Distributing Social Applications

Vincent Leroy

Thèse

THÈSE INSA Rennes
sous le sceau de l'Université Européenne de Bretagne
pour obtenir le titre de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Informatique

présentée par
Vincent LEROY
ECOLE DOCTORALE : MATISSE
LABORATOIRE : IRISA UMR 6074
Institut de recherche en informatique et systèmes aléatoires

Décentralisation des applications sociales

Distributing social applications

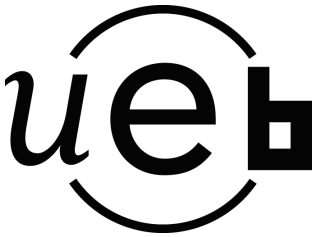
Thèse soutenue le 10/12/2010
devant le jury composé de :

Pascal FELBER
Pr à Université de Neuchâtel / président
Maarten VAN STEEN
Pr à Université de Vrije / rapporteur
Peter TRIANTAFILLOU
Pr à Université de Patras / rapporteur
Sihem AMER-YAHIA
Senior Researcher à Yahoo! Research / examinateur
Anne-Marie KERMARREC
DR à INRIA Rennes / directeur de thèse
Marin BERTIER
MdC à INSA de Rennes / co-directeur de thèse

Décentralisation des applications sociales

Distributing social applications

Vincent LEROY



UMR IRISA

En partenariat avec
In partnership with

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

centre de recherche RENNES - BRETAGNE ATLANTIQUE

ACKNOWLEDGEMENTS

todo

CONTENTS

CHAPTER 1

INTRODUCTION

The so-called Web 2.0 revolution has fundamentally changed the way people interact with the Internet. The Web has turned from a read-only infrastructure to a collaborative platform. The users are now active participants, and contribute to the content of the websites by expressing their opinions and sharing information.

Social networks Social network websites, such as Facebook, were originally created to keep track of friends, and share information and pictures with them. On these platforms, users can publish information to explicitly designated friends. It is also possible, for example, to associate a friend with a picture on which she appears. Facebook has proved to be a very efficient mean to quickly reach thousands of people and organize events.

Social networks have developed tools that enable other websites to access to the user's profile and personalize the Web content depending on her friends, or the city she lives in for instance. Hence, it is possible for a user to recommend a website to her friends. The website can also detect that two friends are interested in the same information which gives them an opportunity to discuss about it. As the content of the Web keeps on growing, leveraging social networks can be an effective way to provide the users with a more interesting and personalized experience.

Social information While social networks were designed to connect people with real life friends, many modern websites have been created to leverage social information. They do not rely on an explicit social network, but let the users express their opinions and preferences. For example, LastFM collects what users listen to, while Delicious is a collaborative bookmarking platform. These websites aggregate the information provided by the users to extract knowledge such as trends in music communities, or hot topics in the Internet. In return, the users benefit from personalized services based on the preferences they expressed. The website can compute similarities between the interests of the user and recommend new content, through collaborative filtering for instance.

Several websites also rely on the users to annotate the content they propose. This phenomenon is known as *crowdsourcing*. Youtube and Flickr are websites in which users can publish videos and

photos respectively. These contents are very difficult to analyze, as opposed to text, and thus difficult to integrate in a search engine. The users have the possibility to annotate the content using freely chosen keywords, also called *tags*, either to facilitate the diffusion of their own content, or to enhance the efficiency of the website they use. This increases the efficiency of search engines, as the tags can be used to index the content. Monitoring the activities of the users also helps discovering new links between items as well as identifying popular content.

Privacy Social systems are crucial to enhance the Web experience of the users. Nevertheless, they also raise serious privacy problems. Indeed, to benefit from these services, users have to publish personal information. The complexity of the interactions makes it very difficult to precisely know which information is recorded and who can access it. Social networks propose configuration options to control who can access the data, but the users are often not aware of how exposed they really are.[1] To the best of our knowledge, websites based on social information always associate the contributions of a user with her identity, which enable anyone to track her activities and interests. In 2009, the Facebook terms of service even included a *perpetual ownership* claim over the user's data.

The privacy problem mainly arises from the mismatch between the user's need to protect their private life, and the website's need for business opportunities. It is well known that friends influence each other, thus companies leverage social networks to increase their sales. For instance, Facebook implemented a functionality which automatically tells a user's friends about what she buys on the Amazon Web store. This behavior clearly threatens the privacy of the users, and Facebook had to remove it due to the user's reactions. Similarly, social information is often sold to other companies for personalized advertising purpose. As a consequence, the users may become reluctant to share their personal data. This is a huge problem, as the efficiency of social platforms depends of the number of users participating. Hence, the websites have to find a balance between the amount of private information they publicize and the quality of the services they propose.

A personalized Web experience leverages the user's private data, either through a social networks or through their social information. While users can greatly benefit from these opportunities, they may feel threatened by the private companies' policies and reject these platforms.

Cost and scalability As the number of social services users keeps on growing, it becomes increasingly difficult for companies to scale with the demand. Facebook claims over 500 millions of active users, while LastFM reports 40 millions accounts. The computation required by personalized services as well as the storage of the user's personal information necessitate large dedicated data centers. This infrastructure is extremely costly, both in terms of machines and in terms of energy. The price of energy and the proximity of energy production sites has even become one of the main placement criteria for data centers. Data centers are often constituted of many regular machines, and new computation paradigms have been developed to leverage their power. Indeed, a data center can no longer be seen as one extremely powerful computer, it is now a distributed architecture [26]. Hence, the programmers have to deal with problems related to the distribution of data, its replication and fault tolerance, in order to increase the reliability of the infrastructure.

Nevertheless, computing personalized recommendations can still be extremely costly. In [4], Amer-Yahia *et al.* explore the trade-offs between processing time and storage capacity for implementing

[1] Even the founder of Facebook had some private pictures made public at the occasion of a privacy configuration changes.

and exact personalized recommendation in Delicious. They show that, even when only a subset of the Delicious data is selected, this problem remains very challenging. In order to scale with the number of users, data centers often aggregate the users into interest communities and recommendations are computed for the whole community. This results in an approximation and a degradation of the service proposed to the user. If more processing power was available, more sophisticated and costly algorithms could further enhance the Web experience of the users.

Processing social information is extremely costly and requires a dedicated infrastructure. The degree of personalization proposed to the users is limited by the power of data centers.

Peer-to-peer (P2P) The thesis we defend is that social networks and social information applications should be deployed over P2P networks. As social applications are user centered, it is very easy to conceive them as distributed systems: each user and her information represents one machine in the P2P network.

P2P networks are decentralized architectures in which all the participants contribute to the operation of a service. This model is often opposed to the client/server one, in which a central machine, the server, has to process all the demands of the clients. In P2P systems, all the clients also execute a share of the server's work. Hence, P2P architectures are known for their scalability, since each client joining the network also increases the resources available to provide the service. The decentralization of the architecture increases its reliability, as the service is replicated on many machines distributed all around the world.

We believe that such an architecture has two main advantages. First, each user controls her information and does not have to store it on a central server that belongs to a private company. This clearly is a huge advantage in terms of privacy. The P2P network is a neutral platform and does not belong to anyone. Second, the scalability of P2P networks in terms of computing and storage offers support for better personalized services. Data centers are already evolving towards distributed architectures. We propose to push the decentralization even further by deploying the applications directly on user's machines. As each user contributes to the platform with her own machine, a P2P social platform does not require a costly dedicated architecture. Furthermore, as more processing power would be available, the P2P architecture could offer new services that go beyond the ones available in centralized existing platforms.

Nevertheless, building social systems in a distributed context also comes with many challenges. As each user manages her own information, the system has to be able to efficiently locate interesting information stored by other users, without generating too many network communications. The architecture needs to be efficient in order to provide functionalities with a quality of service comparable to dedicated data centers. While the scalability of P2P systems offers opportunities for more advanced features, the existing ones should not be impaired due to the distribution of the information. Many different P2P architectures have been proposed to manage distributed information. We rely on gossip algorithms to weakly structure the P2P network and find the relevant information.

Contributions of this thesis

In this thesis, we consider the design of social networks and social information services in the context of P2P networks. We believe that distributing social services over P2P platforms in which users control their own data is an important first step towards better user privacy. Yet, P2P networks do

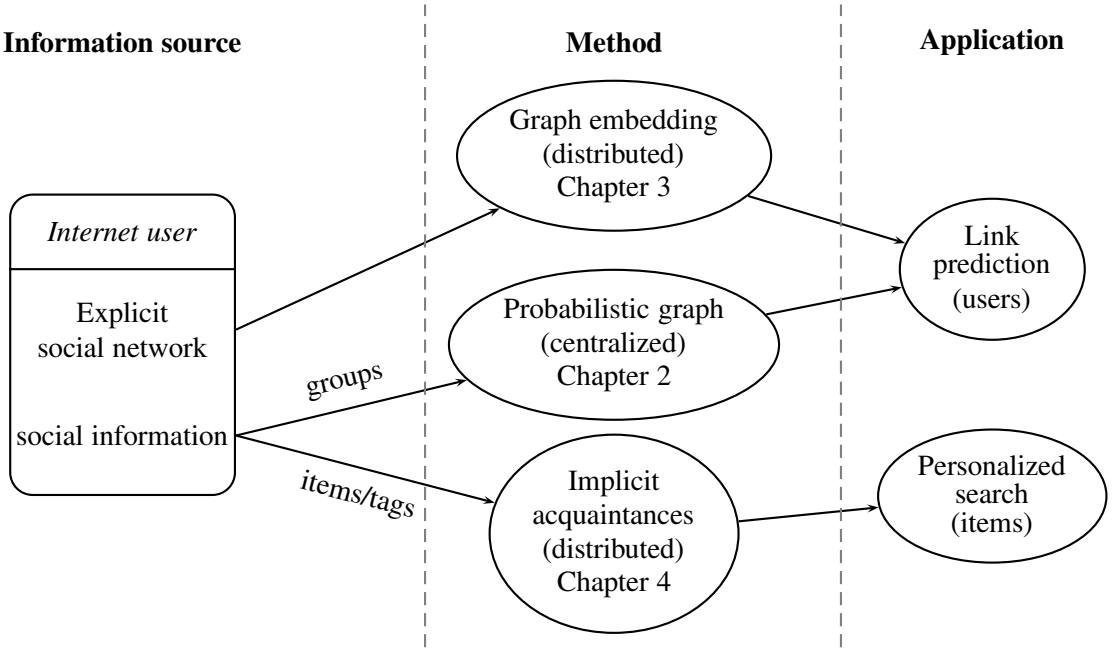


Figure 1.1: Summary of the contributions

not provide a full solution to this problem, and have to be associated with techniques such as onion routing [80] to offer a better protection to the users. The goal of this thesis is to propose scalable, decentralized algorithms, that leverage social networks and social information. Thus, we will mostly be interested on how to efficiently locate and use relevant information. We will mention some privacy protecting mechanisms adapted to the algorithms we propose, but do not pretend to completely solve this problem.

The fist contribution we present shows how social information can be used in a centralized system to infer a private social network. This highlights the privacy problems raised by centralized solutions, and highlights the need for P2P alternatives. The second contribution considers the problem of predicting new social neighbors in a decentralized social network. Finally, the third contribution proposes a P2P system for personalized Web search through social information. Figure 1.1 gives an overview of these three contributions and shows how they are linked to each other.

Link prediction from social information

The link prediction problem consists in predicting new edges in a social graph. Typically, this can be used to recommend new friends to the users of a social network. Most of the link prediction algorithms rely only on the knowledge of the social network to predict new links. In Chapter 2, we show that it is also possible to accurately predict a hidden social network by leveraging social information. We call this problem *cold start link prediction*.

We propose a two-phase method to solve this problem and evaluate it on a real dataset (Flickr). In the first phase, we leverage any social information available about the users to generate a *probabilistic graph*. This probabilistic graph is a prediction of the social network, based purely on the similarity of the social information of pairs of vertices. In our evaluation, we rely on the group memberships of

the users to estimate their probability to be connected in the social graph. The second phase consists in refining this probabilistic graph by taking into account the transitivity of the social relations. Therefore, we consider classic link prediction techniques that rely on the paths in the social graphs. We adapt these algorithms to probabilistic graphs and compare our approach with non probabilistic techniques.

Although group memberships of the users are weakly correlated with the social network, our results show that our method is able to identify quite accurately social links between users. The approach we propose is generic: any available social information can be used in the first phase. Thus, more social information will lead to even more accurate link prediction. We believe that this work demonstrates the difficulty of social information anonymization and the need for decentralized social platforms. The social network of the users is often considered as a sensitive data. In our case, it is kept secret by the user. Nevertheless, our approach is able to infer the social network from a little amount of social information. Several approaches have been proposed to anonymize social information, and make it difficult for an attacker to infer sensitive hidden data. However, many other algorithms have been created to de-anonymize this information. It is in fact very difficult to evaluate the degree of anonymity of a dataset.

Distributed link prediction for social networks

In Chapter 3, we address the problem of the distributed computation of link prediction. As link prediction is a crucial functionality for social networks, finding a fully distributed solution to that problem is crucial for the distribution of social systems. Each vertex of the social network participates to a P2P system, and is responsible for its social links. Contrary to the *cold start link prediction* problem, the social network is not hidden, as users do not have to trust a third party to store it. Nevertheless the knowledge of the social network is partial: each vertex knows who its social neighbors are, but does not now the social neighbors of the other vertices.

Most of the link prediction algorithms were designed for centralized systems, therefore they assume that the whole social graph is available. In our case, navigating the full social graph is a very costly operation since it requires network communications for each edge transition. Furthermore, for privacy reasons, we want to avoid as much as possible sharing the lists of social neighbors between users. We propose SoCS (Social Coordinate Systems), a distributed algorithm for link prediction based on a *distributed embedding* of the social graph.

In SoCS, each vertex computes its social coordinates in a social space. These coordinates can then be used to easily evaluate the social distance between two vertices, without having to navigate the graph. SoCS is inspired from force-based graph drawing algorithms. In these algorithms, the graph simulates a physical systems, with attractions on the edges and repulsions between all pairs of vertices. SoCS performs a distributed force-based embedding for the social graph. However, contrary to graph drawing algorithms, SoCS can use more than 3 dimensions. At each cycle, the SoCS vertices compute the sum of the forces that are applied to them and adjust their position accordingly. The challenge, in our case, comes from the computation of the repulsion forces. A straightforward implementation would generate communications between all pairs of vertices, which would clearly not be scalable. SoCS relies on gossip protocols to approximate these repulsions by only considering the closest vertices in the social space.

We experiment SoCS on real and synthetic social graphs and compare the results against other existing algorithms. Our results show that SoCS obtains good results in link prediction. Furthermore,

we observe that the approximations performed by SOCS because of its distributed nature improve the efficiency of the link prediction. Finally, we compare different force models and show that the state-of-the-art force model for communities representation is not necessarily the best for link prediction.

Distributed personalized Web search

Many modern websites let the users express their opinions by describing *items* with freely chosen *tags*. This information generates a *folksonomy*. Navigating in a folksonomy is a difficult task as the information is unstructured. Users can express the same idea through different tags, and a given tag may not have the same meaning for different users. Additionally, social information exposes the interests of the users and threatens their privacy. In Chapter 4, we present the GOSSPLE [50] Multi-Interest Network (GMIN), a distributed system designed to personalize the Web experience. We illustrate its efficiency through a personalized query expansion mechanism (GQE), designed to facilitate search in a folksonomy.

Several centralized algorithms for personalized search have been proposed. However, they experience scalability problems. The amount of data to store and the processing power required to manage it does not enable centralized systems to fully personalize the search experience. In some solutions, the users are clustered into interest communities, which approximates the personalization. An other possibility is to only personalize a small part of the search process. GMIN relies on P2P system, so each user contributes to the system by sharing her storage and processing power. Thus, GMIN's architecture is inherently more scalable. Furthermore, in GMIN provides anonymity to the users. Each user stores her own profile and receives personalized information based on its content, but a user is never associated with her profile. This would not be possible in a centralized system, as the user would have to trust a third party with her data.

GMIN provides each user with a selection of neighbors that are interested in the same items. This selection is performed through gossip algorithms, and takes care to cover all the interests of the user, even the minor ones. These neighbors are then used to extract proximity between tags and discover new tags. When the user performs a search, GQE uses GMIN to automatically transforms her request. Tags which, according to the user and her neighbors, are similar, are added to the query, and weighted through an algorithm inspired from PageRank.

We experiment our system on real folksonomy data (Delicious and CiteULike). Our results shows that GQE improves the search experience of the user, both in terms of recall and precision.

This thesis only presents the decentralization of the query expansion mechanism. The reader may refer to [7] for a description of a decentralized search process.

CHAPTER 2

LINK PREDICTION FROM SOCIAL INFORMATION

Abstract

In this chapter, we evaluate the possibility of predicting a hidden social network using the social information of the users. We refer to this problem as *cold start link prediction*. We propose a two-phase approach to solve it. First, we build a probabilistic social graph using the social information about the users. Then, in the second phase, we adapt classic link prediction algorithms to the probabilistic graph and refine the prediction. We evaluate our method on a real dataset taken from Flickr, an photo sharing community. We use the group memberships of the users as social information. Our results show that our protocol accurately predicts some of the links in the social network. Additional social information would enable an even more precise prediction and seriously threaten the privacy of the users. We believe that this work highlights the need for a decentralized architecture for social systems.

This work, presented at the KDD 2010 conference [57], was done in collaboration with Berkant Barla Cambazoglu and Francesco Bonchi during internship at Yahoo! Research Barcelona. This internship was funded by the Collège Doctoral International (CDI) and the Université Européenne de Bretagne (UEB).

2.1 Introduction

Context

Social networks have become very popular platforms for people to exchange information with their friends. As previously mentioned, social networks may raise privacy problems, as they contain private information. This is typically why the access to the list of friends of a user is often restricted. However, many websites have access to social information about Internet users, such as their interests or logs of their Web activities. In this chapter, our goal is to predict a social network solely using the social information of its users. We show that any website having a sufficient amount of social information about the users can accurately predict social links among them.

Link prediction, introduced by Liben-Nowell and Kleinberg [59], consists in predicting new links in an existing and available social network. They state that the link prediction problem evaluates to what extent the evolution of a social network can be modeled using features intrinsic to the network itself. Indeed, all the features presented in [59] are based on the link structure of the network. In this chapter, we tackle the problem of *cold start link prediction*. We aim at predicting links within a social graph, but we aim at doing so without any, even partial, knowledge of the existing link structure: the initial social network is empty. However, we have access to some auxiliary social information about the users. This is the case if, for example, the users keep the social network secret for privacy reasons, but some of their activities, like purchases on an Internet store, can be monitored. This setup also represents the information available to a community website that does not support any explicit social network. In that case, the activities of the users can be used to recommend possible friends to users and speed up the expansion of the social network. The question tackled herein is the following: *is it possible to infer the social network (to an acceptable level of accuracy), just using the auxiliary social information?*

In this context, we propose a two-phase method based on the *bootstrap probabilistic graph* for cold start link prediction. In the first phase, based on some limited information (potentially, weakly correlated with the link structure of the network), the method predicts the existence of links. The output of this phase is a probabilistic graph, i.e., a graph where each edge is labeled with a probability representing the confidence of the prediction, or in other terms, the uncertainty of the existence of a link. The second phase takes as input the probabilistic graph and refines it by adopting graph-theoretic measures. These measures are similar to the algorithms used in the classical link prediction setting, but in our case, the input graph is probabilistic and hence the traditional measures must be adapted to deal with this case [71].

We apply our method to a large data collection obtained from Flickr[1], a popular online community for photo sharing. We keep the existing social network (represented as a directed graph) as the ground truth for the link structure that we aim to predict. As the available auxiliary information, we use users' memberships in interest groups.

Three observations are note-worthy. First, the cold start link prediction problem is intrinsically a very difficult binary prediction problem due to the skewness of the target variable. In fact, assuming that the edges in the social network are directed, given n vertices, we have a universe of $n(n-1)$ possible links, of which only a very small fraction exists in the ground truth. In our data, only 0.07% of all possible links are present in the ground truth.

[1] `http://www.flickr.com`

The second observation is that we apply our method starting with little information, which provides a very small coverage of existing links. Indeed, in Flickr, interest group membership[2] is a very weak predictor of links, as a group gathers people interested in photos regarding a specific subject or technique (e.g., *"Nikon Selfportrait"*, *"HDR Panoramas"*, *"Cat and Dog: not Cat or Dog"*), and they are not usually groups of friends or small communities (as it is more often the case in Facebook groups). More precisely, in our data, considering only the users who belong to at least one group, we have approximately 28M links, of which only 1.9% (approximately 550k links) are among two users that share at least one common group. We deliberately select weakly correlated information in order to highlight the benefits of our approach. Despite the difficulty of our prediction task, our two-phase method based on the bootstrap probabilistic graph can achieve good prediction performance.

Finally, it is very important to note that, even though the first phase features used herein target the group membership information, we propose a general framework which can be applied to any input information. In some cases, the available auxiliary information could be much more predictive than the one used in our experiments or more information might be available. If this is the case, all available information should be used in order to bootstrap the probabilistic graph as accurately as possible.

