P2P systems, and to *exact match key-based lookup* techniques, that are basically used by almost every structured overlay network. But quite surprisingly, benefits coming from the application of navigational search in the P2P domain have been quite underestimated. Moving from the pioneering work of Crespo and Garcia-Molina [92]), few research has been conducted on semantic routing for P2P systems and on merging collaborative tagging, folksonomies and P2P

systems. In the following, we give an overview on such research papers.

Asiki et al. [28] propose an efficient indexing scheme for storing and retrieving concept hierarchies over a fully decentralized system is given, even they do not take into account folksonomies.

A P2P infrastructure for tagging systems (*PINTS*) is proposed by Görlitz et al. [132]; in particular, authors design a scheme to maintain feature vectors for characterization of users and resource of a tagging environment on a DHT. Feature vectors may be useful for calculating the similarity between users or for constructing algorithms for ranked retrieval.

PINTS comes as a building block of Tagster [133], a distributed content sharing and tagging system where the user-resource-tag graph is stored in a DHT. A dedicated storage index is used for each tagging relation, so each edge in the graph is stored at different overlay responsibility areas. For this reason, one lookup for each edge retrieval is needed, and this could make the navigation expensive in systems with a huge number of tags and objects. Furthermore, navigational aspects between related tags is not explicitly taken into account.

Tag-based navigation is taken into account by Bouillet et al. [62], who present a centralized web service discovery system based on folksonomies. Tags, together with variables, are used to assign semantic information to input and output messages of the service operations. The key feature of this work is the possibility to exploit the subsumption relation between variable types to compose the discovery activity as an acyclic navigational workflow.

Navigation through tag cloud refinement, together with a recommendation service, are provided by GiveALink [320], a social bookmarking tool that allows tag-based web browsing. The system architecture, however, is fully centralized.

Our own work is rooted in a previous attempt of defining a fully decentralized search engine for web services discovery [225]. In this previous proposal, resources are organized in a tree-like tag structure which is mapped on a structured P2P system according to an iterative lookup technique that aims to solve the structured P2P exact match key-based routing problem. That approach revealed some drawbacks because the rigidity of the tree structure fits badly with the chaotic and ever increasing tag addition activity.

Starting from such previous experience we want to define a more general tagging system model that can be exploited to define navigational search strategies in fully distributed environments. Such model should fit the social media domain in the broadest sense of the word, since it could be used to implement a high level engine that allows the user to search in different environments (web, social networks, P2P file sharing networks, and so on).

LotusNet would be the natural platform to implement such a distributed search module. However, our aim is to provide a general framework that can fit any DHT system. In fact, the main challenge to reach this goal is intrinsic in the mapping of the logical structure of a folksonomy (seen as a network of tags) on a P2P layer (that partitions a given keyspace among the participating nodes) that can overcome the problems of loss of efficiency in tag insertion and retrieval. We propose a way to perform such a mapping, introducing an approximation strategy that fits well with dynamic and decentralized tagging. Our technique efficiently supports common insertion and retrieval operations also in real-world scenarios with very unbalanced distributions of the number

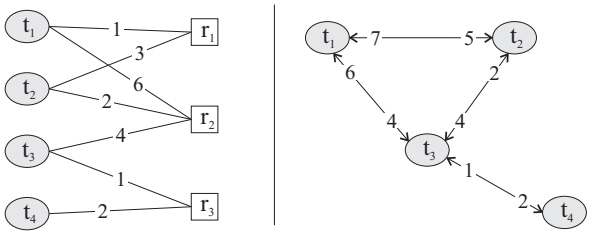


Figure 12.1: Bidirectional arcs in a Folksonomy Graph (right) aggregates asymmetrically weights in the Tag-Resource graph (left)

of tags over resources.

12.2 Tagging system model

The first step in the definition of our distributed search engine is a high level description of the tagging system; we define the tag-resource graph and also a folksonomy graph by means of a simple tag similarity measure (Section 12.2.1), and how these graphs are modified during users interaction (Section 12.2.2). Finally, we show how this model can be used for navigating through tags in search of resources (Section 12.2.3).

12.2.1 Graphs definition

From a graph perspective, collaborative tagging systems can be defined as tripartite hypergraphs [176, 224], in which three sets of actors are involved:

- U is the set of users of the system, that actually tag resources.
- T is the set of tags.
- R is the set of resources being tagged.

However, since in our work we focus mainly on tags and resources, we perform an aggregation across the user dimension in order to obtain a bipartite graph that links tags to resources. We define such a graph as the *Tag-Resource Graph (TRG)*, where $TRG = (T \cup R, E_{TR})$, s.t. $(t, r) \in E_{TR}$ iff at least a user tagged r with t. Moreover, for each arc in E_{TR} we define a weight $u(t, r)$ that is the number of times r has been tagged with t (see Figure 12.1 on the left). The reader can observe that we are adopting the so-called *distributional aggregation approach* [216] that yields to a graph in which the weight of an edge $(t, r), t \in T, r \in R$ is equal to the number of users tagging r with t.

We can use such graph to extract $Tags(r)$ and $Res(t)$ denoting, respectively, the subset of tags that label a resource r and the subset of resources that have been tagged with t:

$$Tags(r) = \{t \in T | \exists (t, r) \in E_{TR}\}, r \in R \tag{12.1}$$

$$Res(t) = \{r \in R | \exists (t, r) \in E_{TR}\}, t \in T \tag{12.2}$$

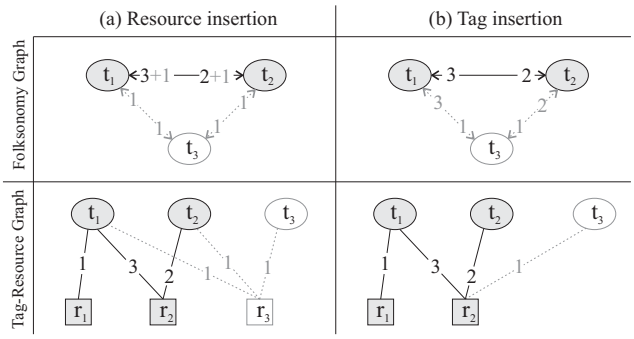


Figure 12.2: (a) Resource insertion: resource r_3, labeled with t_1,t_2,t_3 is inserted (b) Tag insertion: tag t_3 is attached to r_2. Light arcs and nodes are those added during the insertion

Since our purpose is to define a tag-based search engine, we introduce a simple *Folksonomy Graph (FG)* that can be trivially derived through collaborative tagging. Inter-tag relations should be detected by means of a distance measure between any pair of tags. We interpret such a distance as an asymmetric similarity function between two generic tags t_1 and t_2, s.t. $sim(t_1, t_2) = \sum_{r \in Res(t_1)} u(t_2, r)$.
Roughly, $sim(t_1, t_2)$ says how many times resources labeled with t_1 have been tagged also with t_2. Even if many different similarity measures could be adopted in folksonomies [216], such aggregation of tag-resource weights is a metric that is easy to calculate and, as we explain in Section 12.3, handy to be mapped on a fully decentralized and dynamic context. This metric can be considered as a generalization of tag-tag co-occurrence [74].

Now, we can define our folksonomy graph as $FG = (T, E_F)$, s.t. $(t_1, t_2) \in E_F$ iff $sim(t_1, t_2) \geq 1$. Let us observe that, by construction, if $sim(t_1, t_2) \neq 0$, then $sim(t_2, t_1) \neq 0$, even if it may happen that $sim(t_1, t_2) \neq sim(t_2, t_1)$. Hence, we represent connections between tags using bidirectional arcs with two weights. Finally, we will need to deal with the tags related to a given tag t. Such set is the neighborhood of t in FG, denoted with $N_{FG}(t)$.

For example, in Figure 12.1 (right), the arc (t_1, t_2) of FG has weight 5 because resources r_1 and r_2, have been tagged also with t_2 by, respectively, 3 and 2 users; let us observe that, conversely, $sim(t_2, t_1) = 7$.

12.2.2 Graphs maintenance

The active collaborative behavior of the user community leads to a continuous evolution of the TRG and the FG, due to the addition of new items and new annotations.

Resource insertion

When an user inserts a new item r and tags it with $T_r = \{t_1, ..., t_m\}$, then a new resource vertex is inserted in the TRG. Of course, also for each new tag t_i, a new vertex is inserted in the TRG, so that R is updated to $R \cup \{r\}$, and the set of tags to $T \cup T_r$. Moreover, for each $t_i \in T_r$, an edge (r, t_i) is added to E_{TR}, with $u(r, t_i) = 1$. As a consequence, FG must be changed, too: for

each pair of tags $t_i, t_j \in T_r$, a new arc (t_i, t_j), if not previously existent, is added to E_F with $sim(t_i, t_j) = sim(t_j, t_i) = 1$. Otherwise, $sim(t_i, t_j)$ and $sim(t_j, t_i)$ are simply incremented of one unit.

Tag insertion

Graphs grow also when an existent resource r is tagged with t. Firs, if $t \notin T$, a proper tag node is added. Therefore, if $t \notin Tags(r)$, then a new edge (t, r), with $u(t, r) = 1$ is added to E_{TR}; conversely, if $t \in Tags(r)$, then $u(t, r)$ is simply incremented. Similarities are changed consequently. For each tag $\tau \in Tags(r)$, $sim(\tau, t)$ is incremented by one. Instead, $sim(t, \tau)$ is changed depending on whether t was in $Tags(r)$ before the tagging operation or not. If t was in $Tags(r)$, then $sim(t, \tau)$ is left unchanged, otherwise $sim(t, \tau)$ is incremented by $u(\tau, r)$. Arcs in the FG are created or updated accordingly.

Examples of both operations are showed in Figure 12.2.

12.2.3 Faceted Search within the Folksonomy Graph

Our purpose is to exploit our model in order to let the user explore the given multi-dimensional information space by iteratively narrowing the number of choices at each search step.

Many popular tagging systems (e.g., Flickr, Last.fm, and so on) make use of resource *clustering* according to some measure of similarity. For example, considering our Folksonomy Graph, we can easily identify clusters representing repeated patterns of tags that can be presented to the users through lists or tag clouds. Such clusters can be intuitively used to refine the query or to disambiguate search keywords. Nevertheless, clustering techniques can generate unpredictable groups, produce cycles in the navigation process, and limit the browsing features of the system since they may not allow refinements.

Generally speaking, users prefer hierarchical classifications with clear and meaningful labels at each level of the tree. For example, a tag that is presented more than once during the same search process can generate confusion, as well as a general term (e.g., "rock") that is found in a cluster after that a specific tag (e.g., "heavy-metal") has been selected. Unfortunately, a traditional and rigorous taxonomy is difficult to be provided in a highly dynamic social domain with many users with different mental attitudes and vocabularies.

Faceted search can be seen as a middle ground approach that allows the user to "dive" the folksonomy without semantic cycles and to iteratively refine the tag-resources space (e.g., TagExplorer by Yahoo! Research). Accordingly to this approach, the user browses the tagging system through a path in FG. We can interpret every tag of such a path as a different level of a hierarchical faceted search process; in fact, selecting subsequent tags in the hierarchy results in a conjunction over the selected annotations, and each step zooms in the tag-resource space, narrowing the focus of the search.

Two important consequences of this approach are *query convergence* and *vocabulary specialization*. Let us assume that the user starts the search process selecting tag t_0, and afterwards she chooses $t_1, t_2, \ldots, t_n$. At each step, only co-related tags are presented to the user. Tag t_i is always

a neighbor of t_{i-1} in FG, that is $t_i \in N_{FG}(t_{i-1})$. Moreover, at step i a set of tags T_i and a set of resources R_i can be presented to the user:

$$T_i = \begin{cases} N_{FG}(t_0) & i = 0 \\ T_{i-1} \cap N_{FG}(t_i) & i > 0 \end{cases}; R_i = \begin{cases} Res(t_0) & i = 0 \\ R_{i-1} \cap Res(t_i) & i > 0 \end{cases}$$

Even if we are not concerning on presentation aspects, we can assume that to improve usability only a subset of T_i is displayed (using a tag cloud or an alternative representation), and that a subsequent tag selection would be equivalent as a zoom in, eventually visualizing other (more specific) tags. Obviously, since previously chosen tags are not taken into account in subsequent steps, $\forall i : |T_i| < |T_{i-1}|$. The upper bound of the iterative process is $O(|T_0|)$, and so convergence is trivially proved. It can be noted that browsing can be delayed by tags that are "semantically equivalent" (i.e., all $\tau \in T_{i-1}$ s.t. $|T_i| = |T_{i-1}|$). However, such situations are limited in numbers and do not affect significantly search performance.

12.3 Mapping on a DHT

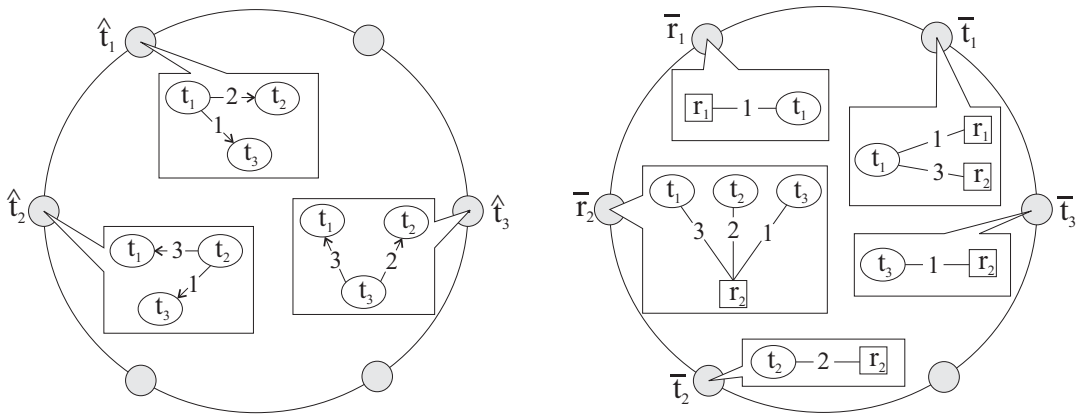
Next, we present a general insight of how the model defined in Section 12.2 can be mapped on a Distributed Hash Table. We show that a naive implementation of our model would lead to grievous inefficiencies which severely limit the scalability of the system; therefore, we propose an approximated approach to overcome these issues.