Contributions

We introduce cold start link prediction as the problem of predicting the link structure of a social network assuming unavailability of an initial network and using other available social information (in this case, group memberships). Our work has privacy and security implications as it sheds light on to what extent a social network can be reconstructed and how resilient an anonymization solution is to link prediction attacks.

We propose a two-step method based on the bootstrap probabilistic graph as a feasible solution to the cold start link prediction problem. We apply our method to predict the link structure of the Flickr social network by using only the interest group membership information. As discussed previously, group membership is a weak predictor for friendship and hence suits well to our purpose of starting the social network with little available information.

We assess the predictive power of various features based on group membership (phase 1), such as the time a user joins a group and the size of the group. The features we use herein can be considered to be somewhat general and are applicable to other problem instances as long as the available information can be mapped to a group structure. For instance, a thread on a discussion board can be mapped to a group, and the same features can be used for predicting friendship between the users of a bulletin board service.

We adapt various graph-theoretic measures to deal with probabilistic graphs (phase 2). Our experiments show that our approach, based on a probabilistic graph, clearly outperforms traditional approaches that rely on a deterministic graph whose edges have been pruned to remove the ones with low probability.

Roadmap

In Section 2.2, we present the formal definition of the problem and the proposed two-phase method. In Section 2.3, we describe the data we use for assessing our method, which is then developed in

[2] `http://www.flickr.com/groups`

Section 2.4 (phase 1) and Section 2.5 (phase 2). We review the existing work in Section 2.6, and we conclude this chapter in Section 2.7.

2.2 Problem and Method

2.2.1 Problem

We are given a set $\mathcal{V}$ of users and a multiset $\mathcal{H}$ of groups of users. We denote the set of groups to which a user u belongs to as her *membership set*:

$$m(u) = \{h \in \mathcal{H} | \, u \in h, u \subseteq \mathcal{V}\} \ .$$

Our task is to reconstruct the links of a social graph $\mathcal{G}_s = (\mathcal{V}, \mathcal{E}_s)$ where the vertices are the users and the edges $\mathcal{E}_s \subseteq \mathcal{V} \times \mathcal{V}$ represent a one-way relation between two users. Reconstructing the social network $\mathcal{G}_s$ means to predict which of the links in $\mathcal{V} \times \mathcal{V}$ actually exist in $\mathcal{E}_s$, or in other terms, to build a function $f : \mathcal{V} \times \mathcal{V} \to \{0, 1\}$.

2.2.2 Method

We propose a two-phase method for cold start link prediction. During the first phase, we predict the existence of links based only on the group membership information. The output of the first phase is the *bootstrap probabilistic graph*, i.e., a directed probabilistic graph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p, p)$, where $\mathcal{E}_p \subseteq \mathcal{V} \times \mathcal{V}$, and every link $(u, v) \in \mathcal{E}_p$ is labeled with a probability $p(u, v) > 0$ representing the confidence (or uncertainty) about the link's existence, i.e., $p : \mathcal{V} \times \mathcal{V} \to [0, 1]$.

In particular, after the first phase, we have $p(u, v) = 0$ and $p(v, u) = 0$ for every user pair (u, v), where $m(u) \cap m(v) = \emptyset$. This is because if two users have no groups in common, a prediction cannot be made about the existence of a link between them. Moreover, we have $p(u, v) > 0$ for every user pair (u, v) such that $m(u) \cap m(v) \neq \emptyset$ (this will also hold for the reverse arc (v, u)). Links with null probabilities do not exist in $\mathcal{G}_p$.

The second phase takes as input the bootstrap probabilistic graph $\mathcal{G}_p$, and it refines the probability distribution p into a new weight distribution w. This phase revisits the graph features traditionally used for link prediction and adapts them to probabilistic graphs. Therefore, the output of the second phase is a weighted graph $\mathcal{G}_w = (\mathcal{V}, \mathcal{E}_w, w)$. After the second phase, some links that previously had $p(u, v) = 0$ can now possibly have a non-null score, $w(u, v) > 0$, thus extending the overall recall of the method.

2.2.3 Result presentation

In most real-world social networks, the links in a social graph $\mathcal{E}_s$ form only a small fraction of the total number of possible links, i.e., $|\mathcal{E}_s| \ll |\mathcal{V}|^2$. This means that *accuracy* is not a very meaningful measure in this context, given that, by predicting always 0 (the link does not exist), it is possible to achieve an accuracy of 99.93% in our data. Also, comparing different predictors by means of *precision* and *recall* is not very appropriate, given the very low maximum recall achievable (only 0.037 in our data). Therefore, in order to compare the performance of different predictive functions by eliminating the skewness between possible and existing links, we adopt the *ROC curve* metric [72] as the main way of presenting our results.

In a ROC curve, all predictions are sorted by order of confidence. The x axis shows the normalized true positives (good prediction) while the y axis represents the false positives (bad predictions). Therefore, taking each prediction in the sorted list, we draw the ROC curve by starting at (0,0) and moving up for a good prediction and right for a bad one. If the predictor was prefect, the ROC curve would go from (0,0) to (0,1), meaning that it would give higher confidence to the predictions that are actually good, and then from (0,1) to (1,1), since all bad predictions would be at the end of the list. A random prediction generates a diagonal ROC curve, that goes from (0,0) to (1,1) in a straight line.

For the best predictor of each feature group, we also provide *recall/fallout* ratios. Recall is the ratio between the number of *true positives* (correctly predicted existing links) and *positives* (existing links) while fallout is the ratio between the number of *false positives* (links erroneously predicted to exist) and *negatives* (not existing links). Given a predictor function f, we may interpret recall and fallout as the following probabilities, respectively: $p(f(u,v) = 1 | (u,v) \in \mathcal{V})$ and $p(f(u,v) = 0 | (u,v) \notin \mathcal{V})$. Now, suppose to have $recall/fallout = 8$ for a predictor f. This means that for two users $(u,v) \in \mathcal{V}$ for whom a link exists, it is 8 times more likely to have $f(u,v) = 1$ than for two users who are not connected. This measure removes the bias caused by the skewness of the dataset and clearly shows how discriminant a feature is.

2.3 Dataset

Flickr is a highly popular online social network, whose primary objective is to facilitate images sharing among people. As Flickr belongs to Yahoo!, we have an easy access to the social data of the users. In Flickr, a user can place other users in three privilege classes: *contact*, *friend*, and *family*. Depending on the class, the user can restrict access to its properties (e.g., images, videos). In this work, we only consider the contact class to form the social graph we are trying to predict. Although Flickr supports directed links, our data shows that only $1/3$ of the links are unidirectional. Thus, most of the features we use in Section 2.4 are symmetric. This means that we predict the same likelihood for links' existence in both directions.

2.3.1 Dataset preparation

We sample a subset of the entire Flickr social network by applying the *snowball sampling* strategy. Starting from a single, highly connected seed user, we follow the contact links and iteratively add them to the sampled graph. This is equivalent to a breadth-first reading of the graph. The adopted sampling strategy increases the chance of selecting more active users, who have higher connectivity in the network (i.e., more links). In our case, this is desirable as users with few or no friends are relatively less interesting for our prediction task. For each user in the sample set, we store all links and groups associated with the user. From the sampled set of users, we remove the ones who are not members of any group. This is because the proposed techniques are only applicable to users who have at least one group membership.

2.3.2 Dataset properties

After the above-mentioned pruning, we are left with $198,315$ users. The number of existing links is reported in Table 2.1. The same table (second line) also reports how many of these existing links are among two users that have at least one common group: this is the maximum number of links

	Possible links	Existing links
Total	39,328,640,910	28,249,755
Phase 1	54,945,936	553,977
Phase 2	1,165,664,850	1,072,595

Table 2.1: Number and type of links

Property	Minimum	Average	Maximum
# of users in a group	1	8.8	3,497
# of groups of a user	1	3.1	172
# of links of a user	1	142.5	11,956

Table 2.2: Dataset properties

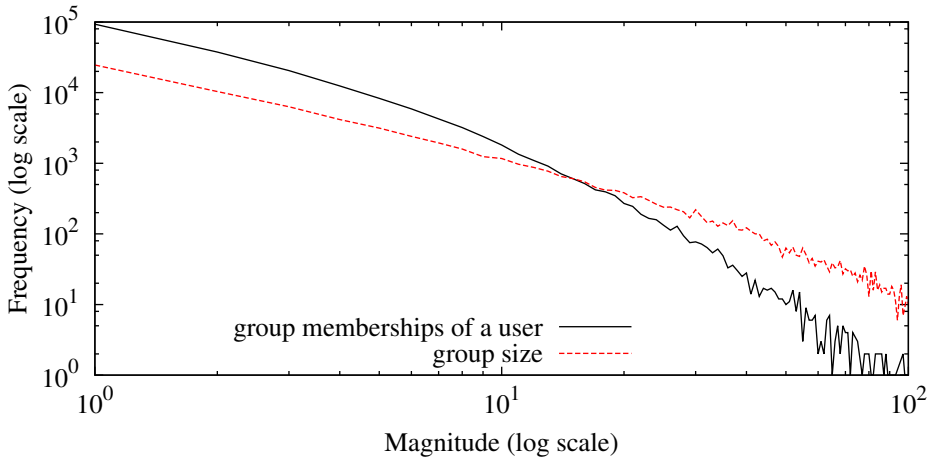


Figure 2.1: Frequency distributions for group membership and group size

predictable in the first phase, or in other terms, links for which we will have $p(u, v) > 0$. In the second phase, we use measures based on paths formed by the links between users. This means that we cannot predict the existence of a link between two users, each belonging to a different connected component of the bootstrap probabilistic graph. The maximum number of links predictable in the second phase is also reported in Table 2.1 (the third line). The number of groups we have is $69,793$. Various properties of our dataset are displayed in Table 2.2.

According to Figure 2.1, the frequency of group sizes follows a highly skewed distribution. There are few very large groups but many very small groups. For instance, 35.3% of groups have only one member and groups of size less than 3 constitute about half of the total number of groups. The frequency distribution for group membership is even more skewed: 47.0% of users are members in only one group while the number of users who are members in at most 10 groups constitutes 95.6% of the total number of users. A highly skewed distribution is also observed in frequency of users' link counts (Figure 2.2). For example, 96.1% of the users have at least one link, and about half of the users have 42 or less contact links.

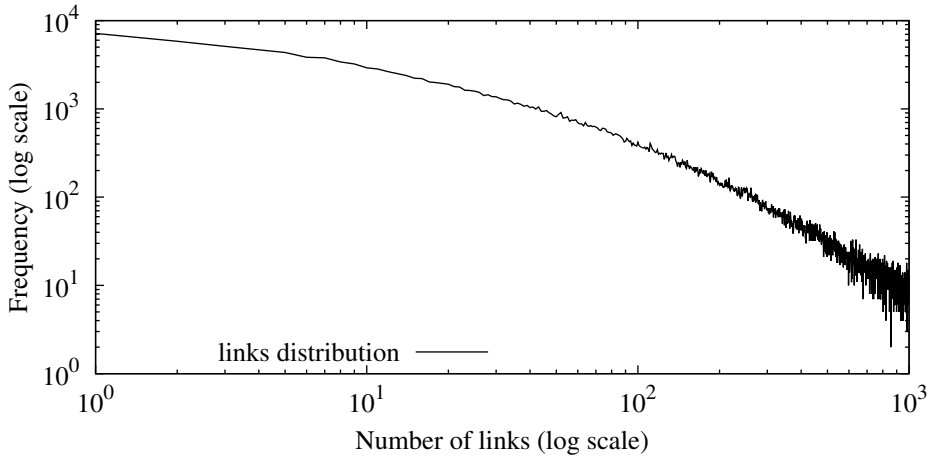


Figure 2.2: Frequency distribution for the number of links that users have

Type	Name	Direction	Formula
Number of groups	`sum`	$\geq$	$\lvert m(u)\rvert + \lvert m(v)\rvert$
	`prod`	$\geq$	$\lvert m(u)\rvert \times \lvert m(v)\rvert$
Common groups	`overlap`	$\geq$	$\lvert c(u,v)\rvert$
	`frac_1`	$\geq$	$\lvert c(u,v)\rvert / \lvert m(u)\rvert$
	`frac_2`	$\geq$	$\lvert c(u,v)\rvert / \lvert m(v)\rvert$
	`jaccard`	$\geq$	$\lvert c(u,v)\rvert / \lvert m(u) \cup m(v)\rvert$
	`cos`	$\geq$	$\lvert c(u,v)\rvert / \sqrt{\lvert m(u)\rvert \times \lvert m(v)\rvert}$
Group size	`min_s`	$\leq$	$\min_{g \in c(u,v)} \lvert g\rvert$
	`avg_s`	$\leq$	$\text{avg}_{g \in c(u,v)} \lvert g\rvert$
	`sum_rec_s`	$\geq$	$\sum_{g \in c(u,v)} (1/\lvert g\rvert)$
	`ad_ad_s`	$\geq$	$\sum_{g \in c(u,v)} (1/\log(\lvert g\rvert))$
Inter-arrival time	`min_t`	$\leq$	$\min_{g \in c(u,v)} t(u,v,g)$
	`avg_t`	$\leq$	$\text{avg}_{g \in c(u,v)} t(u,v,g)$
	`sum_rec_t`	$\geq$	$\sum_{g \in c(u,v)} (1/t(u,v,g)$
	`ad_ad_t`	$\geq$	$\sum_{g \in c(u,v)} (1/\log(t(u,v,g)))$

Table 2.3: Features evaluated in the bootstrap phase

2.4 Bootstrap phase

2.4.1 Basic features

In the first phase, we bootstrap the probabilistic graph using the group membership information. In particular, we explore four types of features: *number of groups*, *number of common groups*, *size of common groups*, and *difference in joining time*. For each of these features families, we experiment intuitive features and adapt classic ones to our dataset. Throughout this section, the reader may refer to Table 2.3 for definitions of the features. Since, in this phase, the probabilities are assigned to only the links between users who share at least one group, in the rest of this section, we report ROC curves

computed only on this subset of links. We denote by $c(u, v)$ the set of groups that are common to both users u and v, i.e., $c(u, v) = m(u) \cap m(v)$. The absolute value of the difference in time that u and v joined group g is denoted by $t(u, v, g)$.

Number of groups

The number of group memberships of a user might be a good indicator of her level of engagement and activity in the social network. As the user is more active, she may tend to have more links. Figure 2.3a shows how the number of links of a user changes as the number of group memberships increases. We observe a very linear behavior, which may indicate a correlation between the number of groups and the number of links. We also observe that the number of out-links increases at a slightly faster rate than in-links as users join more groups.

We evaluate two features based only on the number of groups. Given two users u and v, we define `sum` and `prod`, respectively, as the sum and product of $|m(u)|$ and $|m(v)|$ values. Obviously, as the feature values increase, the likelihood of having a link increases.

Figure 2.3b shows the ROC curves for the two features: according to the plot, the performances of `sum` and `prod` are very close, but `prod` performs slightly better. In Figure 2.3c, we report the recall/fallout ratio for different thresholds of `prod`. The plot gives an indication of how predictive the feature is. As an example, for two linked users u and v, it is 200 times more likely to have $|m(u)| \times |m(v)| \geq 1000$ than for two users without a link.

Common groups

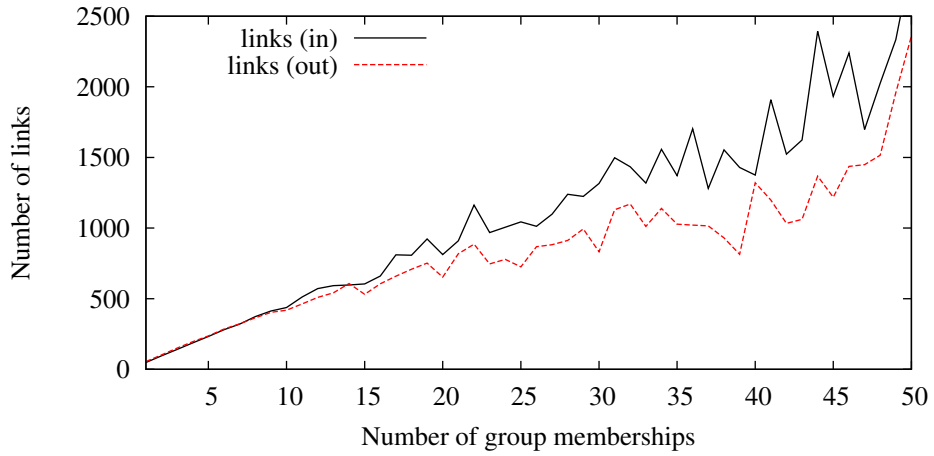
Intuitively, the fact that two users are members of the same groups should be a strong indicator of the existence of a possible link between them. An active group member may influence her existing friends to join the group as new members. This means that members of the same group are more likely to have existing friendship links among themselves. From another perspective, groups may be a suitable medium to meet other users and form friendships, thus groups may lead to the creation of new links. Figure 2.4a verifies this hypothesis by measuring the fraction of links among users having membership in the same groups. Specifically, for each value x of common groups, we compute which proportion of users having x groups in common are friends

$$\frac{|\{(u, v) \in \mathcal{E}_s\,,\, |c(u, v)| = x\}|}{|\{(u, v) \in \mathcal{V} \times \mathcal{V}\,,\, u \neq v \wedge |c(u, v)| = x\}|}\,.$$

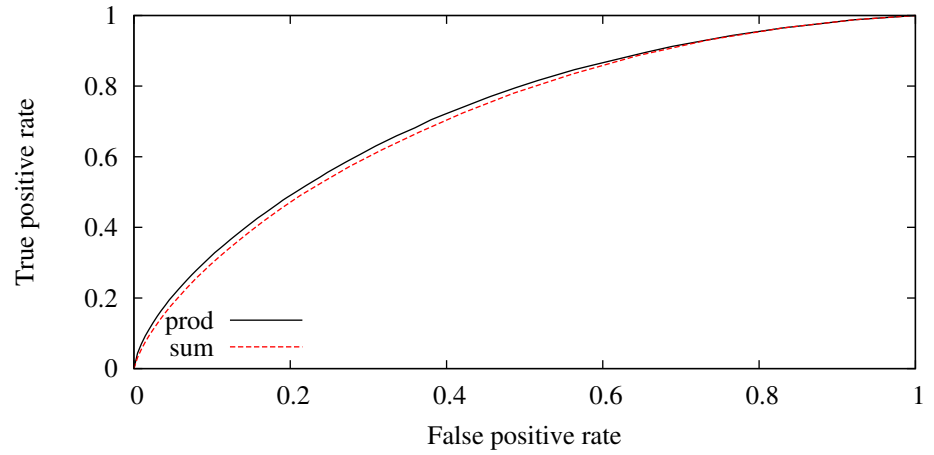
As expected, the fraction of links increases as the users have more groups in common.

We evaluated five different features based on common groups: `overlap`, which is the number of common groups; `frac_1` and `frac_2`, which are the overlap normalized by the number of groups of the first and second users, respectively; `jaccard`, which is the Jaccard coefficient; and `cos`, indicating the *cosine similarity* commonly used in information retrieval.

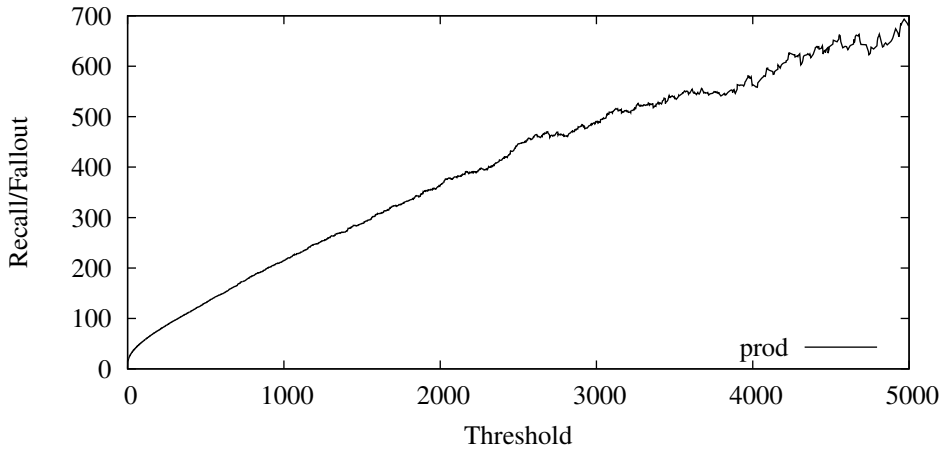
ROC curves of the five features are shown in Figure 2.4b. Interestingly, we observe that all features except for `overlap` perform worse than random prediction, which would correspond to the diagonal of the plot. This can be easily explained with the very high number of pairs of users being members of only one very large group. When two of such users are part of the same group, they receive the maximum value of `frac_1`, `frac_2`, `jaccard`, and `cos` although these two users are very likely not to be linked. We observed this effect also for variations of these features, such as



(a) In-degrees and out-degrees of users as their number of group memberships increases



(b) ROC curves



(c) Recall/fallout curve for `prod`

Figure 2.3: Leveraging the number of group memberships

the weighted cosine similarity feature (using tf-idf weighting[3]), and also for some other features with normalization. It seems that any kind of normalization by the number of groups, which penalizes very active users, has a negative effect on this dataset. This result confirms our previous observations: active users have many friends, thus lowering their scores has a negative impact. Figure 2.4c reports the recall/fallout curve of `overlap` for different numbers of common groups. We can observe that for two linked users, the probability of having more than 10 common groups is approximately 600 times larger than for two users that have no link. This number grows to $1,800$ for the probability of having no less than 24 common groups.

Group size

It can be claimed that two users are more likely to be friends if they are members of a small group than a large group as

- group founders are more likely to prefer their friends over other users in invitations they send
- large groups are more likely to be general-purpose groups, attracting different users with equal likelihood.

We verify this claim in Figure 2.5a, which shows the density of links with increasing group size. The link density is computed as

$$\frac{\sum_{g\in\mathcal{H}\,,\,|g|=x} |\{(u,v)\in\mathcal{E}_s\,,\,u\in g\wedge v\in g\}|}{|\{g\in\mathcal{H}\,,\,|g|=x\}|\times x\times(x-1)}\ .$$

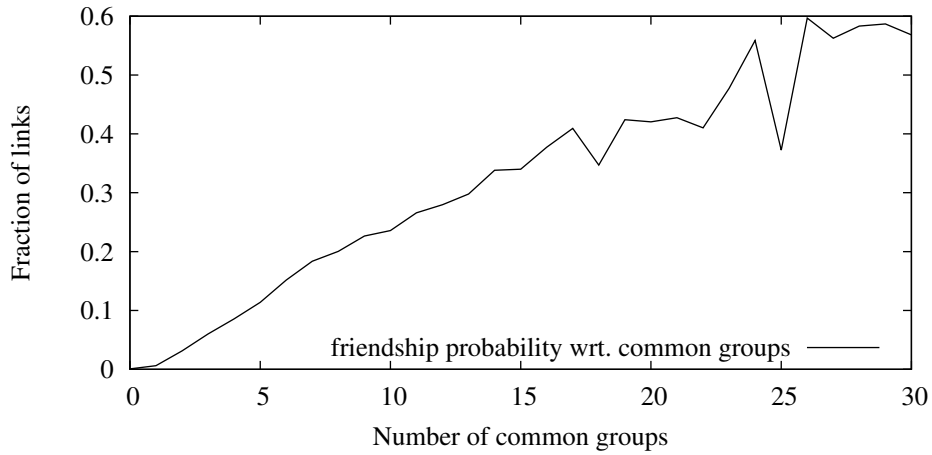
The average link count for groups of size x is normalized with the maximum possible link count ($x\times(x-1)$). We observe a significant drop in density values at very small group sizes, followed by a linear drop as the group size increases.

We try four different features based on the size of common groups: `min_s` and `avg_s` denote the minimum and the average size of the common groups, respectively; the summation of the reciprocal of size is denoted by `sum_rec_s`; and *Adamic/Adar-size* is denoted by `ad_ad_s`. The last feature is inspired by the measure defined by Adamic and Adar in [2] for deciding when two personal home pages are strongly related, and then borrowed and adapted by Liben-Nowell and Kleinberg [59] to deal with common neighbors in the context of link prediction. Here, we re-adapt this measure to deal with the size of common groups and compute

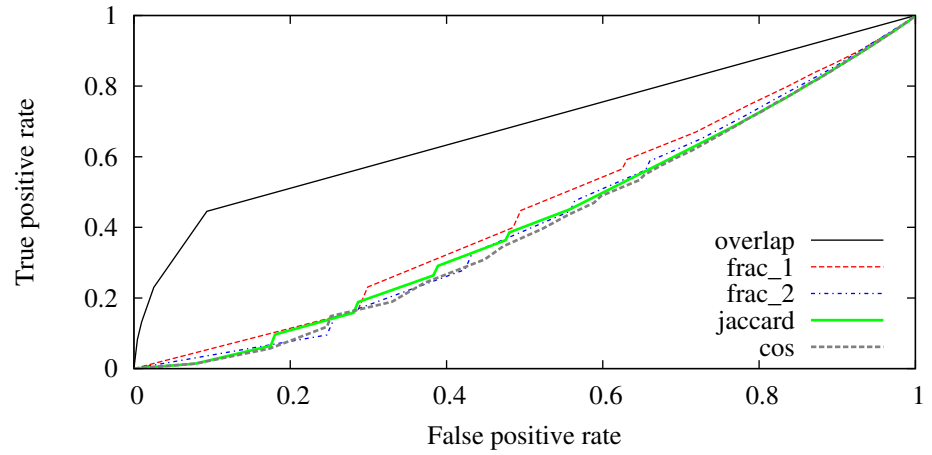
$$\texttt{ad\_ad\_s}(u,v)=\sum_{g\in c(u,v)}\frac{1}{\log(|g|)}\ .$$

Figure 2.5b shows that features that are based on group size perform quite well, with `ad_ad_s` outperforming the others. The recall/fallout curve for `ad_ad_s` is displayed in Figure 2.5c. We observe that the probability of having $\texttt{ad\_ad\_s}(u,v)\geq 1$ for an existing link (u,v) is approximately 200 times larger than for a non-existing link (u,v), and the ratio keeps growing almost linearly.

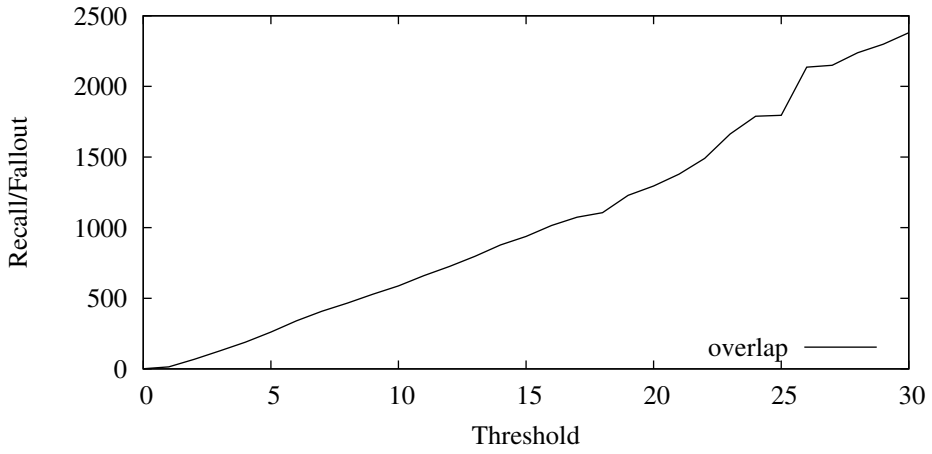
[3] Search engines often use term frequency–inverse document frequency weighting to rank results [73].



(a) Fraction of links between user pairs as their number of common groups increases

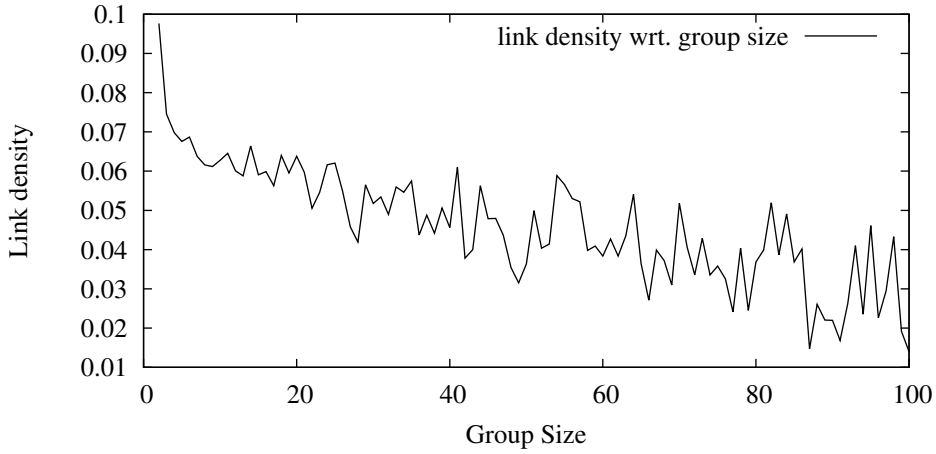


(b) ROC curves

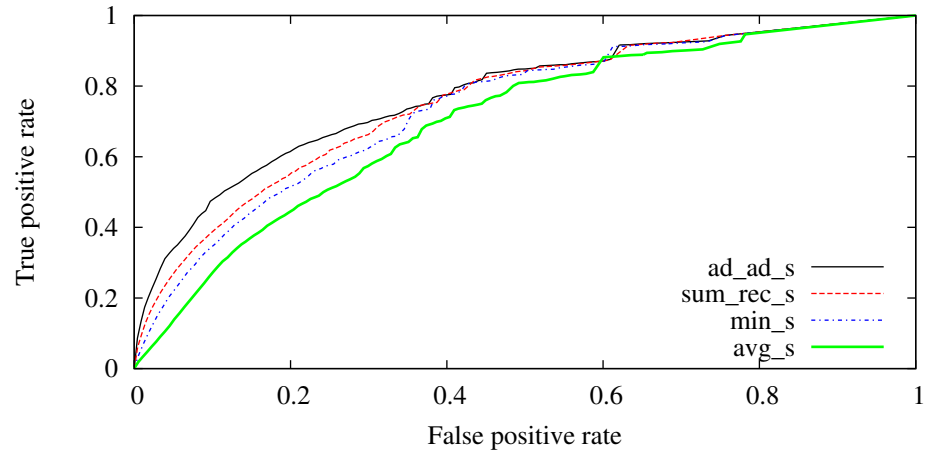


(c) Recall/fallout curve for `overlap`

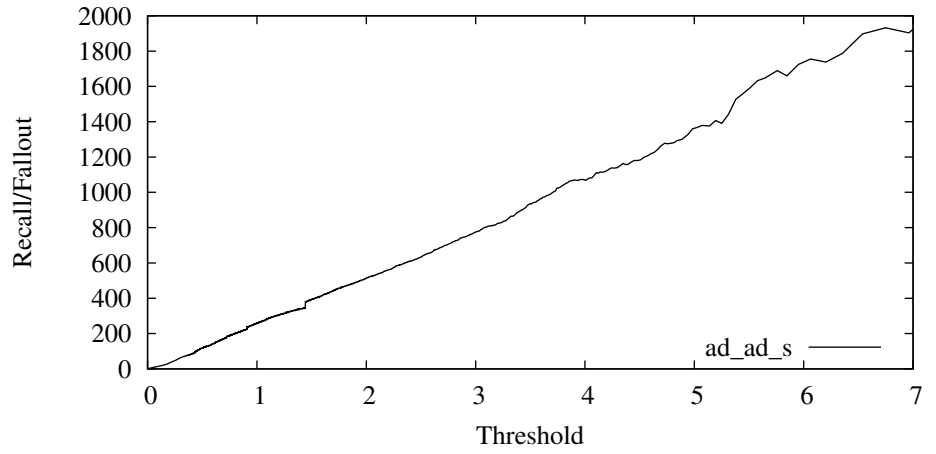
Figure 2.4: Leveraging the number of groups in common



(a) Link density within a group as group size increases



(b) ROC curves



(c) Recall/fallout curve for `ad_ad_s`

Figure 2.5: Leveraging the size of the groups in common

Difference in joining time

In the last set of features, we investigate the temporal consistency between linked users joining the same group. We expect that friends are likely to inform each other of the existence of a group just before (or just after) joining it. Hence, we believe that linked users are likely to join the same group with small time gaps (inter-arrival time). Figure 2.6a shows the fraction of links with increasing inter-arrival time. For each possible inter-arrival time value x (discretized into days), we compute

$$\frac{\sum_{g\in\mathcal{H}} |\{(u,v) \in \mathcal{E}_s\,,\, t(u,v,g) = x \wedge u \in g \wedge v \in g\}|}{\sum_{g\in\mathcal{H}} |\{(u,v) \in \mathcal{V}\times\mathcal{V}\,,\, u \neq v \wedge t(u,v,g) = x \wedge u \in g \wedge v \in g\}|}\ .$$

According to the figure, as expected, linked users are more likely to join the same group with a small inter-arrival time. It is interesting to note that an increase is observed in the likelihood of having a link, around a year inter-arrival time. This may be explained by the existence of "seasonal" groups, i.e., groups that refer to events held once per year and that attract new members in that period (e.g., *"Glastonbury Festival"* or *"Christmas Worldwide"*).

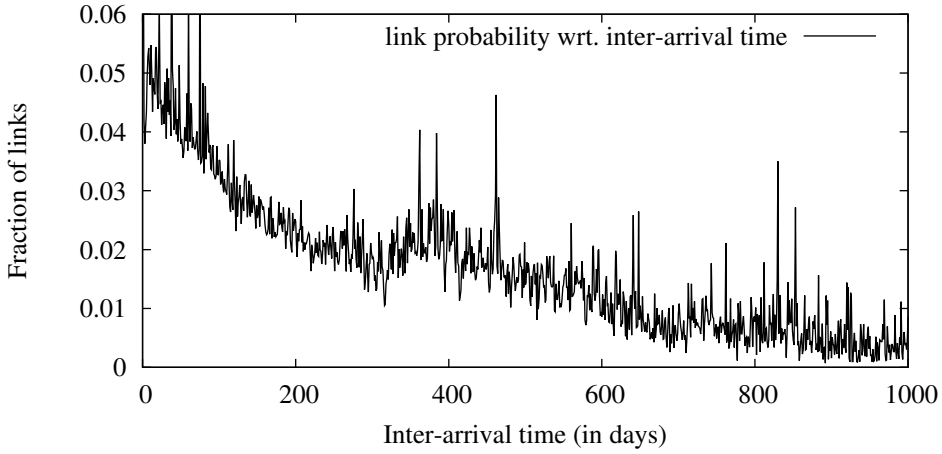
We try features similar to the ones we used for the group size, simply by replacing group size with inter-arrival time: `min_t`, `avg_t`, `sum_rec_t`, and `ad_ad_t`. As in the case of group size features, *Adamic/Adar-time* (`ad_ad_t`) performs the best among all features of this class (Figures 2.6b and 2.6c). The irregular shape of the ROC curve as well as of the recall/fallout curve are due to the seasonal behavior discussed before.

2.4.2 Combining basic features

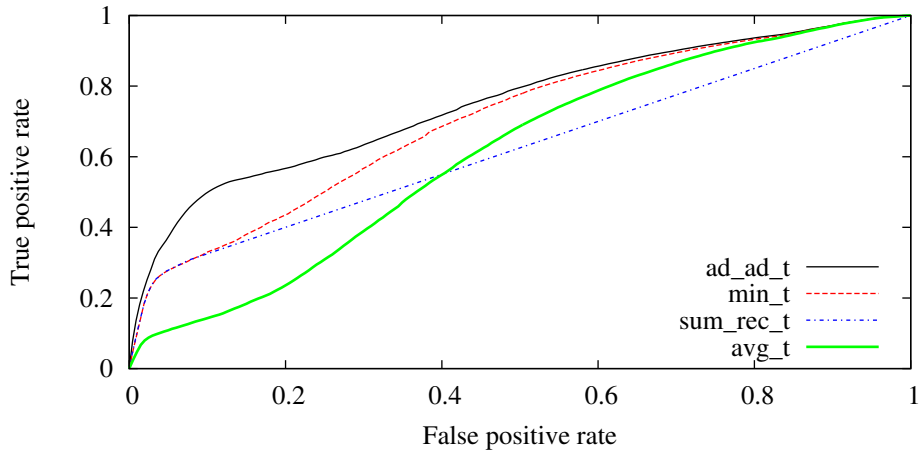
Next, we try to combine a number of features from previous sections to create a single, hybrid feature with higher predictive power than the basic features. For this purpose, we evaluate various possible combinations of our best performing features, trying to find a good trade-off between predictive power and simplicity (which also translates to generality). Optimizing the predictors through machine learning algorithms could lead to better results, but there is absolutely no guaranty that this tuning would also improve the results on an other dataset. The goal in this work is not to create a predictor specific to Flickr; instead just need a generic predictor to generate reasonably accurate predictions and support the second phase of our approach. In our case, best performing combination turns out to be `ad_ad_s` $\times$ `ad_ad_t` $\times \log$ `prod`, referred to as `combined`.

An important observation is that the features `ad_ad_t`, `ad_ad_s`, and `prod` perform relatively well for high, medium, and low confidence intervals, respectively. This is the reason for the `combined` feature, which unifies them, to perform the best across all intervals. No features from the class using the number of common groups is directly used in the combined feature as this is subsumed in the two Adamic/Adar features, which compute a sum over all common groups. Table 2.4 summarizes the composition of the `combined` feature, showing that all the families of features are indeed taken into account.

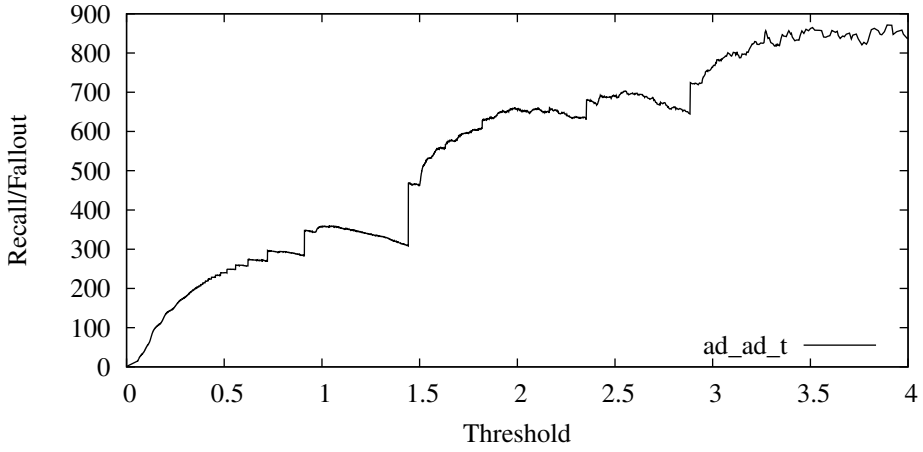
Figure 2.7 compares `combined` against the best-performing feature from each of the four categories. As the figure shows, `combined` takes advantage of all the features that compose it and outperforms all the previous results we obtained. Thus, we use `combined` to bootstrap the probabilistic graph, as shown next.



(a) Fraction of links with varying inter-arrival time



(b) ROC curves



(c) Recall/fallout curve for `ad_ad_t`

Figure 2.6: Leveraging the difference in joining time

Feature family	Feature used
Number of groups	`prod`
Common groups	`ad_ad_t` and `ad_ad_t` (indirectly)
Group size	`ad_ad_s`
Difference in joining time	`ad_ad_t`

Table 2.4: Summary of the features used in `combined`

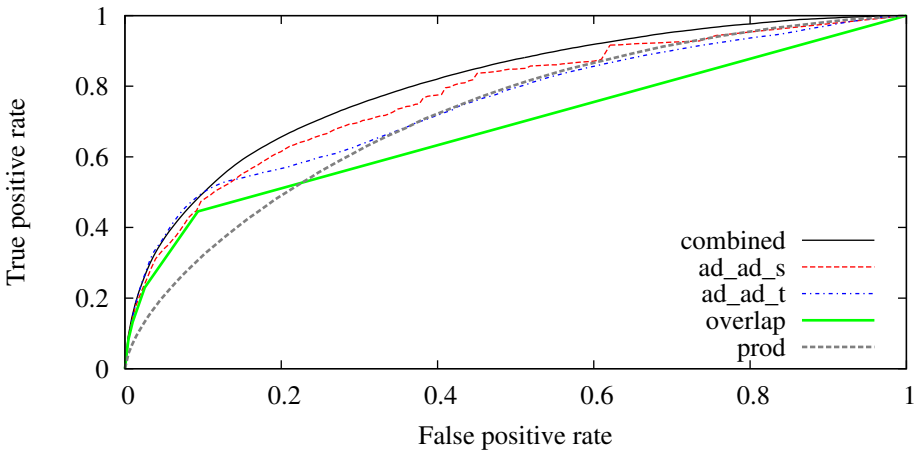


Figure 2.7: ROC curves for the best feature from each category and the best combined feature

2.4.3 Bootstrap probabilistic graph

So far, we have proposed various measures and evaluated their predictive power. We have then combined them under a simple but yet effective feature, namely `combined`. We now finalize the first phase of our method by producing the bootstrap probabilistic graph. As the second phase of our approach uses the paths in $\mathcal{G}_p$ to compute a new score between all pairs of users in the same connected component, we need to be able to combine the values of the edges in a meaningful way. To this end, we have to to convert the scores provided by the `combined` feature into probabilities.

Converting scores to probabilities is not straightforward since the relation between them is often not linear. This problem has been studied for different kinds of classifiers [38, 90, 91, 93], but with score distributions different from the one we observe in our case.