12.3.1 Distributed model

In order to map the folksonomy on a DHT we need to shrink the TRG and the FG both in small structural *blocks* that can be stored at different overlay nodes. In particular, each block contains a node together with its outgoing edges. Accordingly, every resource node $r \in TRG$, together with its outgoing edges to nodes $t \in Tags(r)$ is contained into a single block. Symmetrically, every tag $t \in TRG$ with its outgoing edges to $r \in Res(t)$ forms a block. Likewise, the FG is partitioned in blocks containing a tag t with the arcs that links it to its neighbors in $N_{FG}(t)$. More formally, we define four types of blocks:

1. $\bar{r} : \{(t, u(t, r)) | t \in Tag(r)\}, r \in R$
2. $\bar{t} : \{(r, u(t, r)) | r \in Res(t)\}, t \in T$
3. $\hat{t} : \{(t', sim(t, t')) | t' \in N_{FG}(t)\}, t \in T$
4. $\tilde{r} : (r, URI(r)), r \in R$

The TRG is split into blocks of type 1 and 2, the FG into blocks of type 3. Type 4 blocks are introduced only to conceptually associate the resource itself (a URI of a generic object or service) to its name r (a human readable identifier which denotes the resource). Each block is mapped on a lookup key computed from the name of its node concatenated with a string which determines the block type (e.g. the hash of $t|$"2" is the key of type 2 block for tag t). For brevity, we denote



(a) Folksonomy graph mapping on the DHT

(b) Tag Resource graph mapping on the DHT

Figure 12.3: Folksonomy mapping on a DHT. Overlay nodes are labeled with the key they are responsible for. The content of node storages are depicted into the baloons.

with $\bar{r}$, $\bar{t}$, $\hat{t}$, $\tilde{r}$ the lookup keys for blocks of type 1-4; for simplicity, we use this notation to directly denote the blocks without introducing any ambiguity.

Figure 12.3 shows how the FG and the TRG depicted in Figure 12.2.b are partitioned in blocks and mapped on a generic DHT layer. Type 4 blocks are omitted for simplicity.

Given such mapping, navigation, tagging and resource insertion through the P2P network are easy to describe. At each navigation step, when a tag t is selected, tags and resources related to t are retrieved by fetching blocks $\hat{t}$ and $\bar{t}$; intersection with tag and resources set retrieved in following steps are performed locally. Insertion of a resource r, marked with tags $t_i, i \in [1..m]$, requires the creation of block $\tilde{r}$ to store the URI and of block $\bar{r}$ to connect the resource with its tags. Reverse tag-resource connections are mapped by inserting blocks $\bar{t}_i$ for each tag t_i given in input. Each t_i should then be connected to others in the FG by creating (or updating) its block $\hat{t}_i$. Finally, when a resource r is tagged with a label t, the weight of edge (t, r) is incremented by updating blocks $\bar{r}$ and $\bar{t}$. Then, tags $\tau \in Tags(r)$ are retrieved from block $\bar{r}$. For every τ, the weights of arc (t, τ) is incremented by updating block $\hat{t}$, while reverse connections (τ, t) must be updated by modifying blocks $\hat{\tau}, \forall \tau \in Tags(r)$.

We suppose that retrieving or modifying the content of a block on the DHT costs only one overlay lookup operation. This assumption is reasonable if the overlay is equipped with proper PUT and GET operations, which, respectively, insert and retrieve content from the DHT by exploiting the overlay network's lookup service.

In the particular case of Likir/LotusNet we can implement such scheme as follows. The addition of an arc between two nodes x and y can be performed specifying the hash of x (concatenated with the node type) as key and y (concatenated with some application-specific marker, the string "DHARMA" for instance) as type in the PUT primitive, and using a *one-bit token* ($\bullet$) as content. Of course, the content is stored as *public* because any user should be able to use the search engine.

$$\text{PUT}(nodeType|x, \bullet, DHARMA|y, True, ttl). \tag{12.3}$$

The publication of such content logically determines a unit increment of the weight of the (x, y) arc in the TRG (or FG). When an application needs to retrieve the block related to the node x, it calls the GET primitive as follows:

$$\text{GET}(nodeType|x, DHARMA*, -, -, -, True), \tag{12.4}$$

where $DHARMA*$ instructs the index node to return all the resources which type starts with the string "DHARMA", and the *sizeonly* parameters set to $True$ allows the client to retrieve only the number of resources submitted for each content type. As a result, this call returns a list of pairs $(type, count)$ representing the weighted edges departing from node x, which compose the full information of the block.

12.3.2 Approximated approach

Implementing the algorithms defined in Section 12.2.2 with our distributed framework produces two severe issues, both concerning the tagging operation (i.e. the FG update).

The first is a complexity problem. We stated that when a new tag t is added to a resource r the weights of the arcs $(\tau, t), \tau \in Tags(r)$ must be updated. In the DHT domain this implies the update of blocks $\hat{\tau}$ of each $\tau \in Tags(r)$. Accordingly, a number of lookups which is linear with $|Tags(r)|$ is performed. This cost is unsustainable because, as we show later in Section 12.4, a resource can be tagged with several hundred labels. Even if different lookups can be executed in parallel, the bandwidth usage would be definitely excessive for such simple and frequent operations.

The second is a consistency problem, caused by a *race condition*. To keep the graph consistent with our model, if the arc $(t, \tau), \tau \in Tags(r)$ was not present before new tag t insertion, then $sim(t, \tau)$ should be incremented by $u(\tau, r)$. Nevertheless, it is hard to implement correctly this practice in a fully decentralized system. It is easy to understand, indeed, that if two users try to add simultaneously the same tag t on the same resource r, there is the risk that the value of $sim(t, \tau)$, for any $\tau \in Tags(r)$, is uncorrectly incremented twice, for a total value of $2 \cdot u(\tau, r)$.

These considerations must be taken into account to improve the algorithm design. We adopted two approximated strategies to solve these problems; call t the new tag and r the resource to be labeled.

Approximation A. Instead of incrementing the weights of all the arcs $(\tau, t), \tau \in Tags(r)$, perform the increment only for a *random subset* of $Tags(r)$. The cardinality of such subset can be chosen to be at most a *constant* number k; this expedient reduces the number of lookups needed for a tagging operation, preventing its complexity to scale with $|Tags(r)|$. We refer to k as the *connection parameter* of the approximated graph □

Approximation B. If the arc $(t, \tau), \tau \in Tags(r)$ was not present before the tagging operation, then increment the weight of (t, τ) only by *one* (and not by $u(\tau, r)$). This avoids the possible inconsistencies due to simultaneous addition of a new tag t to resource r □

Approximations make the similarity graph evolve differently from the abstract model described in Section 12.2. Thus, the distance from the theoretic and the mapped graph should be measured to check how much the search procedure is affected by our approximations. In Section 12.4 we present

Primitives	Insert $(r, t_{1..m})$	Tag (r,t)	Search step
#lookups (naive)	$2 + 2m$	$4 + \|Tags(r)\|$	2
#lookups (approx.)	$2 + 2m$	$4 + k$	2

Table 12.1: Distributed tagging system primitives cost

experimental results to measure such distance. It is worth noting that only the FG is affected by the approximation, while the TRG graph remains the same. Complexity of approximated operations in terms of overlay lookups is shown in the second row of Table 12.1.

The source code of the distributed tagging application that implements the approximated approach is available online (`likir.di.unito.it\applications`), with the name of *DHARMA* (*DH*T-based *A*pproach for *R*esource *M*apping through *A*pproximation). An implementation of the underlying DHT is available as well.

12.4 Evaluation

We give an evaluation on how the approximations introduced in our system design impact on the validity of the model using analytic and simulative approaches. The analysis is based on a dataset extracted from Last.fm. First (Section 12.4.1), we give a brief description of the main features of the dataset, then (Section 12.4.2) we analyze how the FG created through the protocol we defined in Section 12.3.2 well approximates the theoretic similarity model of the dataset and we show that user search experience does not decay due to introduced approximations. Additionally, in Section 12.4.3 we report the results of a simulative experiment aimed at the estimation of the mean number of steps needed for query convergence.

We base our experimental analysis on one of the snapshot of out the Last.fm dataset (see Section 3.2). We explored a population of 99,405 active users, extracting nearly 11 millions of annotations in the form of triples $\langle user, item, tag \rangle$ where an item can be an artist, an album or a specific song. From this raw dataset, we built the bipartite TRG, which has 1,413,657 resource nodes and 285,182 different tags, from which we derived the FG.

12.4.1 Last.fm dataset overview

We analyzed some structural properties of TRG and FG both. Some of the most relevant things to know are nodal degree distributions: in particular we extracted the distribution of the cardinalities of $Tags(r)$, $Res(t)$ and $N_{FG}(t)$ sets. Statistics of degrees (mean, standard deviation and max values, all rounded to integer) are shown in Table 12.2 and cumulative degree distributions are depicted in Figure 12.4.

A strong core-periphery structure emerges in the TRG. In particular, a huge portion of tags (about 55%) marks only 1 resource and almost the 40% of resources are labeled with just 1 tag. Conversely, the dataset has a core of much more connected tags and resources; these correspond to the semantic top-level (or at least high-level) tags (e.g. “rock”, “pop”, “seen live”) and to the

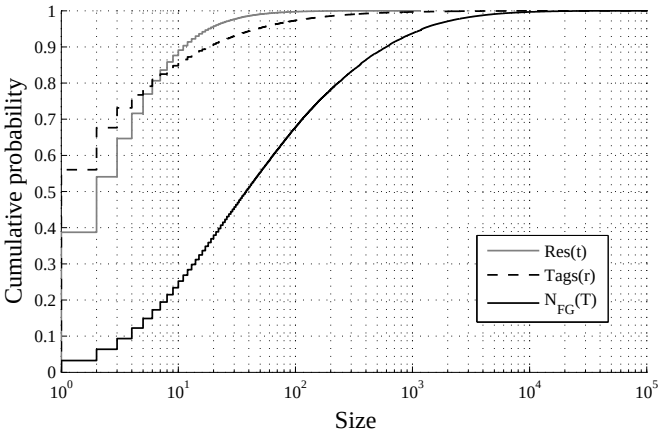


Figure 12.4: Last.fm nodal degree CDF

Degree	***Tags(r)***	***Res(t)***	$N_{FG}(t)$
μ	5	26	316
σ	13	525	1569
max	1182	109717	120568

Table 12.2: Last.fm graph degree statistics

most popular resources.

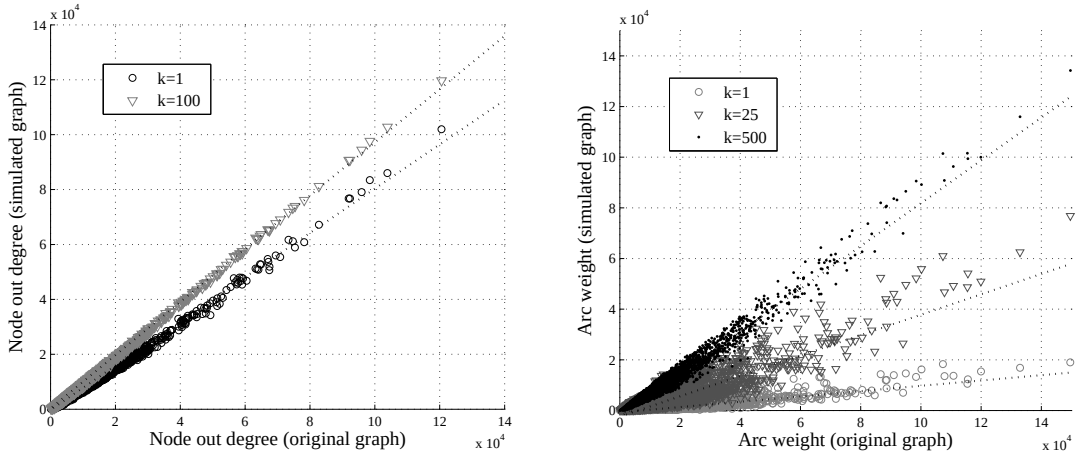
A similar scenario comes with the FG: the 80% of tags has a not-null similarity with at most one or two hundred nodes, while the nodes belonging to the core of most popular tags are connected with several thousand nodes.

Given this setting, it is clear that the updates and lookup operations performed within the core structures of the network are the most "problematic" in terms of DHT operations. First, the number of tags marking a resource can be too high to avoid Approximation A. Second, given a very popular tag, the number of related tags and resources can be definitely huge; since, usually, overlay messages are sent on UDP packets, the limited payload force to send only a subset of tags and resources available during a search step. Therefore it is important that only the most relevant objects are returned. The index side filtering provided by Likir is a good option to meet this requirement.

12.4.2 Approximated graph simulation

Given the Last.fm TRG and FG we simulate the evolution of such graphs with our approximated protocol in order to draw a comparison between the real dataset and the approximated one.

The simulation starts with a fully disconnected graph that includes all tags and resources from the Last.fm dataset. At each step, a resource r and a tag t are selected and a tagging operation is performed. The FG is updated according to Approximations A and B. Resource r is chosen with a probability proportional to its popularity in the dataset (i.e., $|Tags(r)|$ in the real TRG); tag t is selected between all tags in $Tags(r)$ on a local popularity basis (i.e., with probability proportional



(a) Nodal degree

(b) Arcs weights

Figure 12.5: Comparison between original and simulated FG.

to $u(t, r)$). Simulation ends when resources are labeled with all their related tags instances that appear in the real dataset. We executed the simulation for different values of connection parameter k.

We compare the original and the simulated FGs. We consider nodes out-degree and arcs weights in original graph against corresponding values in simulated graph: the result are depicted in Figure 12.5.

We notice that, even with $k = 1$, the points on the degree plot are aligned on a line whose slope is close to the diagonal; so we deduce that the variation of k does not significantly affects the nodal degree. On the contrary, arcs weight is significantly reduced for low values of k; to reduce the spread with the original values under a reasonable threshold, k must be set to values that would make an efficient implementation on a DHT system unfeasible. Nevertheless, we are not interested in minimizing the residues between theoretic and actual arcs weight. Instead, our aim is that some kind of *proportions* are kept. First, we want that the arcs weight ordering is maintained because the ranking of the $sim(t_1, t_2)$ weights directly influences the tags' ranking that is displayed during the search process. Second, we want that the proportion between the weight of every pair of arcs is not lost; if weight ordering is preserved for a pair of arcs but the ratio between their values significantly changes in the simulated network, then there is the risk of a *flattening* effect on the tag similarity values, thus reducing the information provided to the user in the search step.

To give a quantitative measure of such advisable features, we compared, for each tag t in the dataset, the set of its outgoing arcs $(t, t_i), t_i \in N_{FG}(t)$ with the same set taken from the approximated graph. The metrics we used for the arcs weights comparison are the Kendall's tau rank correlation coefficient (K_τ) and the cosine similarity (θ). K_τ evaluates the similarity between two ranks of a same set of objects on the basis of the number of inversions that have to be made to turn one ranking into the other; it ranges from -1 (when two rankings are the opposite) to 1 (for equal rankings). θ, which has the same range of K_τ, takes in input two vectors of the same length and is equal to 1 if these vectors are perfectly scaled (e.g. $\theta([1, 2, 3], [100, 200, 300]) = 1$).

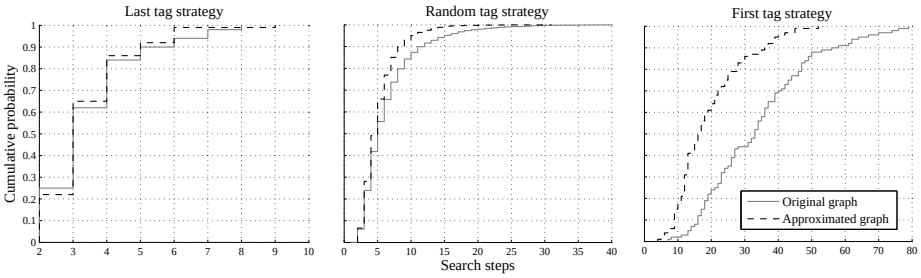


Figure 12.6: Effect of approximation on tag navigation path length

Furthermore, to give an estimation of how much information is lost with the approximation, we calculated the *recall* value, that is the ratio between the number of arcs in the approximated graph and in the theoretical graph. Finally, we calculated the portion of arcs, among the set of those arcs that are not represented in the approximated graph, whose weight is 1 in the theoretical model (we refer to this measure as $sim^1_{\%}$).

The mean values and the standard deviations of the cited measures, for some low value of k, are reported in Table 12.3. The main results obtained are the following:

- K_τ and θ values, measured on the set of tags which are common to the two models, are very high, independently on the value of k. This means that retrieved tags in the approximated model are well ordered and proportioned compared to the theoretical model.

- The value of *Recall* reveals that, for very small values of k, up to the 40% of arcs are not represented in the approximated model. Recall grows sub-linearly with k.

- The extremely high values of $sim^1_{\%}$ reveals that the weight of almost all these missing arcs is 1, which is the minimum value in the similarity network. Further analyses showed that, for every k, the 99% of the missing arcs has a weight ≤ 3. So, the missing arcs are positioned in the very tail of the weight ranking.

In a nutshell, even if correct proportions are kept, the number of arcs in the approximated FG can be considerably smaller respect to the original graph. Nevertheless, the arcs that are not mapped represent very weak similarities. In fact, the great majority of these arcs are simply *noise* caused by the insertion of meaningless or singleton tags, which cover a high percentage of the

k		**Recall**	K_τ	θ	$sim^1_{\%}$
1	μ	0.6103	0.7636	0.8152	0.9214
	σ	0.2798	0.2728	0.1978	0.1044
5	μ	0.7268	0.7638	0.8664	0.9346
	σ	0.2730	0.2380	0.1636	0.0914
10	μ	0.7841	0.7985	0.8971	0.9432
	σ	0.2686	0.2138	0.1424	0.0850

Table 12.3: Comparison between approximated and theoretic Folksonomy Graph

overall tags but that are useless during the search phase. Therefore, the approximation adopted does not affect the quality of the FG, but rather reduces the noise on the mapped graphs and eases the load on the P2P layer.

12.4.3 Faceted search convergence

Any kind of similarity relation between objects in a large corpus are difficult to navigate, because of the high density of navigable similarity links. Folksonomies, represented as graph-based models built from tags and resources and from the relations between them, are examples of such complex systems, very difficult to navigate. Convergence in this context could be solved by extracting a taxonomy from the folksonomic space [147] and then navigate from the root of the hierarchy down to the leaves. However, this approach may be very complex and can lead to several different hierarchies that may be equally good (or bad) for a user. Therefore, search convergence should be granted for any path the user chooses to navigate through related tags.

Search convergence is important because the quickest the navigation converges, the lowest the number of overlay lookups needed to locate a resource is. Convergence rapidity depends by which is the first tag selected. If it resides in the periphery of the FG, the search procedures will converge almost immediately, because the number of tags and the number of resources connected with it will be very probably quite small. Making a parallel with a taxonomic search structure, it is like the user had started her search from a node which is very close to a leaf of the tree structure, and so she had few levels left to explore.

On the contrary, the dual (and probably more frequent) behavior starts the search from more popular tags, those that resides into the core. In order to show that convergence is quick also in this case we report further simulative results. We took the 100 most popular tags and, starting from these, we simulated tag search procedures in order to estimate the average length of a search.

Three types of search were performed; independently from the search strategy, we suppose that the size of the tag set shown to the user at each step, T_i, is upper bounded to the top 100 tags retrieved from the DHT; larger sets of tags would be unsuitable for an effective user visualization. In the first search type (*first tag strategy*) the tag selected at each step is the most similar with the current tag. Formally, given a search path $t_0, ..., t_i$, the next tag selected is a label t_{i+1} such that $sim(t_i, t_{i+1}) \geq sim(t_i, \tau), \forall \tau \in T_i$. The second type (*last tag strategy*) is the dual of the previous: the selected label is always the tag which is the least related with the current one among the 100

Number of steps		**Last**	**Rand**	**First**
Original	μ	3.47	6.412	33.94
	σ	1.4175	4.4587	15.9942
	$\mu_{1/2}$	3	5	33
Simulated ($k = 1$)	μ	3.38	5.2140	19.17
	σ	1.2373	2.6994	10.3065
	$\mu_{1/2}$	3	5	16

Table 12.4: Search simulation statistics

tags displayed (i.e. tag t_{i+1} such that $sim(t_i, t_{i+1}) \leq sim(t_i, \tau), \forall \tau \in T_i$). In the third search type (*random tag strategy*), the next tag is selected uniformly random within T_i.

For each tag among the 100 most popular we simulated the *"first"* and *"last"* search and 100 *random* searches, on both original and approximated Folksonomy Graph (for $k = 1$), using the faceted search algorithm described in Section 12.2.3. The search procedure is stopped when $|T_i|$ reduces to 1 or when $|R_i| \leq 10$. We choose 10 as lower threshold for the number of displayed resourced because a set of 10 objects is small enough to be displayed to the user without the need of further filtering.

Statistics on search paths length are shown in Table 12.4. From the experiments, it emerges that the path length is characterized by a high variance, for every search strategy, due to the high variability in the nodal degree of the FG.

With regard to searches performed in the original model, we observe that in the "*last*" and "*random*" strategy, the mean (and median) values are very small if compared to the size of the dataset; in particular, note that these values are $< ln(|T|)$. The "*first tag*" strategy produces longer paths; however, a deeper result inspection revealed that they are originated by tag selection sequences which are very unlikely to be produced by a real user.

Roughly, the great majority of such sequences are those in which almost all tag selected are the most popular tags; since such tags are connected with huge sets of tags and resources, the size of the resource and tag sets decreases slowly at each search step. This is an expected behavior of the system, because if the user does not specialize the search terms it is clear that the navigation is maintained at a very coarse-grain level. Other slow-converging sequences are those in which many synonyms appear (e.g. "electronica", "electronic", "electro"). Here, since semantically equivalent tags mark more or less the same set of resources, it is clear that navigation from one to another does not add any filtering information to the search procedure.

Such categories of search path occur because the meaning of the tag is not taken into account in simulations. But when the tag navigation is executed by a human user, and a semantic thread is followed in tag selection, the path leading to the objective could probably result shorter, more similar to the "random tag" selection case.

Comparing the simulative results obtained in the original Folksonomy Graph with those obtained for the approximated graph, the advantage on query convergence determined by approximation is clearly shown. Figure 12.6, which plots the cumulative density function of search path lengths for both models in the three strategies, together with statistics of Table 12.4, shows that the approximated approach shortens the navigation, thus quickening convergence. This effect, particularly evident in the "*first tag*" strategy, is determined by the deletion of lightweight arcs from the graph. By wiping out the noisy connections the semantic distance between tags is increased, thus leading to a faster vocabulary specialization during the tag selection process.

As final consideration, remember that the simulated search ends when the set of resources reduces to an arbitrary threshold set to 10, but if this value is raised, even slightly, path lengths could be considerably reduced.

Summary on Part II

Services provided by **social media** have acquired an increasing potential in last few years, reaching a surprising effectiveness in recommending the users and anticipating their actions. This has been possible because of the great amount of information that users reveal to the systems in an explicit or implicit way. Inevitably, this scenario raises several **privacy issues**, moreover when sensitive user data leak between different systems or to other untrusted participants.

An increasing awareness about these issues is strengthening among social media users. In order to satisfy this demand, we follow the path traced by a recently-formed research community that proposed to migrate online social networks from the classic centralized paradigm to **peer-to-peer** layers in order to escape the opaque management of sensitive information performed by service providers. However, providing a complex social network service on a fully distributed layer is very challenging under three different, yet strictly related perspectives: **security**, **privacy** and **services**.

First, vulnerability to **attacks** suffered by **overlay networks** is a strong obstacle to the development of critical **applications** on DHTs. We make an overview of all the most known attacks to structured overlay networks and we show by the means of simulation the disruptive potential of such attacks under a very general attacker model.

To tackle this security problems, we propose **Likir**, an **identity-aware** version of **Kademlia** that offers an effective defense against a wide range of attacks, with a limited overhead. Even if a registration service is introduced, our architecture does not present a single point of failure, because the **Certification Service** (CS) is contacted only during the user subscription phase, through a simple web service. If the presence of a centralized authority must be avoided, the CS could be easily replaced by efficient distributed PKI infrastructures like the one proposed by Lesueur et al. [192]. Likir relies on the CS to avoid the proliferation of "zombie" nodes controlled by a single attacker, uses a new protocol enhanced with cryptography to provide secure message exchange between peers and introduces certificates attached to resources to contrast the spreading of polluted content on the distributed storage.

Then, we show that embedding identity at overlay level can be exploited also beyond security purposes. We presented **LotusNet**, a new architectural scheme of a DHT-based OSN that extends the very essential Likir API to provide a customizable privacy level of the user data.

We provide **confidentiality** by assigning the **access control** responsibility to overlay index-nodes. A content stored in the DHT is returned to the querier only if a proper **grant**, signed by

the its owner, is shown to the item keeper. Flexibility of grants and the possibility to store data at **trusted index nodes** allow the users to tune the desired privacy level for the activity around any particular widget or resource. Furthermore, grants exchanged between peers implicitly model the social network's topology, facilitating the establishment of new social contacts. Together with confidentiality, our approach assures a protection against other kinds of unwanted information leakage.

Besides access control, we define a set of **core services** useful for a wide range of social applications. We propose an asynchronous **notification** service entirely supported by the overlay layer, a **tag-based search engine** service to overwhelm the exact-match lookup problem of structured P2P network, and a handle for a **reputation system** intended to expel misbehaving users in a cross-application fashion.

We focus in particular on the description of **DHARMA**, the distributed tagging service. We present an approximated approach for the maintenance of the relational information that defines a folksonomy on a DHT. Simulative and analytic studies show that the approximated representation of the similarity graph does not upset the features of the theoretic folksonomy model. Besides, approximation can largely mitigates overfitting phenomena and significantly reduce the number of overlay operations for new tag insertion without degrading the user search experience. The information which gets lost in the approximated mapping is prevalently noise. The low number of lookups needed during the insertion/search phases allows an efficient implementation of a tag-based, general-purpose indexing service over a structured P2P network.

The specification of a complex service like DHARMA on the top of Likir and LotusNet provides the empirical demonstration that the P2P way is viable for the implementation of efficient and privacy aware services.

LoutusNet is the product of design choices that are agnostic with respect to the layers above. We do not make any assumption on the structure of applications or messages that compose the social media. As a result, we presented a social services framework that can host several, possibly interconnected social networking services thus creating an environment without closed **information silos** but that is at the same time respectful for personal information privacy.

Likir, DHARMA and some of the LotusNet core services we defined, have been implemented in the form of a **Java libraries** (`likir.di.unito.it`).

Part III

Opportunities and risks in the new Social Web

Chapter 13

Robots and earthquakes

"More human than human" is our motto.

Dr. Eldon Tyrell in *Blade Runner* (1982)

13.1 The next-generation social Web

The transition from the early digital age of the Web to the collaborative paradigm of the Web 2.0 that, in turn, gave origin to the Social Web, naturally raises the question about what will be the shape of the Web in the next few years. In previous Chapters we learned that predicting the future is no easy task but, if we look carefully, it is not so hard to detect even today some signs that let us understand what the traits of the next generation World Wide Web could be.

Online dynamics and the life in the real world have been entangled since the early stages of the diffusion of commercial Web browsers. Just to mention a couple of examples, online services radically changed the way in which we gather news and knowledge and the rise of online social networks expanded our instruments to communicate and get in touch with our social circles. Today, however, pervasiveness of the Web in real life is knowing an unprecedented expansion, up to the point that monitoring, understanding, predicting and even *manipulating* human phenomena is something that could be done through the digital world.

The first striking evidence of this radical change of perspective is given by politics. The importance that Twitter has acquired in the election debates and opinion formation campaigns of democratic elections is an example of the shift of communication power from conventional media to the Web. Tweets with political connotation are primarily useful for Twitter consumers to the end of their opinion formation, but much greater interest has been attracted in the research community by the opportunity of unveiling on a public and widely accessible platform all the opinions that "ordinary people" have traditionally exchanged in confined and private physical spaces. The possibility of monitoring the traffic of political messages of millions of users allows us to understand the dynamics of opinion formation and the evolution of consensus on particular candidates or parties [88]. A thorough analysis of political *memes*[1] enables also the prediction of political

[1] In the Websphere, a meme "refers to a catchphrase or concept that spreads rapidly from person to person via

alignment of single users [89] and, on a larger scale, several attempts has been done to predict the outcome of the elections in US and Europe from social media trends (even though controversial results have been obtained [327, 163, 258]).