Using the ground truth (i.e. the social graph), $\mathcal{G}_s$, we study the existence probability of edges with respect to the score that `combined` assigned to them. This result is presented in Figure 2.8. We observe a logarithmic shape in the distribution of probabilities with respect to scores. Using a curve fitting algorithm, we could map the function to the data, but depending on the feature used, this mapping could be completely different. In our case, for the sake of generality, we only assume the knowledge that it follows a logarithmic distribution. We design a very simple function that maps the highest score output by the combined feature to a probability of 1 and assigns the remaining probabilities as

$$probability = \frac{\log(score + 1)}{\log(max\_score + 1)} \ .$$

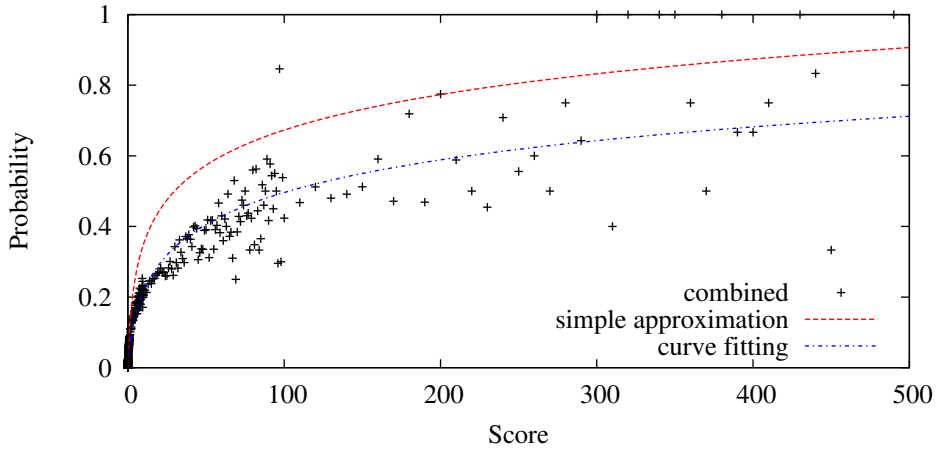


Figure 2.8: Mapping scores to probabilities, for generating the bootstrap probabilistic graph

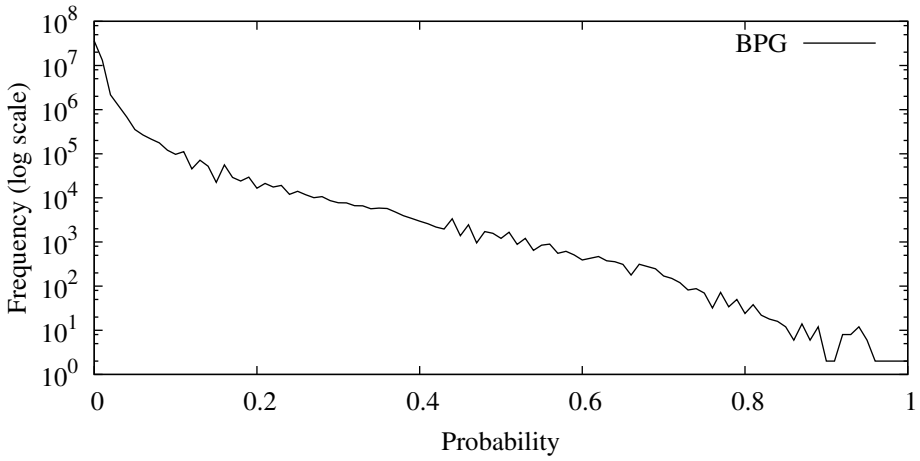


Figure 2.9: Distribution of probabilities in the bootstrap probabilistic graph

Figure 2.8 shows that our simple approximation is very rough. However, it is good enough as it will be shown in the next section. The probability distribution in the bootstrap probabilistic graph is reported in Figure 2.9. The graph consists of $1,238$ connected components, of which 42 have more than $1,000$ vertices, 10 have more than $5,000$ vertices, and the largest connected component has more than $20,000$ vertices.

2.5 Probabilistic graph measures

2.5.1 Graph-theoretic features

In the first phase of our method, we have predicted, for some pairs of users, the probability to have a link in the social network. In the second phase, we refine and extend this prediction by considering transitivity of contact relationship. As shown in [59], users who have many common contacts are more likely to be friends. Using graph-based features, we can spread the link prediction to pairs of

users who have no common groups but share contacts. Therefore, we compute graph-based measures on the bootstrap probabilistic graph for all pairs of users who are in the same connected component. If two users are not connected by a path in the probabilistic graph, then they will have a null probability also after phase 2. In the following, we adapt to the probabilistic case three graph-based measures that are reported to perform well in [59]: `common_neighbors`, `katz`, and `rooted_pagerank`.

Probabilistic common neighbors

Having a high number of common contacts may be an indication of the existence of a link. We adapt this idea to our probabilistic graph in a straightforward way. For a given user pair (u, v), `common_neighbors` simply computes the sum of the probability that each vertex is connected to both u and v, i.e.,

$$\texttt{common\_neighbors}(u, v) = \sum_{w \in \mathcal{V}} p(u, w) \times p(v, w) \ .$$

A consequence of this definition is that all pairs of users who are more than two hops away in the graph are assigned a zero score. Recall that the probability p computed in the first phase is symmetric, i.e., $p(u, v) = p(v, u)$.

Probabilistic Katz

The Katz measure computes a score between two users based on the number of paths existing between them, exponentially damped by length to count short paths more heavily. In other words, a path of length ℓ is weighted by β^ℓ, where $0 \leq \beta \leq 1$. This measures goes beyond the `common_neighbors`, as all pairs of vertices within the same connected component receive a score. It is also more complex than the graph shortest path, since it takes into account the ensemble of path between the vertices. Social networks contain many "accidental" edges connecting otherwise distant vertices. In the small-world theory, these links are called the long links. By taking into account the multiplicity of the path between the vertices, Katz gives higher scores to pairs of vertices that are well connected than to vertices connected through long links. We adapt this measure to deal with probabilistic graphs by further weighting each path by its existence probability, which is the product of the probabilities of the links that compose it.

Let $path_{u,v}^{\langle \ell \rangle}$ be the set of paths of length ℓ between u and v in $\mathcal{G}_p$ and $pathProb(x)$ be the existence probability of a path x. Then, `katz` is computed as

$$\texttt{katz}(u, v) = \sum_{\ell=1}^{\infty} (\beta^\ell \times \sum_{x \in path_{u,v}^{\langle \ell \rangle}} pathProb(x)) \ .$$

Probabilistic rooted PageRank

The `rooted_pagerank` feature computes a score between vertices u and v by running rooted PageRank (personalized PageRank), starting from u. While PageRank [17] computes the centrality of vertices in a graph, rooted PageRank computes this centrality with respect to a limited set of vertices. In PageRank, random walks start on any vertex of the graph and stop with a probability α (usually, $\alpha = 15$). The PageRank score of a vertex is the probability of a random walk to be on

this vertex. In rooted PageRank, instead of starting from any vertex, the random walks start only on the roots. We use an algorithm based on random walks [35] approximation to get an estimation of rooted PageRank scores. Inspired by [71], we adapt it to our probabilistic graph by sampling existing links at each step of a walk using the probabilities in the graph. The walk continues using an existing edge chosen at random as in the classic unweighted version of PageRank. We run W walks for each vertex. `rooted_pagerank`(u, v) is not null if at least one walk starting at u reaches v. Thus, a user u potentially has a positive score with all other users in his connected component. However, if the number of walks and link probabilities are both very low, this may not be the case. If u and v are not in the same component, then the link (u, v) receives a null score.

2.5.2 Experimental results

We evaluate the three features by comparing our method based on the bootstrap probabilistic graph (BPG) with two alternative methods. These methods work on deterministic graphs (DG) obtained from BPG, by selecting only the links with a probability higher than a given threshold. Note that since the probability function we designed to convert the phase one scores into probabilities is monotonic, this is equivalent to a score threshold. It is also worth noting that using a high threshold would seriously compromise the recall of the method. Indeed, a larger threshold implies that fewer links will be selected, which in turn implies a smaller density for the resulting graph and thus a large number of small components. Since graph-based features produce scores only for pairs of users belonging to the same connected component, using a large threshold would give very small recall. Therefore, for the two alternative methods, we use 0 and 0.005 as the thresholds. The methods are accordingly named as DG_0 and DG_0.005. While DG_0 can achieve the same recall as BPG, DG_0.005 can only predict a smaller number of links due to the lower density as discussed above.

Figure 2.10 displays the ROC curves for `common_neighbors`. Despite its simplicity, `common_neighbors` performs quite well with BPG as seen from the sharp rise in the true positive rate for predictions with high confidence (early data points). The ROC curve of DG_0 remains always under the curve of BPG. DG_0.005 produces results closer to BPG, but at the price of a lower recall. In the figure, some portion of the curves are not displayed for better visibility of the rest. The last data point is $(0.607, 0.953)$ for DG_0 and BPG curves.

For `katz`, following [59], we set β to 0.005. For scalability reasons, we also set an upper bound on the path length. Since BPG is quite dense, the number of paths becomes important for large values. As β is small and thus long paths have very little weight, the impact on precision is negligible. In our experiments, pairs that are more than two hops away in BPG receive a zero score. Figure 2.11 shows the performance of `katz`. BPG still outperforms DG_0 and DG_0.005, but the gap is small, relative to `common_neighbors`.

For experiments on `rooted_pagerank`, we set α to 0.15 and W to $1,000$. According to the ROC curves shown in Figure 2.12, `rooted_pagerank` performs poorly, relative to `common_neighbors` and `katz`, in terms of both coverage of predictions and their quality. Increasing W does not have a significant effect on the result quality, but it increases the coverage of predictions. We believe that `rooted_pagerank` can be efficiently applied on a given user to sort the other users by contact probability. However, the scores obtained through this measure are not comparable across different users. This is due to the fact that PageRank shares a fixed, total score among all vertices. The sum of all PageRank values is always 1, since they represent probabilities. Assuming that a vertex u is part of a very clustered community, with 10 potential neighbors connected

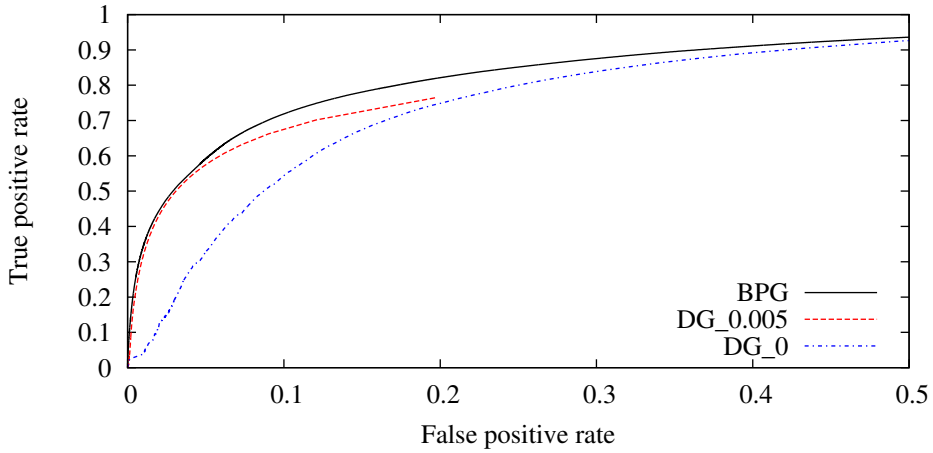


Figure 2.10: ROC curves for the `common_neighbors`

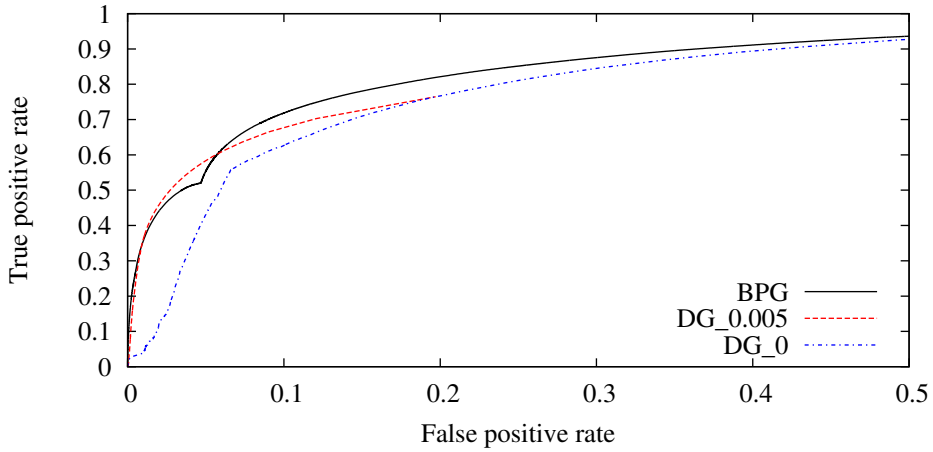


Figure 2.11: ROC curves for the `katz` feature

to u in an identical way. All those neighbors will get a score which is at most 0.1, since they share the total PageRank score. Now, consider the vertex v which is at the extremity of a chain of 3 users in the graph. While there is only one path between v and the other extremity, this vertex will get a very high PageRank score and this prediction will come before the ones of u. As a consequence, a user with many potential neighbors is assigned a score lower than a more isolated one, resulting in poor quality predictions at a full graph scale.

Figure 2.13 brings together the ROC curves for the `combined` feature and the three phase 2 features (assuming the BPG scenario). The plot demonstrates the gain achieved by phase 2 over the results of phase 1. The performance of `katz` is seen to be very close to `common_neighbors`. Note that, in this plot, the true and false positives for the `combined` feature is normalized using the total number of positives and negatives for phase 1.

As a representative, Figure 2.14 shows the precision–recall plot for the `katz` feature. BPG achieves pretty high precision values relative to DG_0 and DG_0.005 under the equal recall constraint.

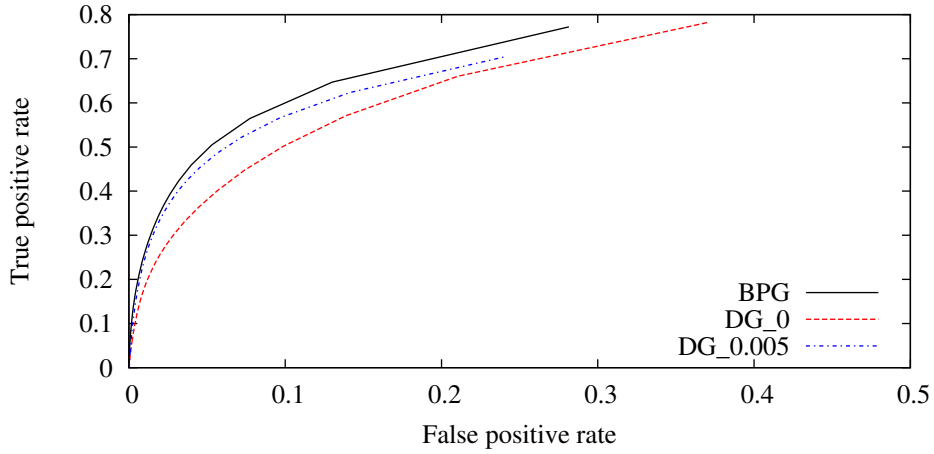


Figure 2.12: ROC curves for the `rooted_pagerank` feature

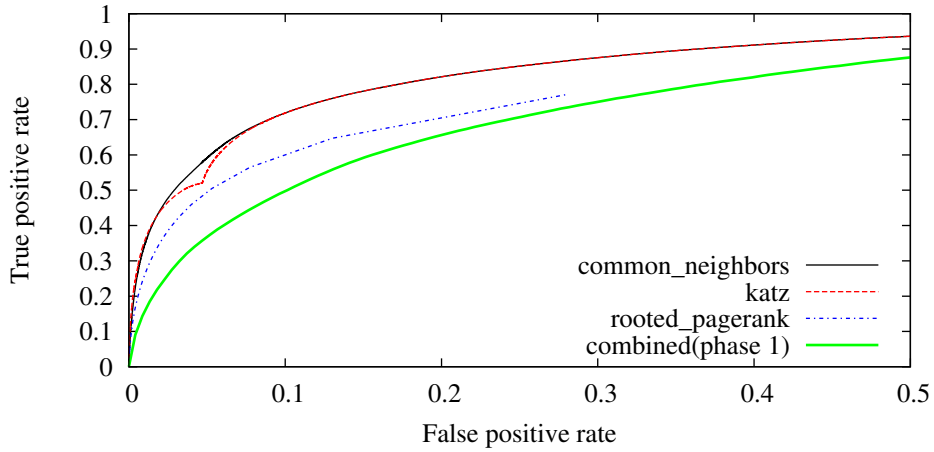


Figure 2.13: ROC curves for the three phase 2 features and the `combined` feature from phase 1

Despite the difficulty of the problem, precision and recall values achieved by BPG indicate the validity of the proposed method (e.g., at a recall of 1%, we observe a precision around 31%).

As another representative, Figure 2.15 shows the recall/fallout curve for the `common_neighbors` feature. BPG is able to leverage the low probability edges to increase recall without losing precision while non-probabilistic approaches either prune this noise and lose recall or use these edges but suffer from low precision.

In this work, we have preferred not to combine the predictive power of the three features of phase 2 (e.g., by means of machine learning techniques). The rationale behind this choice is two-fold. First, as we have explained before, `rooted_pagerank` is not suitable for prediction at a full graph scale. Second, `common_neighbors` can be seen as a special case of `katz` in which the maximum path length is 2 and β is set to 1. Therefore, we expect only little improvement in prediction accuracies by further combinations.

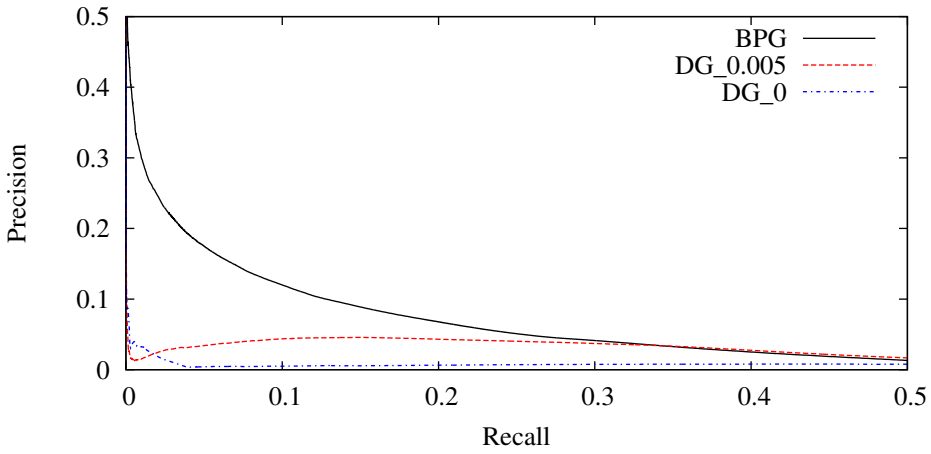


Figure 2.14: Precision–recall for the `katz` feature

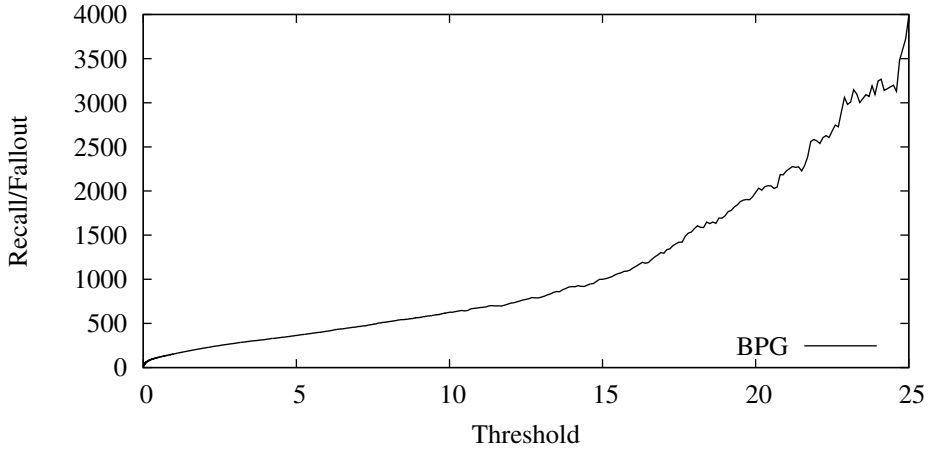


Figure 2.15: Recall/fallout for the `common_neighbors` feature

2.6 Existing work

2.6.1 Link prediction

In this section, we review some of the existing approaches for link prediction. In particular, we detail the work of Liben-Nowell and Kleinberg [59], who introduced the link prediction problem. We also provide some information about the hierarchical random graph approach [25], as we compare against it in Chapter 3.

The link prediction problem *Link prediction*, introduced by Liben-Nowell and Kleinberg [59], refers to a basic computational problem underlying the evolution of social networks in time. Given a snapshot of a social network at time t and a future time t', the problem consists in predicting the new links that are likely to appear in the network within the time interval $[t, t']$. Liben-Nowell and Kleinberg explore a long list of features and evaluate their ability to accurately predict links. These

features include Katz, rooted Pagerank, the number of common neighbors and the graph shortest path. They are classified in three categories: the features based on the vertex's neighborhood, the features based on the ensemble of all paths in the graph, and higher level approaches that transform the graph and are used in conjunction with an other feature. Each feature ranks all possible new edges in decreasing order of confidence. In order to compare the performance of the features, they measure they precision of the top-k predictions for different values of k.

Liben-Nowell and Kleinberg experiment these link prediction features on co-authorship networks from different scientific communities. They compare the performances by measuring the improvement over a random prediction. Their first observation is that, despite the common nature of the experimental graphs, the quality of the features varies a lot depending on the graph. Nevertheless, the Katz feature consistently obtains a good performance. They also note that simple measures like the number of common neighbors are reasonably precise and often outperform much more complex features.

Liben-Nowell and Kleinberg highlight some of the reasons that make link prediction a difficult task. Social graphs have small-world properties [52], which means that they are clustered, but also contain random edges. Thus, these graphs have a small diameter, which limits the performance of the graph shortest path feature. Therefore, a good link prediction feature has to be robust to this noise. An opposite problem comes from the fact that most of the edges that appear are between vertices at distance 3. Hence, the number of common neighbors is not sufficient to predict all the links, and a good feature has to consider the paths in the graph. Finally, the diversity of the social graphs makes it difficult to create a feature that consistently provides accurate predictions. Different communities have different behaviors, and this affects the performance of the predictors.

The graph features presented in Section 2.5 are inspired by Liben-Nowell and Kleinberg's work but they are adapted to probabilistic graphs. These features can be very costly in terms of computation. Liben-Nowell and Kleinberg's experiment on graphs of small size (less than $6,000$ vertices). In our case, the graph contains almost $200,000$ vertices, which makes the computation much longer. Therefore, we had to approximate the results of Katz and rooted Pagerank. This demonstrates the scalability problem of many link prediction approaches.

In this thesis, we evaluate the precision of link prediction algorithms using ROC curves. ROC curves provide an information comparable to the top-k precision, as the sloop of the ROC curve is a good indicator of the precision of the feature. The main difference is that the top-k precision considers the precision over the k first predictions, while the ROC information is closer to the precision at the k^{th} prediction. Liben-Nowell and Kleinberg compare their features with a random predictor. In ROC curves, a random predictor generates a diagonal line, which is why ROC curves are a good way to represent link prediction results.

Hierarchical Random Graph (HRG) Clauset *et al.* [25] propose a method that detects the community structure of the social graph and weights predictions according to the likelihood that the two vertices belong to the same community. Their model, called *hierarchical random graph (HRG)*, samples dendrograms of the social graph depending on their probability to generate the observed graph. The link prediction then combines the scores of the sampled dendrograms. Clauset *et al.* apply their method to a wide variety of graphs including a metabolic network, a terrorist network, and a food web. The main originality of this approach is that it can be applied to both assortative graphs and disassortative graphs. For example, the terrorist network is an assortative graph as is

is very clustered, while the food web is disassortative, as herbivores eat plants but do not eat other herbivores.

They evaluate their results under different graph sparsity conditions, by varying the number of edges removed, and use the AUC measure to present the performances. As many link prediction features, this approach does not scale with the size of the graph, which is why the authors limit their experimentation to graphs of small size. In most cases, they obtain good results, but do not always outperform simple methods like the number of common neighbors.

This method relies heavily on the community structure of the graph, and thus cannot be applied in the setup presented in this chapter, as the social graph is hidden. However, in Chapter 3, we consider a more classic link prediction problem and compare against HRG. The AUC statistic, used by Clauset *et al.*, is equal to the Area Under the ROC Curve. A ROC curve therefore gives more details about the results, as it shows the evolution of the prediction quality, from the first prediction to the last. The AUC is very useful to summarize a ROC curve and present results with respect to an other parameter like the sparsity of the graph in their case.

Other approaches Taskar *et al*. [81] apply link prediction to a social network of universities. They rely on machine learning techniques and use personal information of users (music, books, etc.) to increase the accuracy of predictions. Following a similar approach, O'Madadhain *et al.* [68] focus on predicting events between entities and use the geographic location as a feature. In both cases, the authors improve the efficiency of the link prediction techniques through the use of auxiliary information, but they still have access to the social network. In the cold start link prediction problem, the auxiliary information is the only information available, as the social network is hidden.

Several probabilistic models such as Markov logic [29], relational Markov networks [81], Markov random fields [21], and probabilistic relational models [39] have been used to efficiently capture the relation in data. Again, these approaches have not been proved to scale as they have been tested only on small datasets. We suspect that they would not scale to a large social graph like the one we consider in this chapter.

Van der Aalst *et al.* [82] extract a social network from logs of interactions between workers in a company. Similar works include mining email communications [10] and proximity interactions [32]. In each case, the authors start with a very dense graph and the idea is to identify the social network in this graph. The difficulty of the task is due to the huge amount of data. In our problem, we have the opposite situation: the information used to generate the bootstrap probabilistic graph, which enables link prediction, is very sparse. Hence, the information needs to be spread, not pruned.

2.6.2 Privacy

Since we deal with reconstructing information that is often considered sensitive (the links of a social network), our work has privacy implications. In fact, our method can be used by an attacker to threaten link privacy in a social network, thus it can be used to test the resilience of anonymization solutions. Several papers [6, 43, 64] study the problem of social network anonymization and the impact of the available knowledge on the inference of hidden information that should remain secret. We are particularly interested in the work of Zheleva and Getoor [95]. They show how the Flickr social network and groups can be used to predict private information.

Mixed public and private profiles Zheleva and Getoor [95] consider a social network in which some users hide their profile information by making it private, while other users make it public. They propose several methods to leverage the social links and groups information to predict the content of private profiles. Their experiments include datasets from Flickr and Facebook.

Their Flickr experiments show that the Flickr social network does not leak a lot of private information, as the Flickr social graph connects very different people. However, they obtain much better results using the group membership knowledge. They further improve the quality of their prediction by selecting only the groups that have a low entropy (using the public profiles).

These results are very interesting as the stress the need for decentralized social networks. Their observations confirm our experimental results: the Flickr groups are not very correlated with the social network. In the setup we consider, the social network is completely hidden. Yet, if some of the users made the list of their contacts public, we could improve the performance of our prediction by also taking into account the entropy of the groups.

2.7 Concluding remarks

We presented the cold start link prediction problem and a two-phase method that enables link prediction in the absence of a social network. The first phase of the proposed method generates a bootstrap probabilistic graph using the available social information while the second phase applies various link prediction algorithms to this probabilistic graph. We tested our approach over a dataset obtained from Flickr, by using group memberships as the only available information.

For the sake of generality, we applied our method to interest groups, a very simple and common kind of information in social networks. Thus, the features we present can be applied to other networks. Obviously, as more information is available, higher prediction accuracies can be achieved.

In the context of Flickr, we might use information that is more specific to photography in order to improve prediction performance. For instance, in [76], Singla and Weber study the impact of the social network on camera brands of Flickr users. We could leverage such information to create more accurate predictors. Similarly, as observed in [20], information diffusion often follows the social network. This is known as the social cascade phenomena and can be observed in Flickr when users favorite others' pictures and post comments about them. If this information is available, it can be used as a bootstrap feature to generate a probabilistic social graph that matches the observed social cascade.

As pointed in Section 2.6, we believe that our method can be applied as an attack against link privacy in social networks. Determining to which extent our approach can be combined with existing attacks to improve the predictive power of publicly available attributes is worth future research. Yet, we believe that this highlights the need for a decentralized architecture for social networks. Indeed, the efficiency of these attacks is directly related to the amount of data available. In a decentralized system, each user could be responsible for her own data, which would make the task of collecting a sufficient amount of data to threaten the privacy of the users much more difficult.

In this work, we have adapted the graph-theoretic algorithms in [59] to probabilistic graphs. Potamias et al. [71] introduces different measures of distances in probabilistic graphs and present algorithms to compute k-nearest neighbor queries. We believe that probabilistic graphs are a powerful tool and designing algorithms to extract their characteristics can create new approaches also to other research problems.

CHAPTER 3

DISTRIBUTED LINK PREDICTION FOR SOCIAL NETWORKS

Abstract

In this chapter, we propose SoCS, a distributed algorithm to predict links between the users of a social network. *Link prediction* is a very important feature of social networks, as it helps users to discover new friends. In order to deploy social networks on P2P systems, we need to be able to implement it in a fully decentralized manner. The algorithm we propose embeds the social graph in a metric space. Vertices are assigned social coordinates, which reflect the proximity between users. The closer they are, the more likely they are to be connected. We rely on force based embedding methods, traditionally used in the graph drawing community, to compute the coordinates. SoCS uses P2P gossip algorithms to discover socially close vertices. We evaluate SoCS on real social graphs as well as synthetic topologies. We show that SoCS is able to accurately predict links in social networks, and is adapted to a P2P context, as it is very resilient to churn and requires a small amount of network communications.

This work was realized in collaboration with Anne-Marie Kermarrec and Gilles Trédan. It is now under submission.

3.1 Introduction

Context

P2P networks are becoming an appealing platform for social networks[1] as they offer opportunities to increase the privacy of the users. In this chapter, we address the problem of the distributed computation of *link prediction* [59]. Link prediction is a very important functionality in social network, as it assists users in finding new contacts in the social graph. Decentralized link prediction systems are appealing for two main reasons. First, many link prediction algorithms involves heavy computations and do not scale well with the size of the graph. As social networks now involve millions of users, a distributed approach enables each user to participate to the computation, hence reducing the scalability problem. Secondly, central servers storing all the data enable private companies to predict very accurately the user's behavior [33]. In Chapter 2, we show how very little social information is sufficient to start predicting sensitive data. It is therefore crucial to decentralize social systems, as it is an important first step to protect the privacy of the users.

Building a distributed link prediction system is a challenging task. More precisely, this requires to extract a notion of *proximity* between vertices in the whole social graph. The intuition is that only the *closest* entities to a given vertex are considered to guide the recommendations made to this vertex. In this chapter, we consider the problem of link prediction under its traditional form. We assume that the social network is available, and only rely on the knowledge of the social graph to compute the proximity between vertices. However, contrary to centralized link prediction approaches, we do not rely on features that use the social graph globally. Instead, we assume a distributed and more privacy preserving setting, in which the knowledge of each vertex is limited to its list of graph neighbors. Thus, in the solution we present, vertices do not exchange information about the graph, they are limited to their local view of the social graph. It would be impossible for a vertex to discover new vertices without first learning about their existence. Therefore, in our system, vertices exchange information about their knowledge of the proximity between vertices. They provide information about vertices they consider to be close, but do not reveal which ones of them they are connected to in the social graph.

It turns out that most of the social graphs reflecting real life relationships exhibit a *community structure*. In other words, these graphs are often composed of smalls and internally well-connected subgraphs (the communities). These communities are sparsely connected by *bridges*, or *long links*. A user in a social network is usually connected to some communities such as her professional environment, her university friends, her childhood friends, etc. Connections between these communities may exist and the user precisely acts as a bridge between them. Extracting the community structure from a graph is crucial to guide a recommender system to predict links within the community.

This extraction is however not straightforward and is a computationally expensive task that is hardly compatible with dynamic graphs. Moreover, the traditional definition of community, still under discussion, is fairly rigid: a vertex either belongs to a community or does not. This does not fit the current Internet setting. Overlapping communities have only been recently considered (see *e.g.* [69]). In this chapter, we substitute the (tree-shaped) traditional community structure by a notion of community space, that we call *social* space. In this space, communities are no longer strictly defined as sets, but vertices from the same community belong to the same area. This definition allows

[1] `http://www.joindiaspora.com/`

vertices in the social space to be in the core or at the border of a community, as well as in multiple communities.

To illustrate this intuition, consider a graph representing the friendship relations between people. People living in the same city are more likely to be friends. As a consequence, the majority of a link in the social graph bind people living in the same city. Nevertheless, a user can also have many, geographically distant friends, that all share a common interest for a specific music type for instance, and therefore interact with each other. In this example, a recommender system should be able to differentiate these two communities. It should recommend to the user new friends from both her city and her music community. But it should not predict more links between these two different communities. This user is a bridge between these communities, but they remain different.

Contributions

In this chapter, we introduce a novel decentralized algorithm based on (non-isometric) force based graph embedding to assist a link prediction system. More specifically, we propose SoCS (Social Coordinate Systems), a fully distributed graph embedding algorithm that embeds a social graph into a metric space. The embedding achieved by SoCS preserves the community structure of the input graph thus enabling to easily predict links. SoCS predicts links between vertices that are close in the social space but are not connected in the input graph. SoCS is a very simple algorithm, less than 20 lines of code. SoCS is based on a gossip protocol and does not require any vertex to have a global knowledge of the system.

As we show in the following, force based graph embedding is natural to distribute, and thus, our technique provides a sound basis for distributed link prediction systems. SoCS can be parametrized with any force-based model and we explore in this chapter the application of two well-known energy models to generate a social graph embedding. We provide empirical evidence that the state-of-the-art force model as presented by theoreticians (the LinLog model [66]) is not necessarily the model providing the best results in a practical setting, when considering only the local contacts. Additionally, we obtain the surprising result that distributed versions of our protocol, that solely rely on local knowledge, often provide a better accuracy than their centralized counterparts. This demonstrates a connection with local non-linear reduction algorithms.

We evaluate SoCS in the context of:

- a terrorist collaboration network
- a science co-authorship dataset
- a synthetic small-world Kleinberg topology [52].

Our results show that SoCS is able to achieve social embedding: links removed from social networks are accurately predicted, and SoCS enables to clearly distinguish between short and long-range links in a small-world network. We also observe that SoCS is efficient and highly resilient to dynamics.

Roadmap

The rest of the chapter is organized as follows. In Section 3.2, we present background knowledge about graph embedding and force-based layout. We present SoCS in Section 3.3 and show part of the experimental results in Section 3.4. Finally, we review existing work in Section 3.5 and conclude in Section 3.6.

3.2 Background

In this section, we present the link prediction problem and provide the background on graph embedding required to understand our decentralized protocol. We also introduce the two force models that we are considering in this work. Finally, we present gossip protocols, which are used to maintain P2P networks.

3.2.1 Link Prediction

The link prediction problem, which we already consider throughout Chapter 2, consists in predicting the links that are likely to appear in a social network. In Section 2.6.1, we present this problem and review some existing approaches. In this chapter, contrary to the work we present in Chapter 2, we are interested in algorithm that only use the link structure.

In this work, we compare SOCS' ability to predict links in a social graph against three other measures:

- The graph shortest path (SP): we compute the unweighted graph shortest path between the two vertices and. The closer the vertices are in the graph, the higher is the prediction probability.
- The number of common neighbors (CN): we first compute the graph shortest path between the two vertices, and then the number of common neighbors. This measure refines the graph shortest path, as when the shortest path is of length 2, we sort the predictions according to the number of common neighbors.
- The hierarchical random graphs (HRG): presented in [25], this method relies on detecting the community structure of the graph and weights the prediction according to the likelihood to belong to the same community. We give more details about this approach in Section 2.6.1.

3.2.2 Graph Embedding

Consider $\mathcal{G}(\mathcal{V}, \mathcal{E})$, an undirected graph of n vertices representing a distributed system ($n = |\mathcal{V}|$). Let $\mathcal{P}$ be the *host* space and let $\dim \mathcal{P} = d$. A graph embedding is a *mapping* of the graph vertices to positions in the host space. In other words, each vertex $i \in \mathcal{V}$ get assigned to a point P_i in the host space $\mathcal{P}$.

We denote $d_G(i, j)$ the graph shortest path distance between vertices i and j, and $||\overrightarrow{P_iP_j}||$ the (Euclidean) distance between images for vertices i and j in $\mathcal{P}$ by the embedding. In other words, $||\overrightarrow{P_iP_j}||$ is the distance between the projections of i and j in the host space. Many graph embedding algorithms focus on minimizing the difference between graph distance between two vertices and the distance of their respective images in the host space. Formally, the criterion with respect to this goal is captured by the so-called *distortion*: let f be an embedding of $\mathcal{G}$ into $\mathcal{P}$, and let $c \in]0, 1]$ be the distortion. We have:

$$\forall (i, j) \in \mathcal{V} \times \mathcal{V}, \;\; d_G(i, j) \leq ||\overrightarrow{P_iP_j}|| \leq \frac{d_G(i, j)}{c} \;\; .$$

If $c = 1$, the embedding is isometric. It is important to note that there always exists an isometric embedding in a $(n - 1)$-dimensional host space [85]. The dimension d of the host space is an important parameter to consider for distortion. Minimizing the distortion is an important goal when

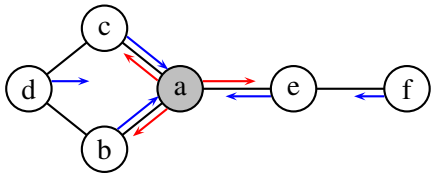


Figure 3.1: Example of a friendship social network

the embedding aims at reflecting as closely as possible the graph distances. Yet, our goal to achieve a social embedding is to precisely have some distortion to distinguish edges that link different communities. Therefore, in SoCS, we use low-dimensional host spaces ($d << n$): our goal is not to minimize the distortion, since an isometric embedding would not respect the community structure.

Force-based graph embedding There are several ways of achieving graph embedding. In this chapter, we explore a subfamily of these techniques, namely the force-based embeddings (FBE), introduced by Eades [31]. Several variations have been proposed since. Intuitively, it is possible to explain the algorithm using a physical analogy: edges represent springs and vertices represent electrically equally charged particles. Edges (springs) attracts the vertices close to each other whereas vertices (particles) repulses each other. This is the physical system simulated by the algorithm. The embedding is achieved once the system reaches an equilibrium.

FBEs are iterative algorithms and rely on two kinds of forces that define the attractions and repulsions that each vertex is subject to in the host space. Initially, each vertex is placed at random in the host space. At each iteration of the algorithm, the forces are applied to the vertices. *Attractive* forces are always applied to a vertex by its *graph neighbors* whereas *repulsive* forces are applied by *all* vertices. We denote by V_i the graph neighbors of the vertex i. Figure 3.1 illustrates this principle: vertex a is attracted by b, c and e, and repulsed by b, c, d, e and f. Let P_i be the image (*i.e.* the position) of vertex i in the host space. Let $||\overrightarrow{P_iP_j}||$ be the distance between P_i and P_j, and let $\overrightarrow{e_{ij}}$ be the direction from P_i to P_j: $\overrightarrow{e_{ij}} = \frac{\overrightarrow{P_iP_j}}{||\overrightarrow{P_iP_j}||}$.

Following Noack's formalism [67], we define forces as proportional to some power of the distance between two images: attraction force $\overrightarrow{A_i}$ and repulsion force $\overrightarrow{R_i}$ applied to a vertex $i \in V$ are respectively defined as

$$\overrightarrow{A_i} = \sum_{j \in V_i} ||\overrightarrow{P_iP_j}||^{fa}.\overrightarrow{ij}, \qquad \overrightarrow{R_i} = -\sum_{j \in V} ||\overrightarrow{P_iP_j}||^{fr}.\overrightarrow{ij} \ .$$

Thus each couple (fa, fr), as respective parameters of the attraction and the repulsion force, defines a new force model. Note that only systems with $(fa > fr)$ produce finite distances (provided the graph is connected).

In this chapter, we study two common FBE models. The first one is Noack's *LinLog* [66] defined by $(fa = 0, fr = -1)$, known as producing layouts that account for community structures. This is considered as the best force model to preserve the community structure of a graph. The second one is the carbon copy of physical Hooke's spring attraction and Coulomb's electrostatic repulsion forces (such as presented in [83]). We denote *HC* this model hereafter, defined by $(fa = 1, fr = -2)$

3.2.3 Gossip protocols

P2P networks can federate millions of machines in a completely distributed manner. In such a large network, it is impossible for a peer to keep track of all the participants. This would require too many network resources, and the volatility of the peers would prevent the peer from maintaining an up to date list of participants. Instead, in P2P, each peer maintains a partial knowledge of the network, which we refer to as its *view*. The graph formed by the peers and the connexions between them is called an *overlay*. The view of each peer contains a few other peers, typically $O(\log n)$, where n is the size of the network. A peer joins the network by contacting an other peer which is already part of the P2P overlay. Many different protocols have been developed to generate different P2P overlay topologies, ranging from very rigid ring structures in distributed hash tables [78], to completely random overlays [46].

In this chapter, we are particularly interested in a specific class of P2P algorithms, known as gossip protocols. Gossip protocols have been widely used to maintain distributed overlays. Each peer regularly contacts an other peer from its view and shares its view information with it. After this exchange, they both update their view, using their previous view as well as the information they received. Following the formalism used in [84], we define a skeleton of a gossip protocol in Algorithm 3.1. This protocol describes the actions performed by a peer u during a cycle of the gossip protocol. The peer v, which is contacted by u upon gossip, performs the exact same actions from line 4. The `makeEntry()` function is used by u to add its own description to the information sent to v. This skeleton contains several undefined functions: `selectDestination()`, `selectEntriesToSend()` and `ComputeView()`. Their implementation defines the behavior of the gossip protocol as well as the properties of the resulting P2P overlay.