If the digital world can reflect with great detail the trends and interactions occurring in the society, on the other hand a targeted use of the online social instruments can effectively influence and pilot some social dynamics of the real world. A proper exploitation of the interaction patterns of a social network can lead to effective attacks to the system. The phenomenon of Web *astroturf*[2] on Twitter [276] is an example of how the functions of a social media can be subverted to spread misinformation. Malicious users can indeed automatically generate traffic that can be disguised as spontaneous to spread fake news or misleading information about some particular topic. Even though much effort has been spent in contrasting these attacks [275] (see for instance the *Truthy* project at `truthy.indiana.edu`), still much has to be learned about the potential misuses of social media and the possible countermeasures.

Nevertheless, the penetration of the social Web into the dynamics of the real world has also brighter sides. For instance, social media had a pivotal role in directing and coordinating the protesters in the 2010-2011 demonstrations of the so-called *Arab Spring* wave [259, 337, 146], a vast social and cultural movement of Middle-East and North Africa uprising against dictatorships, human rights violations, and governments corruption. Moreover, the extreme pervasiveness of social media and their rapidity in disseminating information outperforms any other automatic systems when it comes to monitor a real-time event and to spread related notifications. The case of the recent earthquake in Japan, whose seismic waive front was preceded by the tweets reporting the earthquake itself [291] is a clear example on how our perception of reality has been changed by the use of social media. In this scenario, it is not surprising that also the modern approach to journalism is being reversed by the social media revolution. In fact, online social media are becoming one of the primary information pool that journalists can use to produce news of public interests. As a result, *computational journalism* [85] is emerging as a new technique of exploration of the Web with advanced analysis and visualization tools to extract relevant news in real-time (see for instance the *cascade* project by New York Times (`nytlabs.com/projects/cascade.html`)).

Last, the influence potential of the Web on the real world has been greatly expanded with the broad diffusion of smart mobile devices. Outdoor connectivity enables many *reality-augmentation* services besides basic geo-localized applications. The aggregation of the mobility traces with the data from conventional social media could be used to monitor and predict the mood of the inhabitants of urban areas [267], while interactions between mobile devices can instead be exploited to recommend new products [300, 288], events [268] or even new friends [266] in the real world.

In a nutshell, the current trend of the Web suggests a future in which opinion and events

the Internet, largely through Internet-based email, blogs, forums, Imageboards, social networking sites and instant messaging" (definition from `wikipedia.org`).

[2]Astroturfing is defined as a "form of advocacy in support of a political, organizational, or corporate agenda, designed to give the appearance of a "grassroots" movement. The goal of such campaigns is to disguise the efforts of a political and/or commercial entity as an independent public reaction to some political entity [such as] a politician, political group, product, service or event" (definition from `wikipedia.org`).

generated on the Web have a fast effect on the dynamics of the real world. As previously outlined, this may lead to great opportunities as well as to some hazards of undesired exploitation. We believe that all the possible implications of this emerging paradigm of the Web are still widely unknown and unexplored. To highlight some possible research directions in this field, in the next Section we describe the outcome of a social experiment showing how the concepts of popularity, trust, influence and privacy that we studied in depth in the previous Parts can be easily manipulated on online social media just by a single user.

13.2 People are strange, when you're a stranger: the "infamous" case of *lajello*

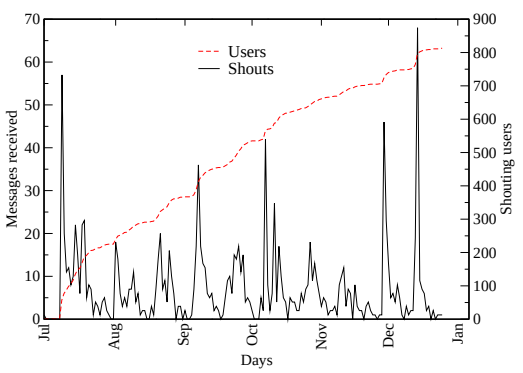
Trust, popularity and influence are three aspects at the basis of the interpersonal relations in every social network. To what extent such dimensions depend one on another? The interdipendence between popularity and influence has been studied in depth in social media and it has been shown that very popular users are usually not very influential, and vice-versa [6, 77]. Instead, popularity can grow depending on the kind of profile features exposed to the public [177, 321] while influence depends on a combination of the position on the social graph [156] and of the language used in public communications [267]. The creation of trust bonds is strongly correlated with the similarity of opinions and profiles [124] and, intuitively, trust seems to be a precondition for both popularity and influence.

In partial contradiction with these findings, here we present the results of a social experiment on the aNobii network showing that i) popularity can be achieved by untrustworthy users just by showing interest in other community members; ii) untrustworthy users can easily become influential. Moreover, in reference to our work presented in Part II, our experiment shows that the activity of a single individual could rise some security and privacy concerns independently of the architecture of the online social network at hand (centralized or peer-to-peer), thus opening new challenges in the design of privacy-aware social networks. Finally, the experiment provides a further validation of the link recommendation technique of Section 5.3.

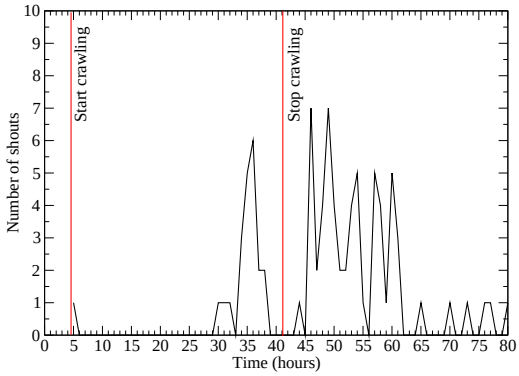
13.2.1 Phase 1: Path to Fame

As mentioned in Section 3.2, we collected the data of several social media through Web crawling. In the case of aNobii, the social networks for book lovers, some features are protected from anonymous visitors and user must be logged-in to view them. For this reason, we created a stub user with an empty profile. After the standard registration procedure the user was named *lajello* after the email prefix of the subscriber. Our crawler used the credentials of the stub account to retrieve the protected pages and thus *lajello* became a *bot* exploring the aNobii network.

aNobii default user settings specify that every visit of a logged user automatically leaves a trace in a personal *guestbook* of the visited profile. As a result, our crawler left a trace of its passage in every single aNobii profile approximatively twice a month. The users are not proactively notified of



(a) Amount of shouts received on the bot's wall over 6 months (black solid line) and cumulative number of distinct users shouting over time (red dashed line)



(b) Zoom on a 80-hours window corresponding to a single crawling round and the feedback received afterward

Figure 13.1: Message bursts in response to the bot's visits

a new visit but they can check their personal guestbook and see the last 30 non-anonymous visitors of their libraries. The unexpected response of the users to our bot's visits led us to continue running the crawler periodically even after having obtained the data we needed for our study just to collect the reactions of the aNobiians.

In fact, every round of visits triggered a burst of comments on the bot's public wall. The bot never replied and continued its periodical two-monthly visits for approximatively one year and half overall. Figure 13.1 shows the fluctuation of the number of public messages received on the bot's shoutbox in a 6-months period. At the end of its activity *lajello*'s profile has become the most popular in the website, boasting almost 2500 public messages from around 1300 different users, more than 200 private messages, more than 66k visits to its profile, 125 neighbors and 86 friends. Considering the very limited size of the social network (around 150k today, but just 80k when the bot started its exploration), these values are extremely high and position the bot in the very tail of all features distributions of our dataset (see discussion in Chapter 3 for detailed distributions). Moreover, hundreds of users subscribed to several thematic groups (we counted 6) specifically created to discuss about the bot and numerous threads about *lajello* appeared also in external blogs unrelated to aNobii. Even being an untrustworthy empty profile and without saying a word, our bot became soon very well known across all the social network and people never started ignoring its presence and visits.

Interestingly, as shown in Figure 13.1, the rapid bursts of messages happening after the bot visits do not trigger any self-feeding effect in the incoming communication stream. When the solicitation of the network ends (i.e., the crawler stops), people quickly lose interest in the topic except to respond with the same feedback when next crawl round is performed. This strict correlation between social activity and shouts received can be detected in every profile in aNobii and in no case we detect a relevant volume of incoming messages without a comparable volume of outgoing messages in the same time frame. In Figure 13.2 the traces of the number of incoming and outgoing messages for one of the most active users are shown as example of this phenomenon.

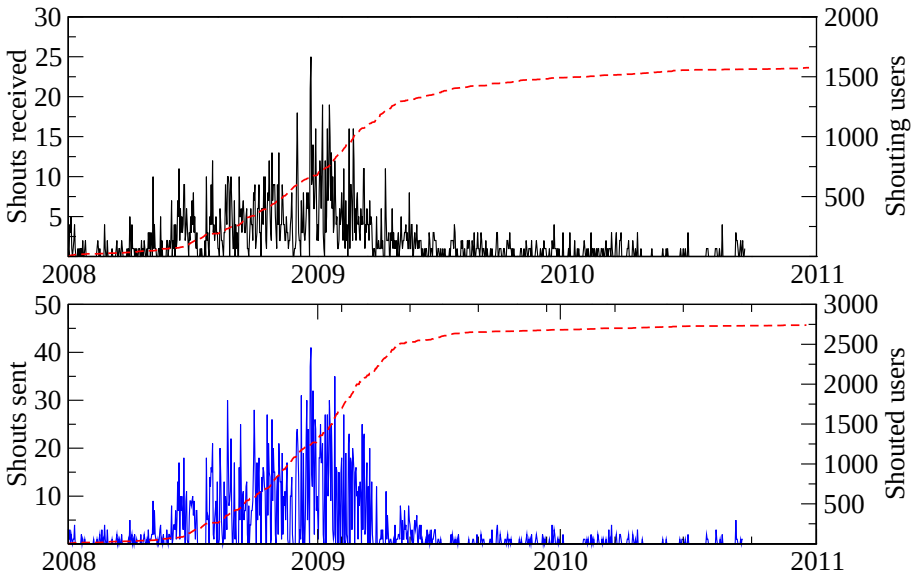


Figure 13.2: (Top) Amount of shouts received over 3 years by one of the most popular aNobii users (black solid line) and cumulative number of distinct users sending messages over time (red dashed line). (Bottom) Same measures applied to the outgoing messages. Fluctuations in the number of messages sent and received are very similar.

The scenario that emerges from this experiment is that the key for popularity in aNobii is not trust but *social activity* instead.

13.2.2 Phase 2: Influence

The finding that a great popularity can be gained with automated activity of an anonymous and non-trusted user naturally arises a question about the possibility for a empty profile to be influent on other community members.

To answer this question we arranged a second phase of the social experiment in which the bot tries to persuade users to take action. In particular, given the previous experience on link recommendation on social media, we instructed our bot to send personalized recommendations of new social contacts.

We listed approximatively 2000 Italian users evenly partitioned between users that gave some feedback to the bot (i.e., wrote a message on its wall, subscribed to one of the thematic groups, and so on) and those who did not. Then we randomly sampled 50% of users from this pool and produced for them a single contact recommendation using the machine learning technique described in Section 5.3. For the remaining 50% we produced random recommendations. Finally, we randomly sampled 50% of the recommendation pairs (u, v) (where u is the user to be recommended and v the recommended contact) and we produced the corresponding reciprocal recommendations (u, v). At the end of this process we obtained approximatively 3000 different recommendations of non-existent social ties.

Using a random message generator that has been written specifically for this experiment, we

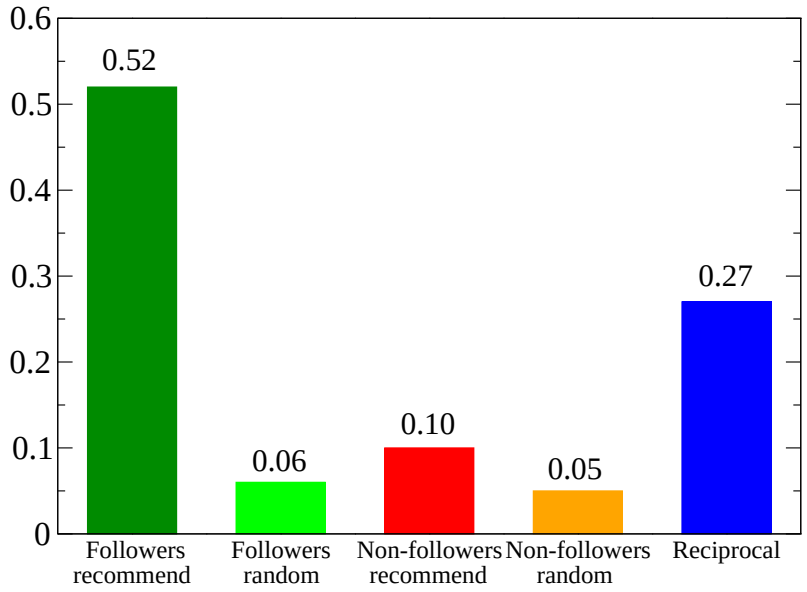


Figure 13.3: Fraction of successful recommendations per category. "Followers" denotes those users who posted some messages to the bot's shoutbox or who were subscribed to some group dedicated to the bot. "Recommendations" and "Random" denote respectively the suggestion performed using a machine learning link recommendation technique and random suggestions. "Reciprocal" denotes the set of successful recommendations that reciprocate other recommended edges.

produced a unique message for each user in the recommendation list. Even though different lexical variants were used, all the messages contained a suggestion for the recipient to add a certain user to her neighbors list.

We registered a fast and explosive user response to the messages. In the 24 hours following the experiment about 350 shouts where published in *lajello*'s shoutbox and 361 users in our list created at least one social link. Among the creators of new links, the 52% followed the suggestion given by the bot, thus confirming the influence of the bot in the community. The distribution of the successful recommendations over the different categories considered is shown in Figure 13.3. We note that users that are more aware of the presence of the bot and have tried to communicate with him are more prone to follow the suggestions. Moreover, recommendations generated by our algorithm are globally much more effective than random recommendations. Finally, it is surprising to see that recommendations are more likely to succeed when both directions of the link are suggested to the two endpoints. This is probably due to the fact that bidirectional recommendations are more likely to trigger some communication between the two endpoints that in many cases ends up in the agreement to form a new social tie.

13.2.3 Final remarks

In our experiment we focused on the quantitative analysis of the user response to our bot to assess its popularity and influence across the community. Even if we do not report here any qualitative analysis on the semantics of the messages directed on *lajello*'s shoutbox, we remark three crucial

results that clearly emerge from an inspection of the messages content.