Algorithm 3.1 Gossip skeleton for peer u gossiping with v

```
1: loop                                              ▷ Do one cycle every T time units
2:     wait T time units
3:     v ← selectDestination()                       ▷ Select a gossip destination
4:     sendBuf ← { makeEntry() } ∪ selectEntriesToSend()
5:     send sendBuf to v                             ▷ Share the view information
6:     receive recBuf from v
7:     view_u ← ComputeView()                        ▷ Update the view
8: end loop
```

Voulgaris and Steen define, in [84], two different gossip protocols using this framework. The first one is Cyclon, a Random Peer Sampling (RPS) protocol, which provides each node with a random view of the network. Therefore, `selectEntriesToSend()` selects half of $view_u$ at random and `ComputeView()` merges the remaining part of view with the peer descriptors received from v. Cyclon ensures the connectivity of the overlay, and has very good load-balancing properties. The second protocol, Vicinity, clusters the network by grouping peer which are similar together. In some cases, clustering the network could partition the P2P overlay: if all the peers hold an integer value $i \in \{1, 2\}$ and select their view according to this value, then the clustering protocol will generate at least two different connected components, one with all peers having $i = 1$ and the other with the peers having $i = 2$. Now consider a new peer joining the network with $i = 1$. If the peer it contacts to join the overlay has a value $i = 2$, then it will not be able to discover peers having $i = 1$, as they are in a different connected component: its view will not converge to the optimal.

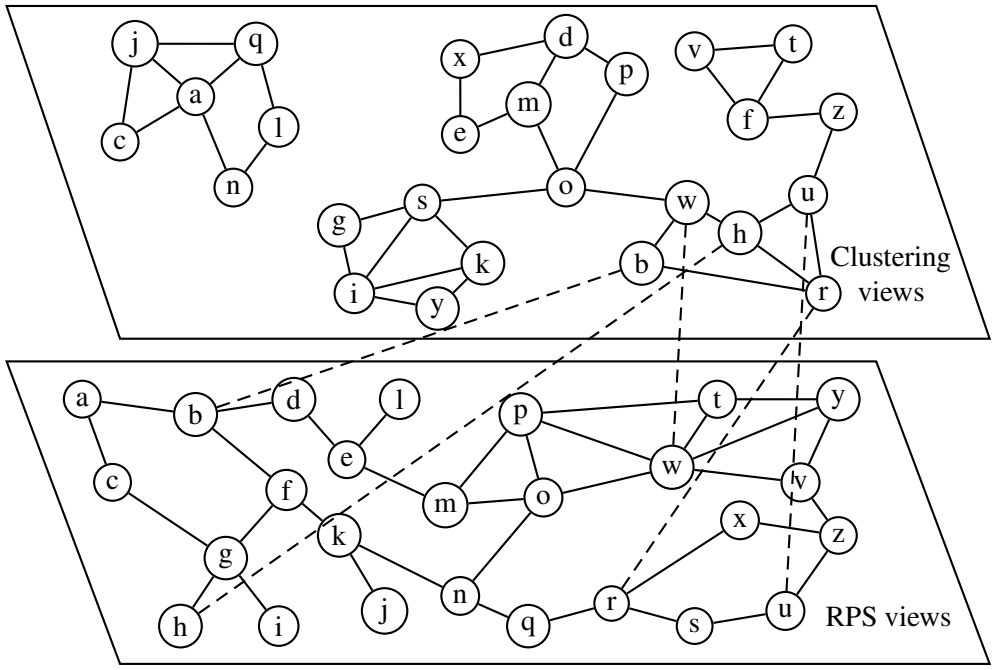


Figure 3.2: Clustering protocol associated with a RPS

Therefore, clustering protocols are used in conjunction with a RPS. In a clustering protocol, the `selectEntriesToSend()` function makes a list of the entries from $view_u$ closest to v according to the clustering criterion, and the `ComputeView()` selects the node which are closest to v from $view_u$, the peer descriptors received from v, and the RPS view of u. The RPS ensures that the clustering view will always converge, as it provides the peer with continuously changing random peers. Figure 3.2 illustrates the association between a clustering protocol and a RPS.

P2P networks are subject to churn, peers join and leave the network. As already mentioned, a peer u joins the network by contacting any already connected peer. The other peers of the network discover its existence through gossip, as u transmits its descriptor (Algorithm 3.1 line 4). The gossip protocol also detects nodes which have disconnected when they are selected as gossip destinations. In [84], the authors use a time-stamp mechanism to ensure that the peer descriptors of disconnected nodes are quickly removed. The `selectDestination()` function is then biased in order to select the oldest entries. The trade-off between the freshness of the view entries and the quality of a RPS is further studied in [46].

In this chapter, we rely on two gossip protocols, a RPS and a clustering protocol, to maintain the SoCS overlay.

3.3 Social Coordinate System (SoCS)

In this section, we describe how SoCS performs a distributed embedding of a social graph. For the sake of clarity, we hereafter add the adjective *social* to every element referring to the SoCS host space. For instance the host space of the embedding is designated as the *social space* and the distance on that space is designated as the *social distance*. Vertices that have close *social positions* are social

neighbors. At this point it is particularly important to distinguish between the graph neighbors of a vertex (the friends of this vertex) and the social neighbors of this vertex, which are not necessarily the same. Indeed, SOCS will recommend to a vertex its closest social neighbors that are not already graph neighbors.

In a nutshell, each vertex that runs a SOCS instance (*i.e.* each graph vertex) regularly computes its position in the social space. Each vertex first gathers its graph neighbors social neighbors positions. Then, using these positions, each vertex computes the forces that are applied to it, and derive its updated position. The rest of the protocol is a gossip protocol that provides each vertex with a list of its social neighbors. This list of social neighbors is then used to compute new positions but also to issue recommendations.

3.3.1 System model

SOCS is a peer-to-peer (P2P) algorithm that embeds a social graph and operates on a P2P overlay network. We consider a system of n vertices and assume a one-to-one mapping between the vertices of the social graph and the machines connected to the P2P network, hence we refer as vertex both to the machine connected to the network and to the logical entity in the social graph. The social graph is an input in SOCS, so each vertex is aware of its graph neighbors. In a dynamic social graph, the set of graph neighbors may vary during the execution. To join an existing SOCS P2P network, a vertex contacts its graph neighbors to establish connexions. Traditionally, in FBEs, each vertex is assigned a random initial position. Indeed, FBEs are usually used in the context of a drawing, so no initial position is known. As SOCS considers the case of a permanent P2P network, a vertex may join an existing network, in which some coordinates have already been computed. In that case, the vertex may initialize its position at the barycenter of its graph neighbors. This first approximation of the optimal coordinates speeds up the convergence since the vertex is initially placed in a good area of the social space.

3.3.2 Rationale

Let us consider the example graph on Figure 3.1. Assume that this graph represents *friendship* relations between people. Vertices d and f are both two hops away from a. From a shortest path perspective, they thus should be considered equally friends to a. Yet, it is very likely that d is *closer* to a since they have two friends in common, namely c and b. This could result in d having twice more chances than f of being invited to a's birthday.

Such social information is typically what SOCS aims at capturing in the resulting social embedding: this is achieved since the attractive forces applied between c and b bring d closer to a in the social space. In addition, due to d's repulsions, c and b are closer to a than e. As a result, the list of friends of a in the resulting space is then, ordered from the closest to the farthest: b and c, e, d, f.

These results can be interpreted as follows: since (a, b, d, c) is a quadrilateral, it has more *cohesion* than a single line, and members of that structure get close positions. The generalization of this intuition is that vertices within the same community get closer positions than vertices from different communities. This is exactly the approach advocated by Noack, that recently bridged community detection and force-based graph embedding [67].

Link prediction algorithms are empowered by the knowledge of communities (see *e.g.* [25]). SOCS achieves such a social embedding, capturing, as the evaluation will confirm, the notion of clusters and communities by placing vertices of a community close to each other in the host space. Since

links are more likely to appear between vertices of the same community, the social distance (*i.e.* the Euclidean distance over the social space) can be directly leveraged to predict links between vertices.

3.3.3 Local repulsions

In SoCS a vertex, in order to compute its position, needs to compute the sum of attractions and repulsions that applies to it. As explained in Section 3.2, attractions take into account the graph neighbors. Since a SoCS vertex is directly connected to all its graph neighbors, computing the attractions is straightforward. We now detail how repulsions are approximated and computed.

Computing repulsion forces based on all vertices is both extremely costly in a large system and hard to achieve in a decentralized way. Instead, SoCS only considers repulsion forces of the *close* vertices in the social space. This optimization was introduced by Fruchterman and Reingold [37] to reduce the time complexity of graph drawing algorithms. It is relevant for several reasons. First, in both force models (especially the HC model), the repulsion force is quickly decreasing with respect to the social distance between vertices: distant vertices' contribution to the force equilibrium is negligible. Second, we target link prediction. We aim at predicting links binding vertices that are already socially close. Long social distances do not necessarily need to be precisely respected. Let $B(i,r)$ to be the set of i's r closest neighbors in P. SoCS computes the following repulsion force:

$$\overrightarrow{R_i} = - \sum_{j \in B(i,r)} ||\overrightarrow{P_i P_j}||^{fr} . \overrightarrow{ij} \ .$$

Therefore, in SoCS, each vertex needs to compute its repulsions based on its local neighborhood. The size of this neighborhood, *i.e.* the number of repulsion is represented by the parameter r. The benefit of this optimization is that each vertex only communicates with a small number of other vertices, either graph or social neighbors (these sets often overlap). We empirically show in Section 3.4 that considering only local forces provides better results while reducing the cost of the algorithm. SoCS relies on a gossip protocol to provide each vertex with its r closest social neighbors in a fully decentralized way.

3.3.4 The SoCS decentralized algorithm

SoCS is a fully decentralized algorithm: each vertex knows only a limited portion of the network, namely its graph neighbors and its social neighbors. SoCS relies on gossip protocols, presented Section 3.2.3, to discover the social neighbors. In SoCS, each vertex runs two different gossip protocols, a *Random Peer Sampling* (RPS) [46] and a *Neighbors Peer Sampling* (NPS). The NPS protocol is based on Vicinity [84] and clusters the vertices according to the social coordinates: the NPS view of a vertex converges to the r closest social neighbors of this vertex. When a vertex joins the network, the NPS view is initialized with its graph neighbors, are they are very likely to be also social neighbors.

Algorithm 3.2 depicts the SoCS algorithm. Initialization is done lines $1-3$. In a static setting ($\mathcal{G}$ does not change over time), we assume that vertices are provided with a function `Converged()` that stops the execution of the protocol on a given vertex. For instance, a vertex that has not changed position over the last couple of rounds, may stop. In a dynamic setting, `Converged()` always returns `false`. Two key parameters allow to tune the algorithm: the number of neighbors considered for computing the repulsions, r and the dimension of the host space, d. The impact of both parameters is detailed in Section 3.4.

Each vertex retrieves the lists of its social neighbors by calling `GetSocialNeighbors()` (this list is obtained from the NPS) and its graph neighbors by calling `GetGraphNeighbors()` (this list is extracted from the graph and present in the vertex's profile).

Instead of computing directly a new vertex position, the presented algorithm computes a couple (speed, direction) toward the ideal position (lines $12 - 13$). This improvement, also used by [37], allows a faster convergence by speeding up vertices in the early steps while avoiding oscillation during the late ones.

Figures 3.3,3.4 and 3.5 illustrate SOCS running on a small-world grid. They respectively illustrate Kleinberg's graph generation process, SOCS' input and output on vertex a, and an illustration of SOCS' system-wide output. Blue vertices represent a's graph neighbors, green vertices represent a's predicted links.

Algorithm 3.2 Graph embedding algorithm, code for Vertex i

1: $P_i \leftarrow$ `GetBarycenter(GetGraphNeighbors())`
2: $v_{prev} = 1$
3: $\overrightarrow{d}_{prev} \leftarrow$ new `RandomDirection()`
4: **while not** `Converged()` **do** ▷ Execute a cycle
5: $\quad\overrightarrow{d} \leftarrow \overrightarrow{0}$
6: $\quad$**for** $j \in$ `GetGraphNeighbors()` **do**
7: $\quad\quad\overrightarrow{d} \leftarrow \overrightarrow{d} + ||\overrightarrow{P_iP_j}||^{fa} \times \overrightarrow{ij}$ ▷ Compute attractions
8: $\quad$**end for**
9: $\quad$**for** $j \in$ `GetSocialNeighbors()` **do**
10: $\quad\quad\overrightarrow{d} \leftarrow \overrightarrow{d} - ||\overrightarrow{P_iP_j}||^{fr} \times \overrightarrow{ij}$ ▷ Compute repulsions
11: $\quad$**end for**
12: $\quad\overrightarrow{d} \leftarrow \frac{\overrightarrow{d}}{||\overrightarrow{d}||}$
13: $\quad v = (\frac{1}{2} \times \cos(\overrightarrow{d}, \overrightarrow{d}_{prev}) + 1)v_{prev}$ ▷ Compute speed
14: $\quad\overrightarrow{d}_{prev} = \overrightarrow{d}$
15: $\quad v_{prev} = v$
16: $\quad P_i = P_i + v.\overrightarrow{d}$ ▷ Move
17: **end while**

3.4 Experimental evaluation

This section presents the experimental evaluation of SOCS. We evaluate SOCS through two different approaches. First, we measure the quality of the embedding generated by SOCS with a link prediction experiment. We present results obtained on two different real social graphs: a computer science co-authorship graph (DBLP) and a terrorist interaction network. We compare these results with some existing approaches and highlight the qualities of SOCS in a distributed environment. Then, in the second part of this section, we evaluate SOCS on synthetic small-world graphs. We generate small-world graphs from a known social topology. This setup provides a full knowledge of the social space and allows us to perform a more precise evaluation of SOCS by comparing the distances in the social space and the ones in the SOCS coordinates space. We present results obtained both in a static setting and in a dynamic network. We demonstrate SOCS' fast convergence and resilience to churn.

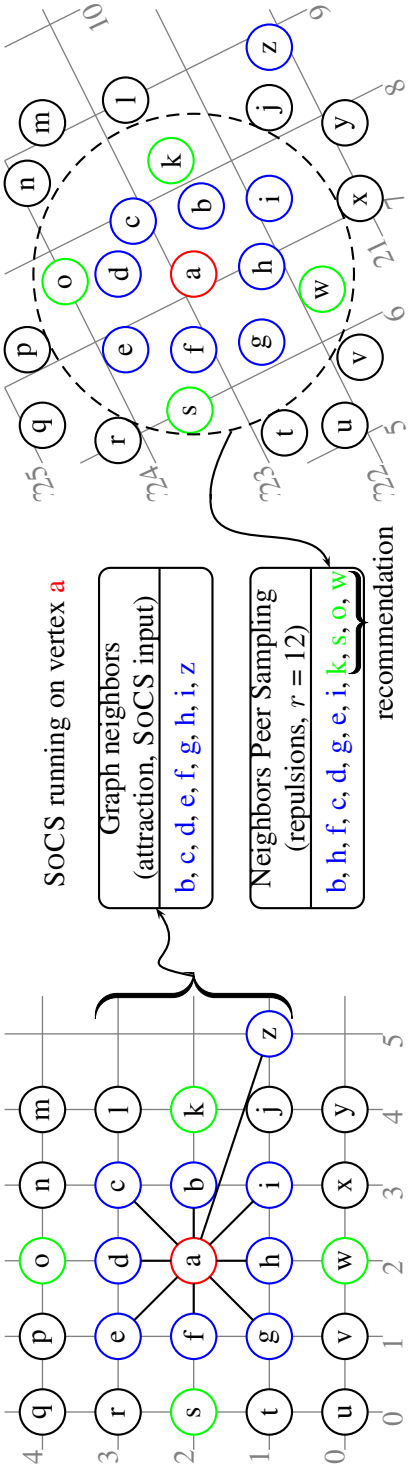


Figure 3.3: Kleinberg small-world grid

Figure 3.4: SoCS on vertex *a*

Figure 3.5: SoCS coordinates space

In our experiments, we evaluate SOCS in different configurations with three parameters. The first one is r, the number of repulsions applied to vertices, introduced in Section 3.3.3. The second one is the force models, presented in Section 3.2.2. The accuracy and the convergence speed of SOCS highly depend on these two parameters. The last parameter is the number of dimensions in the SOCS coordinates space. While it also impacts the precision of SOCS, its influence is much lower than the two previous ones. We show that, quite surprisingly, the configurations that achieve the best prediction quality are very close to the ones that perform well in a distributed environment. Finally, please note that, although SOCS is implemented in a java simulator, it still runs in a fully distributed setting: each vertex is only provided with the list of its neighbors identities, and nothing else. In particular, SOCS does not use any hypothesis about the nature or the size of the graph.

3.4.1 SoCS link prediction quality

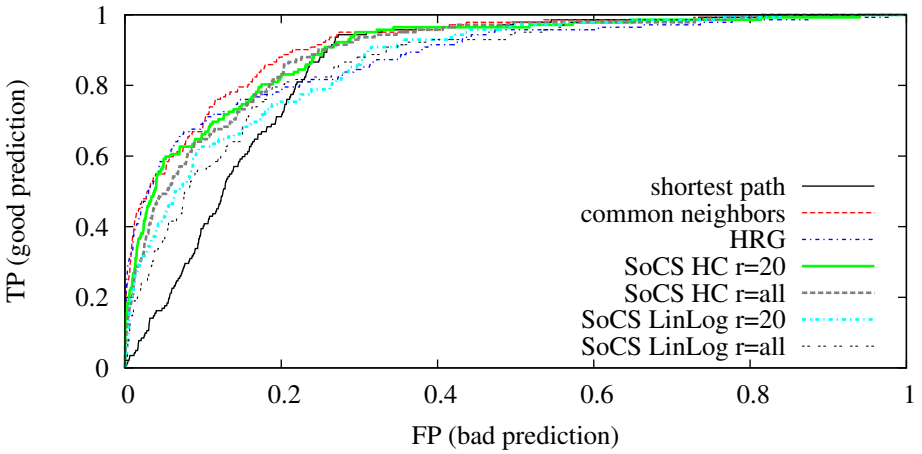
Setup

In this section, we consider two different social graphs. The first one is a terrorist association network [55] of 62 vertices and 152 edges, used in [25] to evaluate HRG. The second one is the DBLP dataset[2], from which we generate a co-author graph. We add an edge between the authors having at least one publication in common, while removing the authors that have less than 20 publications. We keep the biggest connected component, which consists in $27,680$ vertices connected by $211,078$ edges. Note that the edges are not weighted and the number of publications co-authored and the publication dates are ignored in this experiment. In order to perform the link prediction, we rely on the usual approach which consists in removing some edges from the graphs and trying to predict them. More precisely, we select from the social graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a set of edges $\mathcal{E}_h$ at random. We call these edges the hidden links. Then we run the link prediction algorithms on the test graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E} - \mathcal{E}_h)$. The goal is to predict the edges in $\mathcal{E}_h$. The edges removed are always selected at random, but we take care not to disconnect the graph.

We compare SOCS' performance with HRG, as well as two simple approaches introduced in [59]: the graph shortest path (SP) and the number of common neighbors (CN). In the case of CN, we first compute the shortest path between vertices, and the compare the number of common neighbors if the shortest path's length is 2. This feature was also studied in the previous chapter (Section 2.5), but in this case the graph is deterministic so the computation is more simple. We consider the case of a static network, so the social graph is not modified during the execution and we run SOCS until the coordinates have converged. We use ROC curves [72], as well as the AUC measure to display the results. The ROC curves were already presented in Section 2.2.3 and show the evolution of good predictions with respect to bad predictions. The AUC measure represents the Area Under ROC Curve. The bigger this area is, the better the precision is. The AUC can be also interpreted as follows: an AUC value of p means that if a good prediction and a bad prediction are chosen at random, the good prediction has a probability p to receive a higher confidence than the bad prediction. While the AUC is very useful to compare ROC curves, since it summarizes the curve into a single value, it is also less detailed. A ROC curve shows precisely on which portion of the prediction a feature was accurate: two very different ROC curves can have the same AUC.

	SP	CN	HRG	SoCS HC		SoCS LinLog	
				$r = 20$	$r = all$	$r = 20$	$r = all$
1	.851	.914	.881	.899	.894	.869	.864
10	.841	.904	.884	.900	.896	.868	.864
20	.843	.888	.868	.883	.881	.854	.850
30	.837	.871	.859	.869	.869	.837	.835

(a) Terrorist network link prediction AUC



(b) Terrorist network link prediction

Figure 3.6: Terrorist network link prediction

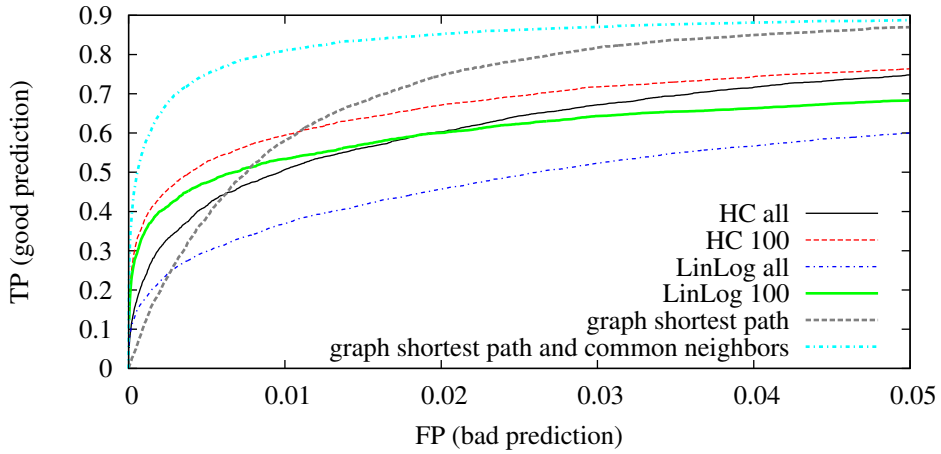


Figure 3.7: DBLP link prediction

Evaluation

Figure 3.6a displays the AUC of the link prediction on the terrorist association graph for an increasing number of edges $|\mathcal{E}_h|$ removed. We configure SoCS to use 10 dimensions. As already observed in [59], SP does not perform well, while CN is very accurate. We observe that SoCS performs best with the HC force model. It's also very interesting to notice that a lower value of r also improves the accuracy of the prediction. By reducing the amount of information gathered by each vertex, SoCS eliminates the impact of distant vertices and reduces the noise in the data. Thus, vertices get a better position and also need less bandwidth to run SoCS. With HC, SoCS outperforms HRG at each level of sparsity. Figure 3.6b provides the ROC curve for the experiment in which $|\mathcal{E}_h| = 1$. This gives more insight on the behavior of the link predictor. HRG is slightly better than SoCS for the very first predictions, and is then outperformed for the remaining ones.

The results obtained on the DBLP dataset, presented Figure 3.7, confirm these observations. We removed 1% of the edges at random , $|\mathcal{E}_h| = 2,110$. Since the graph is larger, we configure SoCS to use 20 dimensions and a larger value of r. We apply two different strategies for SoCS. In the first one, we perform the prediction at a full graph level. This means that we sort the distances between all vertices from the shortest to the longest and predict the links in that order. In the second one, each vertex alternatively predicts a link to the closest social neighbor. The second approach performs better. It is also the one that fits SoCS the best as each vertex makes its own predictions in a distributed manner. This behavior confirms that SoCS generates accurate social coordinates at a local scale. They are usable to compare the distances to social neighbors. Yet, the density of the social space can vary a lot depending on the density of the graph. Thus, it is less accurate to compare two distances from different areas of the social space. Again, SoCS performs best with the HC force model and few repulsions. However, it is always outperformed by CN, and also CP after predicting 60% of the missing edges. Note that at that point, many non existing edges would have been introduced in the graph, so the first predictions are actually the most important. We also believe that the setup of the DBLP experiment favors the CN metric: scientific publications often have more than 2 authors, resulting in many common neighbors for the authors whose link has been removed. We were not able to evaluate HRG on this graph. The implementation provided by the authors quickly runs out of memory on a 16GB memory computer when it's applied to such a large graph.

The experiments show that SoCS can be used for an accurate link prediction. It scales well with the size of the graph, since a small value of r, with respect to the number of vertices, is necessary to obtain the best results. This setting also favors a distributed approach it reduces the network bandwidth usage. On the considered datasets, SoCS is more accurate than HRG, but is outperformed by CN. It is important to understand that computing shortest paths in a large graph is very costly. This is even more problematic when the vertices of the graph are part of a P2P network: it causes a large amount of network communications, and it is very sensitive to churn. When vertices leave or join the graph, the shortest path has to be computer over again. In SoCS, vertices do not compute the list of their common graph neighbors. They just exchange lists of neighboring vertices in the SoCS coordinates space. During our experimentation, we also noticed that SoCS runs faster than HRG, and uses much less memory, which confirms the scalability of our algorithm.

[2] http://dblp.uni-trier.de/xml/dblp.xml.gz

3.4.2 SoCS applied to a small-world network

Setup

In order to evaluate SoCS into details against objective criteria, we experiment the algorithm on a synthetic small-world network. Most of the graphs representing social and semantic relations exhibit a small-world topology. Small-world graphs are characterized by a high clustering coefficient and a small diameter. They have been widely studied from a graph perspective. We rely on the Kleinberg model [52] on the Euclidean plane to generate $1,024$-vertices small-world network from a 2D grid (Figure 3.3). We proceed as follows:

1. We place on a Euclidean plane a square grid of $1,024$ vertices. Each vertex has thus *grid* coordinates ranging from (1,1) to (32,32). We call *grid distance* the Euclidean distance between two vertices in this space.

2. Each vertex is connected with its 8 direct neighbors. These links are called hereafter short links, as they represent links withing communities. In a real social graph, these links represent close friends, who know each other and live in the same city. Vertices at the border of the grid may have less neighbors.

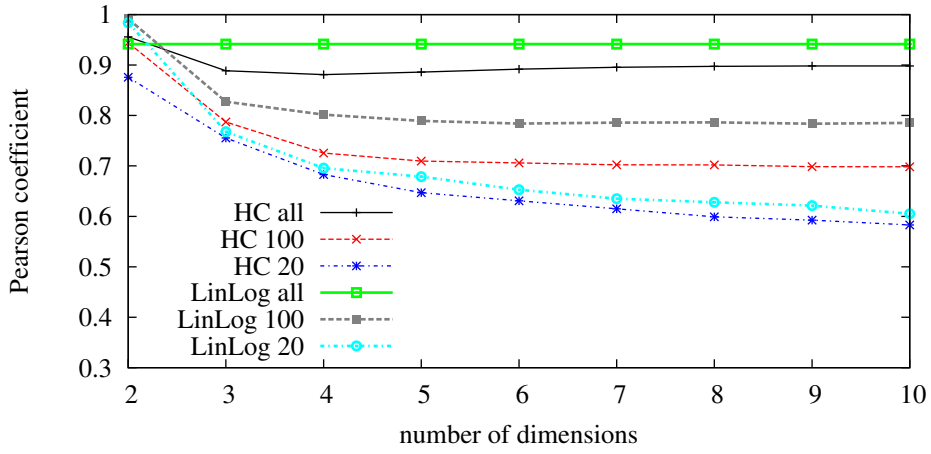
3. Each vertex i also draws one long link to an other vertex. These links represent "unlikely" friends, met on holidays for example. Following Kleinberg's model, the second end of this link, j, is chosen with a probability proportional to $\frac{1}{g(i,j)^2}$, where $g(i,j)$ is the grid distance[3] between i and j. Intuitively, the closer i to j, the higher its probability to be picked as a long link. The distribution of long links is justified by the properties of greedy routing associated, but this is not a property we are using in this chapter.

4. Each vertex has on average 10 neighbors, depending on its position on the grid and how often it is chosen as the end of a long link. For each vertex, we extract this list of neighbors (undirected) and use it as an input for SoCS as depicted Figure 3.4.

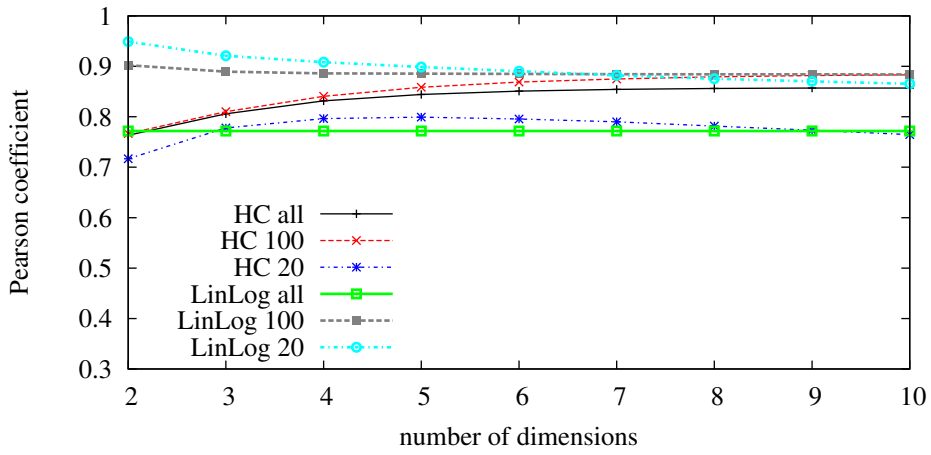
5. SoCS generates a social embedding of the graph (Figure 3.5) and we compare this result with the grid used for generating the graph.

Since the small-world grid used to generate the graph constitutes a detailed ground truth, we are able to perform precise measures to evaluate the embedding performed by SoCS. We are not interested in the absolute value of the coordinates, nor by any scaling or rotation that could occur, as displayed Figure 3.5. Our first experiment consists in computing the Pearson correlation between the social distances and the grid distances. A second experiment consists in using SoCS to allow vertices to differentiate between short and long range links. This is a problem defined by Kleinberg in [53], which was already theoretically addressed [36]. SoCS provides an interesting and more flexible approach to this problem. Each vertex ranks its graph neighbors using the social distances. If SoCS is accurate, all the short links should be shorter than the long links. We evaluate the number of ranking errors as the number of consecutive vertices that need to be switched to obtain a perfect ranking. This can be seen as the number of bubble-sort operations necessary to sort the list. All the results we present are averaged over 100 different generated graphs.

[3] For the sake of uniformity across the experiments, we used an Euclidean distance, whereas [52] used the Manhattan distance. Yet, this does not impact the results.



(a) Global Pearson (all pairs)

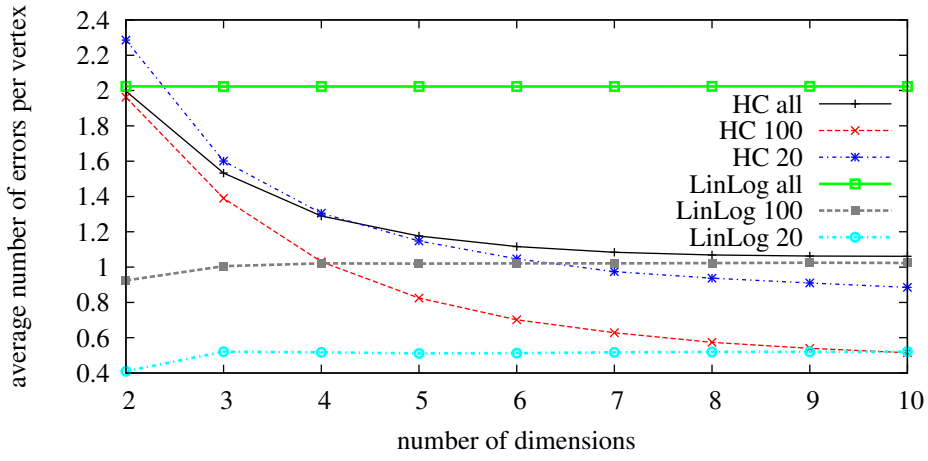


(b) Local Pearson (graph neighbors)

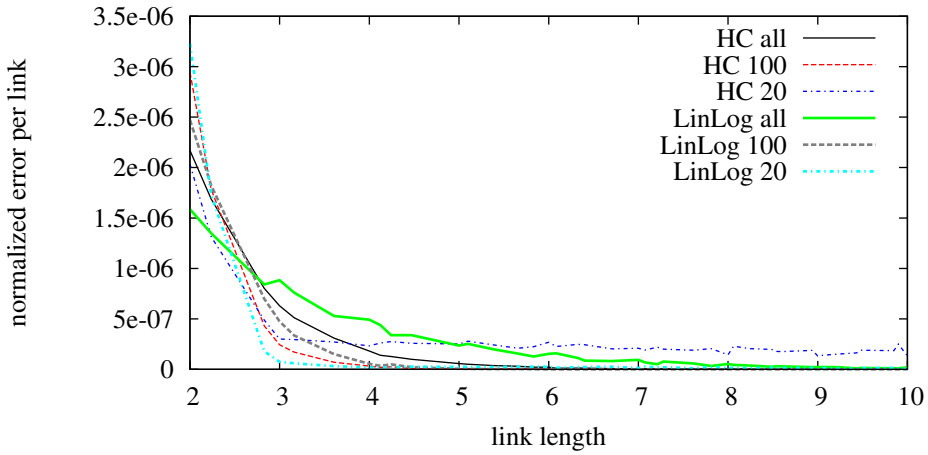
Figure 3.8: Pearson correlation between grid distance and SOCS distance

Static system evaluation

In this set of experiments, we evaluate SOCS on a static graph and wait until the coordinates have converged. We first compute the Pearson correlation between the distance of all the pairs of vertices as seen by g and the distance in the social distances computed by SOCS. The results are presented on Figure 3.8a. Contrary to the previous results, the best performance is obtained with the LinLog force model and a large value of r. The setups in which r is high are very stable, especially in the case of LinLog, and are almost not impacted by the number of dimension. In the other cases, when r decreases, the number of dimensions has a negative impact on the correlation between the social distances and the grid distances. These results contradict the ones obtained in the link prediction experiments in Section 3.4.1. Indeed, link prediction is much more affected by the local positioning of vertices than by the distances between all pairs of vertices. Hence, we define a second measure that we call the local Pearson coefficient. For each vertex, we compute the Pearson correlation by only considering the distances to graph neighbors, and we average this result over all the vertices. This



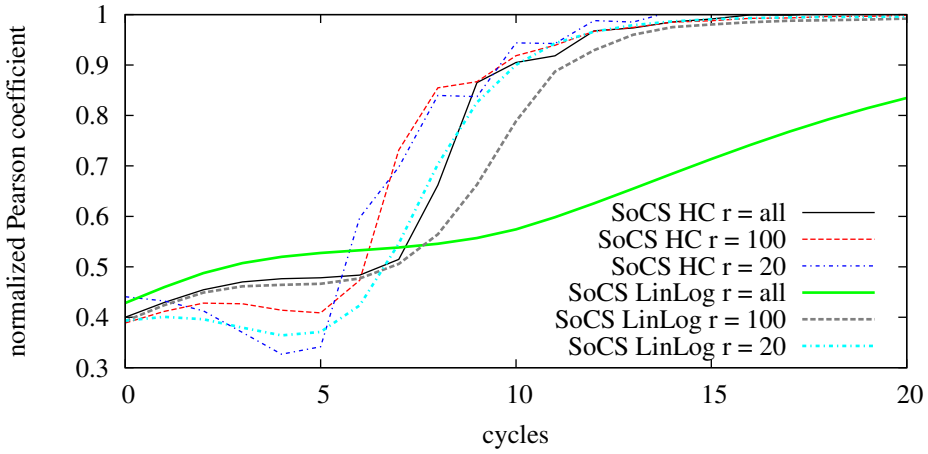
(a) Average number of errors



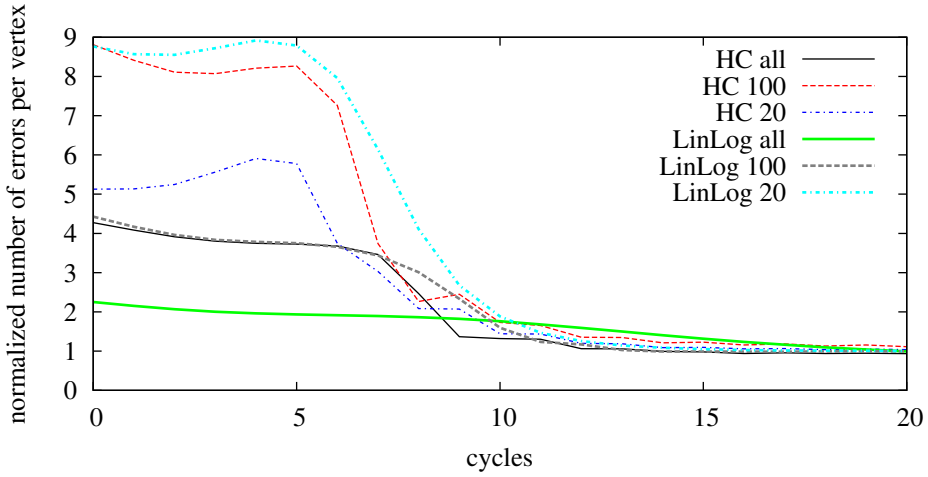
(b) Normalized distribution of the real grid length of misclassified long-range links

Figure 3.9: Links ranking experiment in the static setting

measure reflects the ability of a vertex to differentiate its social neighbors, which relates it to the link prediction problem. As Figure 3.8b shows, the best results are achieved through small values of r. Adding dimensions improves the results with HC, while Linlog performs better with few dimensions. The experiment on the link ranking errors, described in Section 3.4.2, also reflects the quality of the embedding of the close social neighbors. Figure 3.9a presents the average number of ranking errors for each vertex. The configurations that perform well are almost the same: a small r and many dimensions generate fewer errors. LinLog is more stable than HC with $r = 20$, as HC obtains better results with $r = 100$ than with $r = 20$. While the average number of errors per vertex may seem quite high (0.4 in the best case), the classification errors are actually benign. Figure 3.9b gives more insight on the distribution of these errors. It displays the distribution of errors with respect to the grid length of the long links involved in the errors. The number of errors is normalized by the total number of errors of the SoCS configuration, and by the number of such links in the small-world graphs. Clearly, most of the errors are caused by the long links at distance 2. These long links are quite frequent, given the Kleinberg distribution, and sometimes get swapped with diagonal short



(a) Local Pearson



(b) Links ranking errors

Figure 3.10: SOCS in a cold start situation

links, which have a grid distance of $\sqrt{2}$. As these long links are very close according to the grid distance, we do not believe these errors are important. When the long links become more distant, there are almost never involved in errors.

These results confirm our experiments on link prediction: SOCS should be configured with a small r in order to generate an accurate short-range positioning without suffering from noise caused by distant vertices. Taking more repulsions into account generates more accurate long social distances, but it clearly degrades the quality of the close social space representation. Given the applications we target, the close social neighbors are much more important.

Dynamic system evaluation

So far, we have considered a static system. In this section, we consider SOCS in a real P2P environment, where users may join and leave the system (aka churn), creating perturbations. SOCS relies on gossip protocols to maintain the network knowledge of each vertex. These algorithms

have been extensively studied within different churn conditions and have been shown to be very resilient, even in the case of massive failures. In SoCS, the arrival or departure of a vertex modifies the underlying social graph, therefore, the SoCS coordinates should be updated in order to reflect these modifications. In this section, we study the quality of the SoCS coordinates during different churn conditions.

We first consider the case of a "cold start". The $1,024$ vertices of the small-world simultaneously join the network, without running any intermediate SoCS coordinate adjustment. Figure 3.10a and Figure 3.10b display the normalized local Pearson coefficient and the normalized number of links ranking errors respectively. The value 1 is obtained once the system has converged. The HC force model reaches stability much faster (8 cycles), than the LinLog one (up to 40 cycles). We also observe that the convergence time decreases with r.

A massive churn scenario, in which half the network converges and the other half simultaneously joins leads to the same observations. However, the convergence time is faster. Indeed, the vertices that join the network leverage the positions of the vertices already in place and obtain a quite accurate initial placement.

Regular churn, in which vertices join and leave the network at each cycle, has almost not impact on SoCS' precision. When r is low, the perturbations caused by churn are local so they only affect the few vertices that are directly connected to the vertex leaving or joining. They quickly modify their coordinates to take the modification into account.

These experiments show that the HC force model is more adapted to churn than LinLog. This is easily explained by the impact of repulsions: their weight is much lower in the HC model ($fr = -2$), so they generate less perturbations. Taking only the closes vertices into account, i.e. choosing a small r, is crucial for handling churn. It increases the convergence speed of SoCS and makes the system more resilient to churn. The previous experiments show that a small r and the HC force model lead to a better link prediction. Thus, SoCS does not suffer from its distributed nature, link prediction is inherently a distributed application in which each vertex considers its neighbors while ignoring the perturbation from the distant other vertices.

3.5 Existing Work

3.5.1 Graph embedding and drawing

Graph embedding is a mathematical problem that has been receiving a lot of attention for it is a generic problem as explained by Linial, *et al.* [60]:

> Many problems concern either directly or implicitly the distance, or *metric* on the vertices of a graph. It is, therefore, natural to look for connections between such questions and classical geometric structures.

Most of the works in this area extend the results of Bourgain [16] that proved the existence of an embedding of any n-point metric in an $O(\log n)$-dimensional Euclidean space with a logarithmic distortion. They then explore how to exploit these low-distortion mappings to find efficient approximations for various problems such as k-commodity flow [5], or clustering (see Duda and Hart [30], chap. 6). The reader may find in [60] an extensive study of the important results and applications of graph embeddings. We believe that we could use SoCS to adapt many of these algorithms to distributed systems.