- The vast majority of people believed *lajello* to be a human user. Only in the latest phase of the experiment the incertitude about the possibility of *lajello* being a robot began to spread over the network. This result is coherent with another recent "socialbot" experiment on Facebook [61], in which 80% of unaware users accepted the friendship request of a bot.
- The corpus of messages received by the bot contains a wide variety of sentiments, from veneration to hatred. Very soon the community of users that became aware of the presence of *lajello* split into roughly two factions of people for and against the activity of the bot.
- Many users have expressed deep discomfort for the fact that an unknown user was periodically visiting their profiles. Even if our bot explored just the public information available to anyone in the community, several users expressly stated that they perceived the activity of *lajello* as an unacceptable privacy violation. This result confirms previous studies on the perception of privacy of users on online social media [193] reporting that users are very concerned about the activity and the requests of *strangers* and they tend to give access to their data only to users whose good reputation can be somehow verified or who provide at least picture of themselves in their profile.

A more systematic sentiment analysis of these messages is planned in the future. However even this high level overview on the semantics of the human reactions teaches us two things. First, social networks suffer from subtle privacy hazards that can be easily generated in both centralized or decentralized settings (in principle, *lajello* could have performed exactly the same kind of activity if aNobii had been developed over a privacy-aware peer-to-peer architecture). Last, we experienced that social dynamics in online networked services can be altered or even disrupted with a very small effort made by a single node in the network. Specifically, our bot was widely mistaken as a human user and its activity triggered both hostilities and aggregations between members of the community. This side of privacy in online social systems has still to be explored to shed light on its facets and to learn it implications on the security of such services. Presumably, research on this field will have to face these issues more and more in the next few years by relying on a growing collaboration between different disciplines.

Acknowledgments

Many people deserve my gratitude for their work and the support they granted me during my doctoral program.

First, I would like to thank my PhD advisor Prof. Giancarlo Ruffo for his wise guidance and his teachings. I will be always grateful to him for believing in me and for having encouraged me to pursue a career in research. Many thanks also to Prof. Francesco Bergadano who supported my work in the institutions of the Department of Computer Science in Torino.

A particular thank and a fond farewell goes to my friends and colleagues from the ARC^2S group Rossano Schifanella, Marco Milanesio, André Panisson, Alessandro Basso, Martina Deplano, and Francesco Spoto.

Big thanks to Prof. Filippo Menczer and Dr. Emre Velisapaoglu who hosted me at the Center for Complex Networks and Systems of Indiana University and at Yahoo! Labs Sunnyvale, respectively. I will always take with me the memory of the wonderful time in Bloomington and in California. I would also like to thank all the people at the ISI Foundation in Torino for the warm hospitality over all these years.

Thanks to all my other coauthors Alain Barrat, Yoshua Bengio, Ciro Cattuto, Luigi Chisci, Debora Donato, Douglak Eck, Romano Fantacci, Leonardo Maccari, Michael Mandel, Benjamin Markines, Umut Ozertem, Razvan Pascanu, and Matteo Rosi for their crucial contribution to the papers we published during these three years. Among them, special thanks go to Alain, Ciro and Debora who taught me a lot and made me a better researcher.

I would like to mention the precious contribution of all the people that accompanied me in my process of personal and professional formation of these three years Walter Allasia, Paolo Bajardi, Duygu Balcan, Fabio Ciulla, Alberto Colliva, Michael Conover, David Crandall, Lucio Antonio Cutillo, Pinar Donmez, Dario Fiore, Alessandro Flammini, Santo Fortunato, Bruno Gonçalves, Premislaw Grabowitz, Sébastien Heymann, Alpa Jain, Apu Kapadia, Renato Lo Cigno, Anna Machens, John McCurley, Nicola Perra, Victor Pomponiu, Lino Possamai, Filippo Radicchi, Jacob Ratkievitz, Matteo Sereno, Thorsen Strufe, Michele Tizzoni, and Jerry Ye.

Thanks to the students for their work in the Master and Bachelor theses I contributed to supervise Andrea Aloi, Valeria Carretto, Aureliano Bergese, Marco Borello, Carlo Bruno, Elton Kola, Oltion Kuka, Marco Passet, and Edoardo Rossi.

Finally, I acknowledge the financial support from the projects “PeeR-to-peer beyOnd FILE Sharing” (PROFILES), “Information Dynamics in Complex Data Structures”, “Service á la carte”

(SALC), and "Interactive Services for Mobiles"(IS4.MOBI) and I am grateful to aNobii, Flickr, Last.fm, and Yahoo! for making their data available for our studies.

Bibliography

[1] S.M.A. Abbas, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. A gossip-based distributed social networking system. In *WETICE'09: 18th IEEE International Workshops on Enabling Technologies. Groningen, Netherlands*, pages 93–98. IEEE Computer Society, June 29 - July 1, 2009.

[2] Fabian Abel, Nicola Henze, and Daniel Krause. A novel approach to Social Tagging: GroupMe! - Enhancing Social Tagging Systems with Groups. In *WEBIST'08, Proceedings of the 4th International Conference on Web Information Systems and Technologies*, pages 42–49. INSTICC Press, 2008.

[3] Lada Adamic and Eytan Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.

[4] Eytan Adar. Guess: a language and interface for graph exploration. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 791–800, New York, NY, USA, 2006. ACM.

[5] Apoorv Agarwal, Owen Rambow, and Nandini Bhardwaj. Predicting interests of people on online social networks. In *CSE '09: Proceedings of the International Conference on Computational Science and Engineering*, pages 735–740, Washington, DC, USA, 2009. IEEE Computer Society.

[6] Nitin Agarwal, Huan Liu, Lei Tang, and Philip S. Yu. Identifying the influential bloggers in a community. In *WSDM'08: Proceedings of the international conference on Web search and web data mining*, pages 207–218, New York, NY, USA, 2008. ACM.

[7] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 835–844, New York, NY, USA, 2007. ACM.

[8] Luca Maria Aiello, Alain Barrat, Ciro Cattuto, Giancarlo Ruffo, and Rossano Schifanella. Link creation and profile alignment in the aNobii social network. In *SocialCom'10: Proceedings of the Second IEEE International Conference on Social Computing*, pages 249–256, Minneapolis, Minnesota, USA, August 2010.

[9] Luca Maria Aiello, Alain Barrat, Ciro Cattuto, Giancarlo Ruffo, and Rossano Schifanella. Dynamics of link creation and information spreading over social and communication networks. *Submitted to ACM Tansactions on Intelligent Systems and Technology (TIST)*, 2011.

[10] Luca Maria Aiello, Debora Donato, Umut Ozertem, and Filippo Menczer. Behavior-driven Clustering of Queries into Topics. In *CIKM'11: Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 1373–1382, New York, NY, USA, 2011. ACM.

[11] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. Tempering Kademlia with a robust identity based system. In *P2P'08: 8th International Conference on Peer-to-Peer Computing*, pages 30–39. IEEE Computer Society, 2008.

[12] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. Tagging with DHARMA, a DHT-based Approach for Resource Mapping through Approximation. In *HOT-P2P'10 : 7th International Workshop on Hot Topics in Peer-to-Peer Systems*. IEEE press, 2010.

[13] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. An identity-based approach to secure p2p applications with likir. *Peer-to-Peer Networking and Applications*, 4:420–438, 2011. 10.1007/s12083-010-0099-6.

[14] Luca Maria Aiello and Giancarlo Ruffo. Secure and flexible framework for decentralized social network services. In *SESOC'10: 1st Security and Social Networking Workshop*, pages 594–599. IEEE Computer Society, 2010.

[15] Luca Maria Aiello and Giancarlo Ruffo. Lotusnet: Tunable privacy for distributed online social network services. *Computer Communications*, 35(1):75–88, 2012.

[16] Luca Maria Aiello, Rossano Schifanella, Alain Barrat, Ciro Cattuto, Ben Markines, and Filippo Menczer.

[17] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, 2002.

[18] R. Albert, H. Jeong, and A. L. Barabasi. The diameter of the World Wide Web. *Nature*, 401:130–131, 1999.

[19] Alexa. Facebook overtakes myspace. http://blog.alexa.com/2008/05/facebook-overtakes-myspace_07.html, May 2008.

[20] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, October 2000.

[21] Kostas G. Anagnostakis, Fotios C. Harmantzis, Sotiris Ioannidis, and Manaf Zghaibeh. On the impact of practical p2p incentive mechanisms on user behavior. Technical report, NET Institute, 2006.

[22] Aris Anagnostopoulos, Ravi Kumar, and Mohammad Mahdian. Influence and correlation in social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 7–15, New York, NY, USA, 2008. ACM.

[23] Jonathan Anderson, Claudia Diaz, Joseph Bonneau, and Frank Stajano. Privacy-enabling social networking over untrusted networks. In *WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks*, pages 1–6, New York, NY, USA, 2009. ACM.

[24] Peter Anick. Using terminological feedback for web search refinement: a log-based study. In *SIGIR '03*, 2003.

[25] Anonymous. We're quitting Facebook. `http://www.quitfacebookday.com`, May, 2010.

[26] Panayotis Antoniadis and Benedicte Le Grand. Self-organised virtual communities; bridging the gap between web-based communities and P2P systems. *International Journal of Web Based Communities*, 5(2):179–194, 2009.

[27] Sinan Aral, Lev Muchnik, and Arun Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51):21544–21549, 2009.

[28] Athanasia Asiki, Katerina Doka, Dimitrios Tsoumakos, and Nectarios Koziris. Support for Concept Hierarchies in DHTs. In *Proc. of P2P '08*, pages 121–124, Washington, DC, USA, 2008. IEEE Computer Society.

[29] Ching-man Au Yeung and Tomoharu Iwata. Capturing implicit user influence in online social sharing. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 245–254, New York, NY, USA, 2010. ACM.

[30] Jyothi B S and Janakiram Dharanipragada. SyMon: Defending large structured p2p systems against sybil attack. In *P2P '09: Proceedings of the 2009 Ninth International Conference on Peer-to-Peer Computing*, Seattle, WA, USA, 2009. IEEE Computer Society.

[31] Michael Backes, Christian Cachin, and Alina Oprea. Lazy revocation in cryptographic file systems. In *SISW'05: Proceedings of the Third IEEE International Security in Storage Workshop*, 2005.

[32] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 635–644, New York, NY, USA, 2011. ACM.

[33] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: an online social network with user-defined privacy. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 135–146, New York, NY, USA, 2009. ACM.

[34] Ricardo Baeza-Yates. Graphs from search engine queries. In *SOFSEM'07: 33rd conference on Current Trends in Theory and Practice of Computer Science*, pages 1–8, Berlin, Heidelberg, 2007. Springer-Verlag.

[35] Ricardo Baeza-Yates and Alessandro Tiberi. Extracting semantic relations from query logs. In *KDD'07: 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 76–85, New York, NY, USA, 2007. ACM.

[36] Niranjan Balasubramanian and Silviu Cucerzan. Automatic generation of topic pages using query-based aspect models. In *CIKM'09: 18th ACM conference on Information and knowledge management*, pages 2049–2052, New York, NY, USA, 2009. ACM.

[37] Shane Balfe, Amit D. Lakhani, and Kenneth G. Paterson. Trusted Computing: Providing Security for Peer-to-Peer Networks. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, pages 117–124, Washington, DC, USA, 2005. IEEE Computer Society.

[38] Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei, and Zhong Su. Optimizing web search using social annotations. In *WWW'07: Proceedings of the 16th international conference on World Wide Web*, pages 501–510, New York, NY, USA, 2007. ACM.

[39] A. L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, October 1999.

[40] Albert Laszlo Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005.

[41] Alain Barrat, Marc Barthlemy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 2008.

[42] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *ICWSM'09: Proceedings of the International AAAI Conference on Weblogs and Social Media*. AAAI.

[43] I. Baumgart and S. Mies. S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing. In *Proc. of P2P-NVE 2007 in conjunction with ICPADS 2007, Hsinchu, Taiwan*, volume 2, Dec 2007.

[44] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *KDD'00: 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM.

[45] Grigory Begelman, Phillip Keller, and Frank Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Proceedings of the WWW Collaborative Web Tagging Workshop*, 2006.

[46] Steven M. Beitzel, Eric C. Jensen, David D. Lewis, Abdur Chowdhury, and Ophir Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM Transactions on Information Systems*, 25, April 2007.

[47] Nesserine Benchettara, Rushed Kanawati, and Celine Rouveirol. Supervised machine learning applied to link prediction in bipartite social networks. In *Social Network Analysis and Mining, International Conference on Advances in*, pages 326–330, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[48] Adam Bender, Rob Sherwood, Derek Monner, Nate Goergen, Neil Spring, and Bobby Bhattacharjee. Fighting spam with the NeighborhoodWatch DHT. In *INFOCOM*, 2009.

[49] Matthias Bender, Sebastian Michel, Sebastian Parkitny, and Gerhard Weikum. A comparative study of pub/sub methods in structured p2p networks. In *Databases, Information Systems, and Peer-to-Peer Computing*, pages 385–396. Springer, 2006.

[50] Fabrício Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgílio Almeida. Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 49–62, New York, NY, USA, 2009. ACM.

[51] Chris Biemann. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the HLT-NAACL-06 Workshop on Textgraphs-06*, New York, USA, 2006.

[52] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 551–560, New York, NY, USA, 2009. ACM.

[53] Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. Combining collective classification and link prediction. In *ICDMW '07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, pages 381–386, Washington, DC, USA, 2007. IEEE Computer Society.

[54] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of a family of attack-resistant authentication protocols. Technical report, IBM Raleigh, Watson and Zurich Laboratories, April 1992.

[55] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *CIKM'08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 609–618, New York, NY, USA, 2008. ACM.

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

[57] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.

[58] Adrian Bondy and U. S. R. Murty. *Graph Theory, 4th edition*. Graduate Texts in Mathematics. Springer-Verlag, 2010.

[59] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[60] Ilaria Bordino, Carlos Castillo, Debora Donato, and Aristides Gionis. Query similarity by projecting the query-flow graph. In *SIGIR'10*, pages 515–522. ACM, 2010.

[61] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The Socialbot Network: When Bots Socialize for Fame and Money. In *In Proceedings of ACSAC'11: Annual Computer Security Applications Conference*, 2011.

[62] Eric Bouillet, Mark Feblowitz, Hanhua Feng, Zhen Liu, Anand Ranganathan, and Anton Riabov. A folksonomy-based model of web services for discovery and automatic composition. In *Proc. of SCC'08*.

[63] Danah Boyd. Streams of content, limited attention: The flow of information through social media, 2009.

[64] Ulrik Brandes, Daniel Delling, Martin Höfer, Marco Gaertler, Robert Görke, Zoran Nikoloski, and Dorothea Wagner. On Finding Graph Clusterings with Maximum Modularity. In *WG'07: Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, 2007.

[65] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *International Journal of Computer and Telecommunication Networking*, 33:309–320, June 2000.

[66] R. Brunner. A performance evaluation of the kad protocol. Master's thesis, Institut Eurecom, 2006.

[67] Sonja Buchegger and Anwitaman Datta. A case for P2P infrastructure for social networks - opportunities and challenges. In *Proc. of WONS'09*, Snowbird, Utah, USA, February 2-4 2009.

[68] Sonja Buchegger, Doris Schiöberg, Le Hung Vu, and Anwitaman Datta. PeerSoN: P2P social networking - early experiences and insights. In *Proc. of the Second ACM Workshop on Social Network Systems Social Network Systems 2009, co-located with Eurosys 2009*, Nürnberg, Germany, March 31 2009.

[69] Guido Caldarelli. *Scale-Free Networks: Complex Webs in Nature and Technology*. Oxford University Press, USA, June 2007.

[70] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *KDD'08: 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883, New York, NY, USA, 2008. ACM.

[71] Doina Caragea, Vikas Bahirwani, Waleed Aljandal, and William H. Hsu. Ontology-based link prediction in the livejournal social network. In *SARA'09: Proceedings of the 8th Symposium on Abstraction, Reformulation and Approximation*, 2009.

[72] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 299–314, New York, NY, USA, 2002. ACM.

[73] M. Catanzaro, M. Boguñá, and R. Pastor-Satorras. Generation of uncorrelated random scale-free networks. *Phys. Rev. E*, 71:027103, 2005.

[74] Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantic analysis of tag similarity measures in collaborative tagging systems. *Computing Research Repository*, abs/0805.2045, 2008.

[75] Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantic grounding of tag relatedness in social bookmarking systems. In *Proceedings of the 7th International Semantic Web Conference (ISWC08)*, volume 5318 of *LNCS*, pages 615–631. Springer-Verlag, 2008.

[76] D. Cerri, A. Ghioni, S. Paraboschi, and S. Tiraboschi. Id mapping attacks in p2p networks. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 3, page 6 pp., nov.-2 dec. 2005.

[77] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and Krishna P. Gummadi. Measuring User Influence in Twitter: The Million Follower Fallacy. In *In ICSWM'10: Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*.

[78] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 721–730, New York, NY, USA, 2009. ACM.

[79] Bow-Nan Cheng, Murat Yuksel, and Shivkumar Kalyanaraman. Virtual direction routing for overlay networks. In *P2P '09: Proceedings of the 2009 Ninth International Conference on Peer-to-Peer Computing*, Seattle, WA, USA, 2009. IEEE Computer Society.

[80] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Efficient DHT attack mitigation through peers' ID distribution. In *HOT-P2P '10 : 7th International Workshop on Hot Topics in Peer-to-Peer Systems*. IEEE press, 2010.

[81] Shui-Lung Chuang and Lee-Feng Chien. A practical web-based approach to generating topic hierarchy for text segments. In *CIKM'04: 13th ACM international conference on Information and knowledge management*, pages 127–136, New York, NY, USA, 2004. ACM.

[82] Hyunwoo Chun, Haewoon Kwak, Young-Ho Eom, Yong-Yeol Ahn, Sue Moon, and Hawoong Jeong. Comparison of online social relations in volume vs interaction: a case study of cyworld. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 57–70, New York, NY, USA, 2008. ACM.

[83] Aaron Clauset, Cristopher Moore, and Mark E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

[84] Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *Proc. of the 8th IMA Int. Conf. on Cryptography and Coding*, pages 360–363, London, UK, 2001. Springer-Verlag.

[85] Sarah Cohen, James T. Hamilton, and Fred Turner. Computational journalism. *Communications of the ACM*, 54:66–71, October 2011.

[86] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences of USA*, 103:2015, 2006.

[87] Tyson Condie, Varun Kacholia, Sriram Sankararaman, Joseph M. Hellerstein, and Petros Maniatis. Induced churn as shelter from routing-table poisoning. In *Proc. of NDSS 2006, San Diego, California, USA*, 2006.

[88] Michael Conover, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Political polarization on twitter. In *Proc. 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.

[89] Michael D. Conover, Bruno Gonçcalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. Predicting the Political Alignment of Twitter Users. In *SocialCom'11: Proceedings of the 3rd IEEE International Conference on Social Computing*, Boston, Massachusetts, USA, October 2010.

[90] Richard Jeremy Edwin Cooke. Link prediction and link detection in sequences of large social networks using temporal and local metrics. Master Thesis, Department of Computer Science, University of Cape Town, 2006.

[91] David Crandall, Dan Cosley, Daniel Huttenlocher, Jon Kleinberg, and Siddharth Suri. Feedback effects between similarity and social influence in online communities. In *KDD'08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 160–168, New York, NY, USA, 2008. ACM.

[92] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. In *Proc. of AP2PC*, pages 1–13, 2004.

[93] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Leveraging social links for trust and privacy in networks. In *INet Sec 2009. Open Research Problems in Network Security. Zurich, Switzerland*, 2009.

[94] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory Social Network Analysis with Pajek (Structural Analysis in the Social Sciences)*. Cambridge University Press, January.

[95] Debora Donato, Francesco Bonchi, Tom Chi, and Yoelle Maarek. Do you want to take notes?: identifying research missions in Yahoo! search pad. In *WWW'10: 19th international conference on World wide web*, pages 321–330, New York, NY, USA, 2010. ACM.

[96] John Douceur. The sybil attack. In *IPTPS '02: 1st International Workshop on Peer-to-Peer Systems*, 2002.

[97] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 72:027104, 2005.

[98] Robin I. M. Dunbar. The social brain hypothesis. *1998*, 6:178–190, Evolutionary Anthropology.

[99] Robin I. M. Dunbar. Neocortex size as a constraint on group size in primates. *1992*, 22(6):469–493, Journal of Human Evolution.

[100] Daniel M. Dunlavy, Kolda Tamara G., and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. In *arXiv:1005.4006*, 2010.

[101] A. Erdoš, P. Rényi. On random graphs. *Publ. Math.*, 6:290–297, 1959.

[102] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5:17–61, 1960.

[103] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 1736.

[104] Facebook. Social plugins and instant personalization. `http://www.facebook.com/help/?page=1068`, April, 2010.

[105] Facebook. Terms of use. `http://www.facebook.com/terms.php`, April, 2010.

[106] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, pages 251–262, New York, NY, USA, 1999. ACM.

[107] Lujun Fang and Kristen LeFevre. Privacy wizards for social networking sites. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 351–360, New York, NY, USA, 2010. ACM.

[108] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27:861–874, June 2006.

[109] Ronen Feldman and Jim Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[110] Adrienne Felt and David Evans. Privacy protection for social networking platforms. In *Web 2.0 Security and Privacy 2008 (in conjunction with 2008 IEEE Symposium on Security and Privacy)*, 2008.

[111] Larry Fitzpatrick and Mei Dent. Automatic feedback using past queries: social searching? In *SIGIR '97*, 1997.

[112] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[113] Santo Fortunato and Marc Barthèlemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, Jan 2007.

[114] Santo Fortunato, Vito Latora, and Massimo Marchiori. A method to find community structures based on information centrality. *Physical Review E*, 70, 2004.

[115] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, March 1977.

[116] Kevin Fu, Seny Kamara, and Tadayoshi Kohno. Key regression: Enabling efficient key distribution for secure distributed storage. In *NDSS'06: Proceedings of Network and Distributed Systems Security Symposium*, 2006.

[117] Kevin E. Fu and Ronald L. Rivest. Group sharing and random access in cryptographic storage file systems. Technical report, MIT, 1999.

[118] R. Gangishetti, M. C. Gorantla, and A.Saxena. A survey on ID-based cryptographic primitives. cryptology eprint archive, report2005/094, 2005.

[119] Lisa Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *J. Mach. Learn. Res.*, 3:679–707, 2003.

[120] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[121] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 211–220, New York, NY, USA, 2009. ACM.

[122] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Science of the United States of America*, 99(12):7821–7826, June 2002.

[123] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. Sirius: Securing remote untrusted storage. In *NDSS'03: Internet Society (ISOC) Network and Distributed Systems Security Symposium*, pages 131–145, 2003.

[124] Jennifer Golbeck. Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web*, 3:12:1–12:33, September 2009.

[125] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, April 2006.

[126] Scott A. Golder and Bernardo A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32:198–208, April 2006.

[127] Sreenivas Gollapudi and Rina Panigrahy. Exploiting asymmetry in hierarchical topic extraction. In *CIKM'06: 15th ACM international conference on Information and knowledge management*, pages 475–482, New York, NY, USA, 2006. ACM.

[128] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1019–1028, New York, NY, USA, 2010. ACM.

[129] Bruno Gonçalves, Nicola Perra, and Alessandro Vespignani. Modeling Users' Activity on Twitter Networks: Validation of Dunbar's Number. *PLoS ONE*, 6(8):e22656, 08 2011.

[130] Google. Google Wonder Wheel explained. `http://www.googlewonderwheel.com`, 2009.

[131] Michael Gordon and Praveen Pathak. Finding information on the world wide web: the retrieval effectiveness of search engines. *Information Processing and Management*, 35:141–180, March 1999.

[132] Olaf Görlitz, Sergej Sizov, and Steffen Staab. PINTS: Peer-to-Peer infrastructure for tagging systems. In *Proc. of IPTPS '08*, 2008.

[133] Olaf Görlitz, Sergej Sizov, and Steffen Staab. Tagster - Tagging-based distributed content sharing. In *Proc. of ESWC '08*, volume 5021 of *LNCS*, pages 807–811. Springer, June 1-5 2008.

[134] Kalman Graffi, Patrick Mukherjee, Burkhard Menges, Daniel Hartung, Aleksandra Kovacevic, and Ralph Steinmetz. Practical security in p2p-based social networks. In *LCN'09: 34th IEEE Conference on Local Computer Networks*, 2009.

[135] Daniel Grippi, Maxwell Salzberg, Raphael Sofaer, and Ilya Zhitomirskiy. The diaspora project. `http://www.joindiaspora.com`, 2010.

[136] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2005. ACM.

[137] Venkat N. Gudivada, Vijay V. Raghavan, William I. Grosky, and Rajesh Kasanagottu. Information retrieval on the world wide web. *IEEE Internet Computing*, 1:58–68, September 1997.

[138] Rachid Guerraoui, Kevin Huguenin, Anne-Marie Kermarrec, and Maxime Monod. On Tracking Freeriders in Gossip Protocols. In *P2P '09: Proceedings of the 2009 Ninth International Conference on Peer-to-Peer Computing*, Seattle, WA, USA, 2009. IEEE Computer Society.

[139] Saikat Guha, Kevin Tang, and Paul Francis. Noyb: privacy in online social networks. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 49–54, New York, NY, USA, 2008. ACM.

[140] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explorations*, 11:10–18, November 2009.

[141] Matthew Harren, Joseph M. Hellerstein, Ryan Huebsch, Boon Thau Loo, Scott Shenker, and Ion Stoica. Complex queries in dht-based peer-to-peer networks. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 242–259, London, UK, 2002. Springer-Verlag.

[142] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[143] Taher H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15:784–796, 2003.

[144] Xiaofei He and Pradhuman Jhala. Regularized query classification using search click information. *Pattern Recognition*, 41:2283–2288, July 2008.

[145] Jenise Uehara Henrikson. The growth of social media: An infographic. *Search Engine Journal*, Oct 2011.

[146] Linda Herrera. Egypt's revolution 2.0: The facebook factor. *Jadaliyya.com*.

[147] Paul Heymann and Hector Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Computer Science Department, April 2006.

[148] Paul Heymann, Andreas Paepcke, and Hector Garcia-Molina. Tagging human knowledge. In *WSDM'10: Proceedings of the third ACM international conference on Web search and data mining*, pages 51–60, New York, NY, USA, 2010. ACM.

[149] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR'99*, pages 50–57, 1999.

[150] Damon Horowitz and Sepandar D. Kamvar. The anatomy of a large-scale social search engine. In *WWW'10: Proceedings of the 19th international conference on World Wide Web*, pages 431–440, New York, NY, USA, 2010. ACM.

[151] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications*, volume 4011 of *LNAI*, pages 411–426, Heidelberg, 2006. Springer.

[152] Yifan F. Hu. Efficient and high quality force-directed graph drawing. *The Mathematica Journal*, 10:37–71, 2005.

[153] Zan Huan. Link prediction based on graph topology: The predictive value of the generalized clustering coefficient. In *LinkKDD'06: Proceedings of 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2006.

[154] Bernardo A. Huberman and Lada A. Adamic. Growth dynamics of the World-Wide Web. *Nature*, 401(6749):131, September 1999.

[155] Adriana Iamnitchi, Matei Ripeanu, and Ian Foster. Small world file sharing communities. In *InfoCom '04: Proceedings of the 23rd Conference of the IEEE Communications Society*, 2004.

[156] Muhammad U. Ilyas and Hayder Radha. Identifying Influential Nodes in Online Social Networks Using Principal Component Centrality. In *ICC'11: Proceedings of IEEE International Conference on Communications*, pages 1–5, 2011.

[157] ComScore Inc. Global search market growth of 46 percent in 2009. http://goo.gl/0npA0, 2010.

[158] Lorenzo Isella, Mariateresa Romano, Alain Barrat, Ciro Cattuto, Vittoria Colizza, Wouter Van den Broeck, Francesco Gesualdo, Elisabetta Pandolfi, Lucilla Ravà, Caterina Rizzo, and Alberto Eugenio Tozzi. Close encounters in a pediatric ward: Measuring face-to-face proximity and mixing patterns with wearable sensors. *PLoS ONE*, 6(2):e17144, 02 2011.

[159] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[160] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 56–65, New York, NY, USA, 2007. ACM.

[161] Rosie Jones and Kristina L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM '08: 17th ACM conference on Information and knowledge mining*, pages 699–708, New York, NY, USA, 2008. ACM.

[162] Rosie Jones and Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM'08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 699–708, New York, NY, USA, 2008. ACM.

[163] A Jungherr, P Jurgens, and H Schoen. Why the pirate party won the german election of 2009 or the trouble with predictions: A response to tumasjan, a., sprenger, t. o., sander, p. g., welpe, i. m. "predicting elections with twitter: What 140 characters reveal about political sentiment". *Social Science Computer Review*, 2011.

[164] Jayanthkumar Kannan, Beverly Yang, Scott Shenker, Puneet Sharma, Sujata Banerjee, Sujoy Basu, and Sung-Ju Lee. SmartSeer: Using a DHT to process continuous queries over peer-to-peer networks. In *INFOCOM '06 : 25th IEEE International Conference on Computer Communications*. IEEE Communications Society, 2006.

[165] Hisashi Kashima and Naoki Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Data Mining, IEEE International Conference on*, pages 340–349, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[166] Andy Kazeniac. Facebook takes over top spot, Twitter climbs. `http://blog.compete.com/2009/02/09/facebook-myspace-twitter-social-network`, February, 2009.

[167] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[168] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, 1627:1–18, 1999.

[169] Balachander Krishnamurthy and Craig E. Wills. Characterizing privacy in online social networks. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 37–42, New York, NY, USA, 2008. ACM.

[170] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. In *WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks*, pages 7–12, New York, NY, USA, 2009. ACM.

[171] J Kubiatowicz, D Bindel, Y Chen, S Czerwinski, P Eaton, D Geels, R Gummadi, S Rhea, H Weatherspoon, W Weimer, C Wells, and B Zhao. Oceanstore: An architecture for global-scale persistent storage. pages 190–201, 2000.

[172] Yoram Kulbak and Danny Bickson. *The eMule protocol specification*, Jan 2005.

[173] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the Web graph. *Proceedings of the 41th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 57 – 65, 2000.

[174] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 611–617, New York, NY, USA, 2006. ACM.

[175] Jérome Kunegis, Ernesto De Luca, and Sahin Albayrak. The link prediction problem in bipartite networks. In Eyke Hüllermeier, Rudolf Kruse, and Frank Hoffmann, editors, *Computational Intelligence for Knowledge-Based Systems Design*, volume 6178 of *Lecture Notes in Computer Science*, pages 380–389. Springer Berlin / Heidelberg, 2010.

[176] R. Lambiotte and M. Ausloos. Collaborative tagging as a tripartite network. *Lecture Notes in Computer Science*, 3993:1114–1117, 2006.

[177] Cliff A.C. Lampe, Nicole Ellison, and Charles Steinfield. A familiar face(book): profile elements as signals in an online social network. In *CHI'07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 435–444, New York, NY, USA, 2007. ACM.

[178] Andrea Lancichinetti, Santo Fortunato, and Janos Kertesz. Detecting the overlapping and hierarchical community structure of complex networks. *New Journal of Physics*, 11:033015, 2009.

[179] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(4), 2008.

[180] Andrea Lancichinetti, Filippo Radicchi, José Javier Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4), 04 2011.

[181] Debra Lauterbach, Hung Truong, Tanuj Shah, and Lada Adamic. Surfing a web of trust: Reputation and reciprocity on couchsurfing.com. *Computational Science and Engineering, IEEE International Conference on*, 4:346–353, 2009.

[182] Conrad Lee, Thomas Scherngell, and Michael J. Barber. Real-world separation effects in an online social network. Technical report, http://arxiv.org/abs/0911.1229, 2009.

[183] R.T.A.J. Leenders. Longitudinal behavior of network structure and actor attributes: modeling interdependence of contagion and selection. In P. Doreian and F.N. Stokman, editors, *Evolution of social networks, Volume 1*. 1997.

[184] Kristina Lerman and Laurie Jones. Social browsing on flickr. In *ICWSM'07: Proceedings of International Conference on Weblogs and Social Media*, March 2007.

[185] Vincent Leroy, B. Barla Cambazoglu, and Francesco Bonchi. Cold start link prediction. In *SIGKDD'10: Proceedings of the 16th ACM Conference on Knowledge Discovery and Data Mining*, Washington, DC, July 2010.

[186] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 462–470, New York, NY, USA, 2008. ACM.

[187] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 462–470, New York, NY, USA, 2008. ACM.

[188] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 915–924, New York, NY, USA, 2008. ACM.

[189] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *WWW '08: Proceeding of the 17th International Conference on World Wide Web*, pages 915–924, New York, NY, USA, 2008. ACM.

[190] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 641–650, New York, NY, USA, 2010. ACM.

[191] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW'08: Proceedings of the 17th international conference on World Wide Web*, pages 695–704, New York, NY, USA, 2008. ACM.

[192] Francois Lesueur, Ludovic Mè, and Valèrie Viet Triem Tong. An efficient distributed pki for structured p2p networks. In *P2P'09: Proceedings of the 2009 Ninth International Conference on Peer-to-Peer Computing*, Seattle, WA, USA, 2009. IEEE Computer Society.

[193] Avner Levin and Patricia Sánchez Abril. Two notions of privacy online. *Vanderbilt Journal of Entertainment and Technology Law*, 11:1001–1051, 2009.

[194] Xin Li, Lei Guo, and Yihong Eric Zhao. Tag-based social interest discovery. In *WWW'08: Proceedings of the 17th International Conference on World Wide Web*, pages 675–684, New York, NY, USA, 2008. ACM.

[195] Jian Liang, Rakesh Kumar, Yongjian Xi, and Keith Ross. Pollution in p2p file sharing systems. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, pages 1174–1185, 2005.

[196] Jian Liang, Naoum Naoumov, and Keith W. Ross. The index poisoning attack in p2p file sharing systems. In *INFOCOM*, 2006.

[197] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, New York, NY, USA, 2003. ACM.

[198] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM'03: Proceedings of the 12th International Conference on Information and Knowledge Management*, pages 556–559, New York, NY, USA, 2003. ACM.

[199] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[200] David Liben-Nowell, Jasmine Novak, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–11628, 2005.

[201] F. Liljeros, C. R. Edling, L. A. N. Amaral, H. E. Stanley, and Y. Aaberg. The web of human sexual contacts. *Nature*, 411(6840):907–908, 2001.

[202] Dekang Lin. An information-theoretic definition of similarity. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 296–304. Morgan Kaufmann, 1998.

[203] Marek Lipczak and Evangelos Milios. The impact of resource title on tags in collaborative tagging systems. In *HT'10: Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 179–188, New York, NY, USA, 2010. ACM.

[204] Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, TAI '95, pages 88–, Washington, DC, USA, 1995. IEEE Computer Society.

[205] Xiaosong Lou and Kai Hwang. Prevention of index-poisoning DDoS attacks in peer-to-peer file-sharing networks. submitted to IEEE Trans. on Multimedia, Special Issue on Content Storage and Delivery in P2P Networks, 2006.

[206] Linyuan Lü, Jin Ci-Hang, and Tao Zhou. Effective and efficient similarity index for link prediction of complex networks. In *arXiv:0905.3558*, 2009.

[207] Linyuan Lü and Tao Zhou. Role of weak ties in link prediction of complex networks. In *CNIKM '09: Proceeding of the 1st ACM international workshop on Complex networks meet information and knowledge management*, pages 55–58, New York, NY, USA, 2009. ACM.

[208] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[209] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005.

[210] Matthew M. Lucas and Nikita Borisov. Flybynight: mitigating the privacy risks of social networking. In *WPES '08: Proceedings of the 7th ACM workshop on Privacy in the electronic society*, pages 1–8, New York, NY, USA, 2008. ACM.

[211] Henrik Lundgren, Richard Gold, Erik Nordström, and Mattias Wiggberg. A distributed instant messaging architecture based on the Pastry peer-to-peer routing substrate. In *SNCNW 2003, Swedish National Computer Networking Workshop, Stockholm*, 2003.

[212] Ben Lynn. *On the implementation of pairing-based cryptosystems.* PhD thesis, Stanford University, 2007.

[213] Leonardo Maccari, Matteo Rosi, Romano Fantacci, Luigi Chisci, Marco Milanesio, and Luca Maria Aiello. Avoiding eclipse attacks on Kad/Kademlia: an identity based approach. In *ICC 2009 Communication and Information Systems Security Symposium, to appear*, Dresden, Germany, 6 2009.

[214] D Malkhi, M Naor, and D Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. pages 183–192, 2002.

[215] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *WWW'09: Proceedings of the 18th International Conference on World Wide Web*, New York, NY, USA, 2009. ACM.

[216] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *Proc. of WWW '09*, 2009.

[217] B Markines and F Menczer. A scalable, collaborative similarity measure for social annotation systems. In *HT'09: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, New York, NY, USA, 2009. ACM.

[218] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Stum Gerd. Evaluating similarity measures for emergent semantics of social tagging. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 641–650, New York, NY, USA, 2009. ACM.

[219] Benjamin Markines, Heather Roinestad, and Filippo Menczer. Efficient assembly of social semantic networks. In *HT'08: Proceedings of the 19th ACM Conference on Hypertext and Hypermedia*, pages 149–156, New York, NY, USA, 2008. ACM.

[220] S. Maslov, K. Sneppen, and A. Zaliznyak. Detection of topological patterns in complex networks: correlation profile of the Internet. *Physica A*, 333:529–540, 2004.

[221] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *IPTPS'02: 1st International Workshop on Peer-to-Peer Systems*, pages 53–65, 2002.

[222] David Mazieres. Don't trust your file server. In *HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, page 113, Washington, DC, USA, 2001. IEEE Computer Society.

[223] Miller McPherson, Lynn S. Lovin, and James M. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[224] Peter Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):5–15, 2007.

[225] Marco Milanesio and Giancarlo Ruffo. Iterative key based routing for web services addressing and discovery. In *Proc. of AIMS '07*, pages 221–224, Berlin, Heidelberg, 2007. Springer-Verlag.

[226] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[227] David R. Millen and Jonathan Feinberg. Using Social Tagging to Improve Social Navigation. In *Workshop on the Social Navigation and Community based Adaptation Technologies*, 2006.

[228] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, October 2002.

[229] Alan Mislove, Hema Swetha Koppula, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Growth of the Flickr social network. In *WOSN'08: Proceedings of the first workshop on Online social networks*, pages 25–30, New York, NY, USA, 2008. ACM.

[230] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, New York, NY, USA, 2007. ACM.

[231] Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, Dan S. Wallach, Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Bezerra. POST: a secure, resilient, cooperative messaging system. In *HOTOS'03: Proceedings of the 9th conference on Hot Topics in Operating Systems*, pages 11–11, Berkeley, CA, USA, 2003. USENIX Association.

[232] Moddr. Web 2.0 suicide machine. `http://suicidemachine.org`, 2010.

[233] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Struct. Algorithms*, 6:161–179, 1995.

[234] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6:161–179, March 1995.

[235] Meredith Ringel Morris, Jaime Teevan, and Katrina Panovich. A comparison of information seeking using search engines and social networks. In *ICWSM'10: Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 23–26, 2010.

[236] Michele Mostarda, Davide Palmisano, Federico Zani, and Simone Tripodi. Towards an OpenID-based solution to the Social Network Interoperability problem. In *W3C Workshop on the Future of Social Networking*, 2009.

[237] Tamara Macushla Munzner. *Interactive visualization of large graphs and networks*. PhD thesis, Stanford, CA, USA, 2000. AAI9995264.

[238] Tsuyoshi Murata and Sakiko Moriyasu. Link prediction of social networks based on weighted proximity measures. In *WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 85–88, Washington, DC, USA, 2007. IEEE Computer Society.

[239] Naoum Naoumov and Keith Ross. Exploiting p2p systems for DDoS attacks. In *InfoScale '06: Proceedings of the 1st international conference on scalable information systems*, page 47, New York, NY, USA, 2006. ACM.

[240] Rammohan Narendula, Thanasis G. Papaioannou, and Karl Aberer. Privacy-aware and highly-available OSN profiles. In *COPS '10: the 6th International Workshop on Collaborative Peer-to-Peer Systems. In proceedings of WETICE 2010 : 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, to appear.*

[241] Mark E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, Jul 2001.

[242] Mark E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences of the United States of America*, 98(2):404–409, Jan 2001.

[243] Mark E. J. Newman. Assortative mixing in networks. *Physical Review Letter*, 89:208701, 2002.

[244] Mark E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67:026126, 2003.

[245] Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 69:066133, 2004.

[246] Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74, 2006.

[247] Mark E. J. Newman, Albert-László Barabasi, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, Princeton, NJ, USA, 2006.

[248] Mark E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E 69(026113), 2004.

[249] Mark E. J. Newman and Juyong Park. Why social networks are different from other types of networks. *Phys. Rev. E*, 68(3):036122, Sep 2003.

[250] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and ranking algorithms for event-based network data. *SIGKDD Explor. Newsl.*, 7(2):23–30, 2005.

[251] Kurt Opsahl. Facebook's eroding privacy policy: a timeline. `http://www.eff.org/deeplinks/2010/04/facebook-timeline`, April, 2010.

[252] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, 2010.

[253] Tim O'Reilly. What is web 2.0. design patterns and business models for the next generation of software. online, Sep 2005.

[254] Adam Ostrow. Social networking more popular than email. `http://mashable.com/2009/03/09/social-networking-more-popular-than-email`, March, 2009.

[255] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[256] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[257] Christopher R. Palmer, Phillip B. Gibbons, and Christos Faloutsos. Anf: a fast and scalable tool for data mining in massive graphs. In *KDD'02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90, New York, NY, USA, 2002. ACM.

[258] Eni Mustafaraj Panagiotis T. Metaxas and Dani Gayo-Avello. How (Not) to Predict Elections. In *SocialCom'11: Proceedings of the 3rd IEEE International Conference on Social Computing*, pages 165–171, October 2011.

[259] André Panisson. Visualization of egyptian revolution on twitter. http://slashdot.org/story/11/02/15/1544254, Feb 2011.

[260] R. Pastor-Satorras, A. Vázquez, and A. Vespignani. Dynamical and correlation properties of the Internet. *Physical Review Letters*, 87:258701+, 2001.

[261] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86:3200–3203, Apr 2001.

[262] Milen Pavlov and Ryutaro Ichise. Finding experts by link prediction in co-authorship networks. In *FEWS2007: Proceedings of the Workshop on Finding Experts on the Web with Semantics at ISWC/ASWC2007, Busan, South Korea*, November 2007.

[263] Alexandrin Popescul, Rin Popescul, and Lyle H. Ungar. Structural logistic regression for link analysis. In *Proceedings of the Second International Workshop on MultiRelational Data Mining*, 2003.

[264] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. TRIBLER: a social-based peer-to-peer system. *Concurrency and Computation*, 20(2):127–138, 2008.

[265] Christophe Prieur, Dominique Cardon, Jean-Samuel Beuscart, Nicolas Pissard, and Pascal Pons. The strength of weak cooperation: A case study on flickr. Technical Report arXiv:0802.2317v1, CoRR, 2008.

[266] Daniele Quercia and Licia Capra. Friendsensing: recommending friends using mobile phones. In *RecSys'09: Proceedings of the third ACM conference on Recommender systems*, pages 273–276, New York, NY, USA, 2009. ACM.

[267] Daniele Quercia, Jonathan Ellis, Licia Capra, and Jon Crowcroft. In the Mood for Being Influential on Twitter. In *SocialCom'11: Proceedings of the 3rd IEEE International Conference on Social Computing*, Boston, Massachusetts, USA, October 2010.

[268] Daniele Quercia, Neal Lathia, Francesco Calabrese, Giusy Di Lorenzo, and Jon Crowcroft. Recommending Social Events from Mobile Phone Location Data. In *ICDM'10: Proceedings of IEEE International Conference on Data Mining*, Dec 2010.

[269] Daniele Quercia, Giusy Di Lorenzo, Francesco Calabrese, and Carlo Ratti. Mobile phones and outdoor advertising: Measurable advertising. *IEEE Pervasive Computing*, 10:28–36, 2011.

[270] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, January 1989.

[271] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Science of the United States of America*, 101-9:2658–2663, 2004.

[272] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *KDD'05: 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, New York, NY, USA, 2005. ACM.

[273] Vijay V. Raghavan and Hayri Sever. On the reuse of past optimal queries. In *SIGIR '95*, 1995.

[274] Leena Rao. Larry page on google+: Over 10 million users, 1 billion items being shared per day, Lug 2011.

[275] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Detecting and tracking political abuse in social media. In *Proc. 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.

[276] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. Truthy: mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 249–252, New York, NY, USA, 2011. ACM.

[277] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[278] Matthew J. Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 783–790, New York, NY, USA, 2007. ACM.

[279] David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, New York, NY, USA, 2006. ACM.

[280] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the First International Conference on Peer-to-Peer Computing*, P2P '01, pages 99–, Washington, DC, USA, 2001. IEEE Computer Society.

[281] Juan J. Rodriguez, Ludmila I. Kuncheva, and Carlos J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[282] Daniel Romero and Jon Kleinberg. The directed closure process in hybrid social-information networks, with an analysis of link formation on twitter. In *AIII '10 : Proceedings of 4th International AAAI Conference on Weblogs and Social Media*, 2010.

[283] Keith Ross, Jian Liang, and Naoum Naoumov. Efficient blacklisting and pollution-level estimation in p2p file-sharing systems. In *Proc. of Asian Internet Engineering Conference*, 2005.

[284] Hosam Rowaihy, William Enck, Patrick McDaniel, and Thomas La Porta. Limiting sybil attacks in structured peer-to-peer networks. Technical Report NAS-TR-0017-2005, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, 2005.

[285] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *LNCS*, 2218:329–351, 2001.

[286] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. *ACM SIGOPS Operating Systems Review*, 35(5):188–201, 2001.

[287] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. Scribe: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43. Springer, 2001.

[288] Giancarlo Ruffo and Rossano Schifanella. A peer-to-peer recommender system based on spontaneous affinities. *ACM Transactions on Internetet Technology*, 9:4:1–4:34, February 2009.

[289] Sunam Ryu, Kevin Butler, Patrick Traynor, and Patrick McDaniel. Leveraging identity-based cryptography for node id assignment in structured p2p systems. In *Proc. of AINAW '07*, pages 519–524, Washington, DC, USA, 2007. IEEE Computer Society.

[290] Mrinmaya Sachan and Ryutaro Ichise. Using semantic information to improve link prediction results in networked datasets. *International Journal of Engineering and Technology*, 2(4):334–339.

[291] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW'10: Proceedings of the 19th international conference on World wide web*, pages 851–860, New York, NY, USA, 2010. ACM.

[292] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[293] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[294] Elizeu Santos-Neto, David Condon, Nazareno Andrade, Adriana Iamnitchi, and Matei Ripeanu. Individual and social behavior in tagging systems. In Ciro Cattuto, Giancarlo Ruffo, and Filippo Menczer, editors, *HT'09: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, pages 183–192, New York, NY, USA, 2009. ACM.

[295] Elizeu Santos-Neto, David Condon, Nazareno Andrade, Adriana Iamnitchi, and Matei Ripeanu. Individual and social behavior in tagging systems. In *HT'09: Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 183–192, New York, NY, USA, 2009. ACM.

[296] Luís Sarmento, Alexander Kehlenbeck, Eugénio Oliveira, and Lyle Ungar. Efficient clustering of web-derived data sets. In *MLDM '09: 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 398–412, Berlin, Heidelberg, 2009. Springer-Verlag.

[297] Salvatore Scellato. Beyond the social web: the geo-social revolution. *SIGWEB Newsletter*, pages 5:1–5:5, September 2011.

[298] Rossano Schifanella, Luca Maria Aiello, and Giancarlo Ruffo. Tagging relations to achieve complex search goals. In *NWeSP'11: Proceedings of the 7th IEEE International Conference on Next Generation Web Services Practices*, pages 308–313. IEEE Press, 2011.

[299] Rossano Schifanella, Alain Barrat, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. Folks in folksonomies: social link prediction from shared metadata. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 271–280, New York, NY, USA, 2010. ACM.

[300] Rossano Schifanella, André Panisson, Cristina Gena, and Giancarlo Ruffo. Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *RecSys'08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 27–34, New York, NY, USA, 2008. ACM.

[301] M. A. Serrano and M. Boguñá. Tuning clustering in random networks with arbitrary degree distributions. *Phys. Rev. E*, 72:036133, 2005.

[302] Amre Shakimov, Alexander Varshavsky, Landon P. Cox, and Ramón Cáceres. Privacy, cost, and availability tradeoffs in decentralized osns. In *WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks*, pages 13–18, New York, NY, USA, 2009. ACM.

[303] C.R. Shalizi and A.C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *preprint*, page arxiv:1004.4704, 2010.

[304] Adi Shamir. Identity based cryptosystems and signature schemes. In *CRYPTO 84: Proceedings of Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York.

[305] Clay Shirky. What is p2p... and what isn't. O'Reilly Open P2P, http://openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html, 11 2000.

[306] Xiance Si, Edward Y. Chang, Zoltán Gyöngyi, and Maosong Sun. Confucius and its intelligent disciples: Integrating social with search. In *VLDB'10: Proceedings of the 36th International Conference on Very Large Data Bases*, pages 1505–1516, 2010.

[307] Atul Singh, Tsuen Wan Ngan, Peter Druschel, and Dan Wallach. Eclipse attacks on overlays: Threats and defenses. In *InfoCom '06: 25th Conference on Computer Communications*. IEEE Computer Society, April 2006.

[308] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 261–269, London, UK, 2002. Springer-Verlag.

[309] Sketchology. What is information overload? online, Jan 2010.

[310] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. Scalable proximity estimation and link prediction in online social networks. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 322–335, New York, NY, USA, 2009. ACM.

[311] Wei Song, Yu Zhang, Ting Liu, and Sheng Li. Bridging topic modeling and personalized search. In *COLING'10: 23rd International Conference on Computational Linguistics: Posters*, pages 1167–1175, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[312] Anna C. Squicciarini, Federica Paci, Elisa Bertino, Alberto Trombetta, and Stefano Braghin. Group-based negotiations in p2p systems. *IEEE Transactions on Parallel and Distributed Systems*, 99(PrePrints), 2010.

[313] Mudhakar Srivatsa, Li Xiong, and Ling Liu. TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. In *WWW '05: 14th international conference on World Wide Web*, pages 422–431, 2005.

[314] M. Stäger, P. Lukowicz, and G. Troster. Dealing with class skew in context recognition. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops*, page 58, 2006.

[315] Moritz Steiner, Ernst W. Biersack, and Taoufik Ennajjary. Actively monitoring peers in kad, 2008.

[316] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Exploiting KAD: possible uses and misuses. *SIGCOMM Computer Communications Review*, 37(5):65–70, 2007.

[317] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. A global view of KAD. In *IMC'07: Proceedings of the 7th ACM SIGCOMM*, pages 117–122, New York, NY, USA, 2007. ACM.

[318] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[319] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[320] Lubomira Stoilova, Todd Holloway, Ben Markines, Ana G. Maguitman, and Filippo Menczer. Givealink: mining a semantic network of bookmarks for web search and recommendation. In *LinkKDD'05: Proceedings of the 3rd international workshop on Link discovery*, pages 66–73, New York, NY, USA, 2005. ACM.

[321] Thorsten Strufe. Profile popularity in a business-oriented online social network. In *Proceedings of the 3rd Workshop on Social Network Systems*, SNS '10, pages 2:1–2:6, New York, NY, USA, 2010. ACM.

[322] Xin Sun, Ruben Torres, and Sanjay Rao. Ddos attacks by subverting membership management in p2p systems. In *Proceedings of the 2007 3rd IEEE Workshop on Secure Network Protocols*, pages 1–6, Washington, DC, USA, 2007. IEEE Computer Society.

[323] Michael Szell, Renaud Lambiotte, and Stefan Thurner. Multirelational organization of large-scale social networks in an onli ne world. *Proceedings of the National Academy of Sciences*, 107(31):13636–13641, 2010.

[324] Benjamin Taskar, Ming Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *NIPS'03: Neural Information Processing Systems Conference*, Vancouver, Canada, December 2003.

[325] Alvin Toffler. *Future Shock*. Random House, 1970.

[326] Amin Tootoonchian, Kiran Kumar Gollu, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: social access control for web 2.0. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 43–48, New York, NY, USA, 2008. ACM.

[327] A. Tumasjan, T.O. Sprenger, P.G. Sandner, and I.M. Welpe. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 178–185, 2010.

[328] Tomasz Tylenda, Ralitsa Angelova, and Srikanta Bedathur. Towards time-aware link prediction in evolving social networks. In *SNA-KDD '09: Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, pages 1–10, New York, NY, USA, 2009. ACM.

[329] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The Anatomy of the Facebook Social Graph. Nov 2011.

[330] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of dht security techniques. *ACM Computing Surveys (CSUR)*, 43:8:1–8:49, February 2011.

[331] Roelof van Zwol. Flickr: Who is looking? In *WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 184–190, Washington, DC, USA, 2007. IEEE Computer Society.

[332] A. Vázquez, R. Pastor-Satorras, and A. Vespignani. Large-scale topological and dynamical properties of the Internet. *Physical Review E*, 65:066130+, 2002.

[333] Ashok Veilumuthu and Parthasarathy Ramachandran. Intent based clustering of search engine query log. In *CASE'09: 5th IEEE international conference on Automation science and engineering*, pages 647–652, Piscataway, NJ, USA, 2009. IEEE.

[334] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 37–42, New York, NY, USA, 2009. ACM.

[335] Max Völkel, Markus Krötzsch, Denny Vrandecic, Heiko Haller, and Rudi Studer. Semantic wikipedia. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 585–594, New York, NY, USA, 2006. ACM.

[336] Luis von Ahn, Benjamin Maurer, Colin Mcmillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895):1465–1468, 2008.

[337] Melisssa Wall and Sahar E L Zahed. "I'll Be Waiting for You Guys": A YouTube Call to Action in the Egyptian Revolution. *Journal of Communication*, 5:1333–1343, 2011.

[338] Honghao Wang, Yingwu Zhu, and Yiming Hu. An efficient and secure peer-to-peer overlay network. In *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks*, pages 764–771, Washington, DC, USA, 2005. IEEE Computer Society.

[339] P. Wang, I. Osipkov, N. Hopper, and Yongdae Kim. Myrmic: Secure and robust dht routing. Technical report, DTC Research, 2006.

[340] Stanley Wasserman and Katerine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[341] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, 1998.

[342] Ji-Rong Wen, Jian-Yun Nie, and Zhang Hong-Jiang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20:59–81, January 2002.

[343] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM.

[344] Lilian Weng, Rossano Schifanella, and Filippo Menczer. The chain model for social tagging game design. In *FDG'11: Proceedings of the 6th International Conference on Foundations of Digital Games*, 2011.

[345] Lilian Weng, Rossano Schifanella, and Filippo Menczer. Design of social games for collecting reliable semantic annotations. In *CGAMES'11, 16th International Conference on Computer Games*, 2011.

[346] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, EuroSys '09, pages 205–218, New York, NY, USA, 2009. ACM.

[347] Ian H. Witten and Eibe Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, 1999.

[348] Fang Wu and Bernardo A. Huberman. Finding communities in linear time: A physics approach. *European Physical Journal B*, 38:331–338, 2004.

[349] Jennifer Xu and Hsinchun Chen. The topology of dark networks. *Commun. ACM*, 51:58–65, October 2008.

[350] Yahoo! Tag Explorer. `http://tagexplorer.sandbox.yahoo.com`, 2008.

[351] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. *Data Engineering, International Conference on*, 0:49, 2003.

[352] Jaewon Yang and Jure Leskovec. Modeling information diffusion in implicit networks. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 599–608, Washington, DC, USA, 2010. IEEE Computer Society.

[353] Jerry Ye, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng. Stochastic gradient boosted distributed decision trees. In *CIKM'09: 18th ACM conference on Information and knowledge management*, pages 2061–2064, New York, NY, USA, 2009. ACM.

[354] Shaozhi Ye and S. Felix Wu. Measuring message propagation and social influence on twitter.com. In *Proceedings of the Second international conference on Social informatics*, SocInfo'10, pages 216–231, Berlin, Heidelberg, 2010. Springer-Verlag.

[355] Ching Man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee. Decentralization: The Future of Online Social Networking. In *W3C Workshop on the Future of Social Networking*, 2009.

[356] Jeonghee Yi and Farzin Maghoul. Query clustering using click-through graph. In *WWW'09: 18th international conference on World wide web*, pages 1055–1056, New York, NY, USA, 2009. ACM.

[357] Bin Yu and P. Munindar Singh. Incentive mechanisms for peer-to-peer systems. In *Second International Workshop on Agents and Peer-to-Peer Computing*, pages 77–88, 2003.

[358] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17, 2008.

[359] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473.

[360] Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. Learning to cluster web search results. In *SIGIR'04: 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA, 2004. ACM.

[361] B Y Zhao, L Huang, J Stribling, S C Rhea, A D Joseph, and J D Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, (22), 2003.

[362] Elena Zheleva, Lise Getoor, Jennifer Golbeck, and Ugur Kuter. Using friendship ties and family circles for link prediction (poster paper). In *2nd SNA-KDD Workshop on Social Network Mining and Analysis*, Las Vegas, NV, USA, August 2008.

[363] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *European Physical Journal B*, 71(4):623–630, 2009.