Another active and related branch of graph embedding is the graph drawing problem. In this context, the host dimension is always 2 or 3 [40, 54, 83]. The objective is here to produce results that are visually meaningful or to match certain applications requirements such as minimizing the number of crossing when integrating electrical components on a board. Many graph drawing evaluations are purely aesthetic. Since SOCS uses more than 3 dimensions to embed complex graphs, it is difficult to rely on visual observation to determine the quality of the embedding, which is why we rely on link prediction and the comparison with a ground truth in the experiments. However, during the first steps of our experiments on SOCS, we also used images of the layout of small graphs (typically 100 vertices) in 2 dimensions to get an idea of SOCS' performance.

In the remaining of this section, we survey related works in distributed graph embedding, not specifically targeting content recommendation, and works related to content recommendation typically non-linear dimensionality reduction.

3.5.2 Distributing Graph embedding algorithms

To the best of our knowledge, SOCS is the first distributed force-based graph embedding algorithm. However, it is important to consider that some older works closely relate to embedding graphs in a distributed fashion.

As opposed to FBE, another approach to graph embedding relies on the graph spectral properties. The basic idea is to use the Eigenvectors associated to the k largest Eigenvalues of a graph's Laplacian matrix to embed the graph in a k-dimensional host space. This technique was introduced by Hall [41] and is extensively described in [54]. Recently, Kempe and McSherry [49] provided an algorithm to compute the k first Eigenvectors of the adjacency matrix of a graph, based on decentralized orthogonal iteration. The number of steps the algorithm has to run to achieve an error ϵ is $O(\log^2(\frac{n}{\epsilon}) \times \tau_{mix}(\mathcal{G}))$, where $\tau_{mix}(\mathcal{G})$ denotes the graph $\mathcal{G}$ mixing time. The algorithm thus presents attractive costs for fast mixing graphs. However, the application of these works to graph embedding is not straightforward. Indeed, spectral layout sometimes generates representations that facilitate the comprehension of the graph, but the distances between vertices are often difficult to understand. Furthermore, the stability of this algorithm when facing churn or dynamics of the graph has yet to be studied.

Assigning Internet coordinates to vertices in a fully distributed way is a problem that received a lot of attention. Dabek *et al.* proposed Vivaldi [28], a distributed protocol that aims at predicting Round Trip Times (RTT) between machines on the Internet. The system is modeled as a valued graph where machines are vertices and edges are measured RTT values. In a nutshell, a distributed spring embedding algorithm is used to embed the graph in a low (2-3) dimensional space. The idea is that even though each Internet machine maintains an up-to-date RTT to some Internet machines, the vertex's distance in the host space efficiently approximates the RTT between them. The main difference with SOCS is that the embedding aims to be as isometric as possible. Furthermore, in Vivaldi, machines can easily compute, through a ping measure, their optimal distance to any given machine and adjust their position accordingly. In SOCS, the vertices only have information about their graph neighbors and cannot do such a measure with respect to any vertex in the graph. Finally, the social networks we consider in SOCS can be arbitrarily complex and unpredictable. For example, the long links in the social networks break the community clustering and add some difficult constraints to the layout. In the case of Vivaldi, the RTT between machines is bound to the physical reality of the network, therefore the structure of the graph is much more simple.

3.5.3 Non linear dimensionality reduction

Another class of works that relate to SoCS is dimensionality reduction (DR). DR consists in discovering close items from a of set items described by highly dimensional data. These methods are heavily used in machine learning and recommendation systems. Finding relationships between the describing variables that are not linearly correlated led people to consider variables correlated in a manifold (such as the distances between cities of the Earth globe). The goal of the non-linear DR algorithm is to flatten this data into a Euclidean space (a world map). The approach followed by these methods (*e.g.* KPCA, LMDS [23], LLE [74]) is to achieve DR while preserving the "local structure" of the data.

Recently, Chen and Buja [23] pointed out the close link between graph drawing and non linear DR. Their conclusion is that graph drawing, under certain circumstances, is achieving DR (or proximity analysis). This allows to see SoCS as a non linear DR algorithm that "flattens" the social manifold. Moreover, the importance of locality in non-linear DR fits with the distributed implementation of SoCS.

3.6 Concluding Remarks

In this chapter, we presented SoCS, a fully distributed algorithm for social graph embedding. SoCS distributes traditional force-based embedding algorithms to embed social graphs while preserving their community structure [67]. SoCS exploits the benefits of the community structure knowledge for link prediction [25] to achieve meaningful recommendations. SoCS relies on gossip protocols to approximate the forces applied to the vertices and achieve scalability in large and dynamic graphs.

We extensively analyzed the performance of SoCS on both synthetic and real data. Our experiments show that SoCS is able to accurately assign social coordinates to vertices. The configurations that obtain the best results are the ones that favor a precise representation of the local social neighborhood to global accuracy. Not only does this setting improve the quality of the results, it also offers better scalability and more resilience to large-scale network perturbations such as churn.

We believe that link prediction is one of the most important features in social networks. It is crucial to the development of the social graph. It is also one of the most difficult to achieve in a distributed environment. Indeed, in social networks, most of the activities involve interactions between friends. Link prediction is one of the only cases in which it is imperative to explore the social graph in order to predict new connections. Our results are very encouraging, as they show that a P2P social network could be implemented without sacrificing this functionality.

Future work includes evaluating the impact of other force models in order to find the best trade-off between accuracy and applicability in P2P systems, as well as achieving a better understanding of the algorithm parameter space. We also would like to study further the privacy-protecting properties of SoCS, and the applications of onion routing [80] in its context.

CHAPTER 4

DISTRIBUTED PERSONALIZED WEB SEARCH

Abstract

In this chapter, we propose a decentralized approach for social applications. As social information can be used to accurately predict social information, we consider the case of a P2P architecture, in which each user keeps control over her data. We present GOSSPLE [50] Multi-Interest Network (GMIN), a distributed algorithm that provides each user with a set of personalized acquaintances that share her interests. GMIN relies on a *set cosine similarity* to select neighbors which cover all the interests of the user. GMIN discovers these neighbors through a *gossip-on-behalf* protocol, which protects the privacy of the users through the use of onion routing techniques. We use the neighbors provided by GMIN in a Web search query expansion application named GQE. We show that, by taking into account the user's personal view of the relations between tags, it is possible to improve the quality of the results. We experiment our system on a wide variety of datasets to demonstrate the benefits achieved.

This work, presented at the Middleware 2010 conference [13], was done in collaboration with Marin Bertier, Davide Frey, Anne-Marie Kermarrec and Rachid Guerraoui. A companion paper [7], not presented in this thesis, describes the decentralization of the search process. This work is supported by the ERC Starting Grant GOSSPLE number 204742.

4.1 Introduction

Context

Personalized Internet services have radically changed the way people navigate in the Internet. Websites, such as LastFM[1], Flickr[2], CiteULike[3] and Delicious[4], contain large amounts of user generated social information. They leverage it to recommend content to the users, which greatly facilitates their discovery of new interesting content.

Yet, this social knowledge can also be used to threaten the privacy of the users. As we show in Chapter 2, it is possible to accurately predict the social network of a user from her social information. Social information is also often used for personalized advertising purposes. Therefore, users might become reluctant to expose their interests and share their knowledge. Since the efficiency of social applications is directly related to the amount of social knowledge they gather, this would clearly affect their quality. Some sensitive topics, may not receive any more contributions.

Furthermore, these centralized websites are limited by their processing power. The social knowledge they contain is mostly unstructured. The users express their interests in *items* through freely chosen keywords, also called *tags*, forming a *folksonomy*. While the freedom left to the users is very appealing and can lead to the emergence of unexpected and interesting knowledge, it is also an impediment to the navigation. Users can express the same idea through different tags, and do not always assign the same meaning to a given tag. The processing power required to extract all this information and fully personalize the experience of the users necessitates extremely costly data centers, which can be out of reach for many Web services. Hence, they approximate the recommendations and do not exploit the full extent of the personalization possibilities.

In this chapter, we tackle the privacy and scalability issues by proposing a P2P framework for social applications. As argued in Chapter 3, this improves the privacy of the users, as they keep control over their personal information. It also increases the scalability of the system, since users contribute to the system with their own machine. Each user is associated with a network of *anonymous acquaintances* that are interested in similar items, independently of the vocabulary they use to express their interests (i.e. which keywords they use to tag items). These acquaintances can then be leveraged in personalized Web search applications.

Capturing associations between millions of users is challenging as the amount of information available grows not only with the number of users, but also with the size of the users' profiles. The acquaintances assigned to a user should provide a significant amount of information in order to enhance her search experience, but without drowning her within tons of useless data. They have to reflect the whole range of her interests, even minor ones, in order to provide information for a wide range of queries. The system also has to adapt to the dynamics of the network to account for users joining and leaving the system, but also to match the evolution of the users' profiles. Finally, the system has to protect the privacy of the users: their interests should not be exposed.

[1] http://www.last.fm
[2] http://www.flickr.com
[3] http://www.citeulike.org
[4] http://www.delicious.com

Contributions

We introduce GOSSPLE [50] Multi-Interest Network (GMIN), a system that personalizes the Web experience of the users by automatically inferring interest-based connections between them. We then leverage the neighbors discovered by GMIN in a personalized query expansion application called GQE.

We devise a distributed protocol to detect and select acquaintances on a distributed manner. GMIN users continuously gossip digests of their tagging profiles and locally compute a personalized *view* of the network, which is then leveraged to improve their Web navigation.

This protocol detects proximity between users' profiles without violating anonymity: the association between users and profiles is hidden. Basically, every GMIN user has a proxy, chosen randomly, gossiping her profile digest *on her behalf*. The user transmits her profile to her proxy in an encrypted manner through intermediary users, which cannot decrypt the profile.

To reduce bandwidth consumption, the gossip exchange procedure is thrifty: users do not exchange their full profiles but only Bloom filters of those until similarity computation reveals that the two users might indeed benefit from the exchange.

Our gossip exchange procedure is able to build a GMIN overlay quickly, even starting from a completely random configuration. Additionally, the bandwidth consumption remains low and the anonymity of the users is preserved through the use of onion-routing techniques.

To limit the number of profiles maintained by each user, while encompassing the various interests of the user, we introduce a new similarity metric, we call the *set cosine similarity*, as a generalization of the classical cosine similarity metric [19, 88], as well as an effective heuristic to compute this new metric. The views generated by GMIN cover multiple interests without any explicit support (such as explicit social links or ontology).

GMIN can directly be used as a recommendation and search systems, but it can also provide information to more sophisticated algorithms. We illustrate the effectiveness of GMIN through GQE, a query expansion application for collaborative tagging systems (folksonomies). We compute scores between tags using the acquaintances provided by GMIN and adapt rooted PageRank to leverage the relative centrality of the tags through an algorithm we call GRANK. We evaluate our query expansion application independently and show how we retrieve items that state-of-the-art search systems do not (recall, also called completeness) whilst improving precision (accuracy) at the same time. We show, among other things, how GQE retrieves personalized results, even when they contradict what a majority of the other users would have expected.

We evaluated our system with a wide-range of Web application traces, including Delicious, CiteULike, LastFM and eDonkey through simulation and in a real distributed network (PlanetLab). Our results clearly show the benefits achieved by the GMIN personalized network. The algorithm we use for query expansion requires a lot of computation, which would be prohibitive in a centralized system. This illustrates how P2P social applications can increase the quality of the Web search of the users, while also protecting their privacy.

Roadmap

We detail the GMIN protocol in Section 4.2 and evaluate its performance in Section 4.3. In Section 4.4, we describe how GQE leverages GMIN in a query expansion application. In Section 4.5, we survey the existing work. Finally, in Section 4.6, we conclude and highlight some of the limitations of our system.

4.2 GMIN Protocol

GMIN is a fully decentralized protocol aimed at building and maintaining dynamic communities of anonymous acquaintances. Such communities differ from networks of *explicitly* declared friends (e.g. Facebook) which are often not the most suited to enhance the search procedure as reported in [11].

4.2.1 Overview

More specifically, GMIN provides each user with GNET, a network of semantically close anonymous interest profiles. Building and maintaining a user's GNET goes first through determining a metric to identify which profiles exhibit similar interests. This is particularly challenging for the goal is to capture the whole range of a user's interests while limiting the number of profiles in the GNET for scalability and personalization reasons. Another challenge is to devise an effective communication procedure that limits the amount of network traffic generated upon profile exchange and hides the association between users and profiles to preserve anonymity.

We detail in the following how GMIN addresses these challenges. We assume, for presentation simplicity, a one-to-one mapping between a user and a machine: we refer to a user in the remaining of this chapter. From an abstract perspective, we model the profile of a user as a set of items $I = \{i_1, \ldots, i_n\}$. Each item can be described by a potentially large amount of meta-information, such as tags it in the case of a folksonomy. Depending on the target application, this set may represent downloaded files (eDonkey), URLs (Delicious and CiteULike), artists (LastFM), etc. In the following, we first present a novel multi-interest (set item cosine similarity) metric; then, we describe our thrifty gossip-based protocol to establish and maintain connections between users (i.e., compute GNET) using this metric. For presentation simplicity, we describe our protocol in a modular manner: we present first how a user communicates with other users to review a large set of profiles, how it encodes a profile, and finally how it preserves its anonymity.

4.2.2 Selecting acquaintances

In order to improve her Web search experience, a user has to leverage the knowledge of other users whose interests are similar. There are many possible definitions of similarity between users. In the link prediction work applied to Flickr (Section 2.4), we presented many features that leveraged the users' group membership to determine their similarity and predict friendship. In this context, we rely on the profiles of the users, which are lists of items they are interested in. Thus, the metrics we describe in this section will be used by the GMIN users to rate other users based on their profiles and discover relevant acquaintances.

Rating individuals

One way to build a user's GNET is to individually rate other users and select the ones that score the highest. Cosine similarity [61] is widely used in the area of data mining. The score between two users increases when interests are similar and specific overlapping of interests are favored over large profiles. Our preliminary experiments have shown indeed that cosine similarity outperforms simple measures such as the number of items in common. Thus we implemented cosine similarity as a reference in our experiments. More specifically, let I_n be the set of items in the profile of user u,

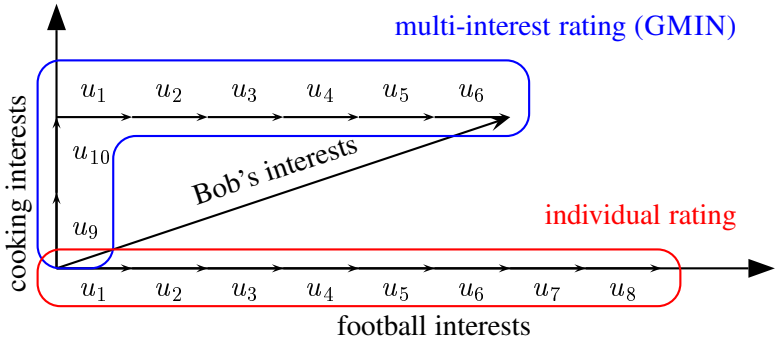


Figure 4.1: Multi-interest rating *vs.* Individual rating

the item cosine similarity between two users u and v is defined as follows:

$$ItemCos(u,v) = \cos(I_u, I_v) = \frac{|I_u \cap I_v|}{\sqrt{|I_u| \times |I_v|}} \ .$$

Individual rating, and thus cosine similarity, selects users having the most similar profiles. Yet, this may lead to consider only the dominant interest, ignoring minor, potentially important, ones. In a large-scale system where a user can only keep track of a limited number of profiles, individual rating cannot capture emerging interests until they represent an important proportion of the profile, which they might never. This is a critical issue since the moment a user discovers a new topic also is the moment where it is the most important to improve her search.

Consider Bob whose tagging actions reveal that 75% of his interests are in football while 25% are in cooking. Selecting the profiles of the closest users might lead to only selecting those interested in football: the cooking topic will not be represented enough to provide interesting information. In Figure 4.1, the individual rating selects a view of 8 users interested only in football (users u_1, u_2, ... u_8). On the contrary, the GNET should preserve the distribution of 75% of football and 25% of cooking. We achieve this by rating a set of profiles as a whole rather than each profile independently. This is crucial to achieve scalability and limit the size of the GNET with respect to the overall size of the network. In the example, multi-interest scoring would lead to selecting 2 users interested in cooking: u_9 and u_{10} and 6 in football, covering Bob's interest in cooking as well.

Rating sets

Rating a set of profiles as a whole involves a balance between the extent to which users in the set share interests with a given user u, and how well the distribution of the interests in the set matches the one in u's profile. This is precisely what the GMIN *item set cosine similarity* metric achieves.

Let $IVect_u$ be the vector that represents the items in the profile of the user u. $IVect_u[i] = 1$ if item i is in the profile of u, 0 otherwise. Following the same principle, $SetIVect_u(s)$ builds an item vector that represents the distribution of items in s with respect to the user u where s is a set of users. Each value of this vector is computed as follows:

$$SetIVect_u(s)[i] = IVect_u[i] \times \sum_{v \in s} \frac{IVect_v[i]}{\| \ IVect_v \ \|} \ .$$

The rationale behind this metric is to represent the distribution of interests in GNET while normalizing the contribution of each user to favor specific interests. The items that are not present in the profile of user u are discarded since their distribution should not impact the score of GNET. Indeed, adding two identical items to $SetIVect_n$ does not impact the norm of the vector the same way as two different ones ($||[1,1]|| = \sqrt{2}$ while $||[2,0]|| = 1$). If n is not interested in any of those two items, there is no reason to prefer one choice over the other, which is why they are discarded.

Following the cosine similarity between users, a user u computes a score for a set of users as follows:

$$SetScore_u(s) = (IVect_u \cdot SetIVect_u(s)) \times \cos(IVect_u, SetIVect_u(s))^b \ .$$

The first part of the formula sums all the values in the vector representing the distribution of items in s, while the second represents how well the distribution of the items in s matches the one in n's profile, i.e. how fair the contribution to all the interests of n is. The parameter b is the *balance* between the amount of shared interests and a fair distribution that does not favor any item. The set that scores the highest forms the user's GNET. It is important to notice that for $b = 0$ (no cosine impact), the distribution is not considered and the resulting GNET is exactly the same as the one obtained from the individual rating:

$$\begin{aligned}
SetScore_u(s) &= (IVect_u \cdot SetIVect_u(s)) \times \cos(IVect_u, SetIVect_u(s))^0 \\
&= IVect_u \cdot SetIVect_u(s) \\
&= \sum_{v \in s} \frac{IVect_v \cdot IVect_u}{\parallel IVect_v \parallel} \\
&= \sum_{v \in s} \frac{|I_u \cap I_v|}{\sqrt{|I_v|}} \\
&= \sqrt{|I_u|} \sum_{v \in s} \frac{|I_u \cap I_v|}{\sqrt{|I_u| \times |I_v|}} \\
&= \sqrt{|I_u|} \sum_{v \in s} ItemCos(u, v) \ .
\end{aligned}$$

Since $\sqrt{|I_u|}$ does not depend on s, maximizing $SetScore$ is exactly the same as selecting the users that individually maximize $ItemCos$ when $b = 0$.

4.2.3 Discovering acquaintances

GMIN's multi-interest metric is key to selecting the best set of profiles to fill a given user's GNET. This would however be of little use without an effective mechanism to review a large set of candidate profiles while ensuring rapid convergence to the ideal GNET. GMIN achieves this through a gossip protocol to establish and maintain connections between users. For the sake of clarity, we first present the non-anonymous variant of this protocol and describe Section 4.2.5 how it preserves anonymity.

The GMIN protocol relies on two sub-protocols: a Random Peer Sampling protocol (RPS) and a multi-interest clustering protocol (GNET protocol). Each user maintains two corresponding data structures, a *random view* and the GNET. Gossip protocols are presented in Section 3.2.3. GMIN is built along the lines of the skeleton described in Algorithm 3.1.

RPS

As already explained, a RPS [46] provides each user with a random subset of the network. Each entry in the random view contains:

- the IP address and the GMIN ID of the user
- a digest of its profile in the form of a Bloom filter (discussed in Section 4.2.4)
- the number of items in the profile (used for the normalization during cosine computation, as a Bloom filter does not provide the cardinality of the set).

GMIN relies on Brahms [15], a byzantine resilient RPS, which provides the building blocks to build GMIN's anonymity (presented in Section 4.2.5).

GNet protocol

GNET contains c entries composed of the same fields as the entries in the random view. In addition, each entry also contains the full profile of users that have been chosen as acquaintances. The parameter c selects the trade-off between the amount of available information and its personalization degree. A large view provides results averaging the whole system's information, while a small view provides information closer to the user's opinion. We denote by $GNet_u$ the GNET of a user u. Note that a user in $GNet_u$ shares at least one item (interest) with u.

Although the GNET protocol is built along the lines of clustering protocols [84] and on top of the gossip skeleton, for simplicity reasons, we provide a full description in Algorithm 4.1. The GNET protocol maintains a list of (implicit) acquaintances. Upon receiving the other user's GNET, u and v update their own GNET structure. They first compute the union of $GNet_u$, $GNet_v$ and their own RPS view, then they select the c users that maximize the GMIN metric described in Section 4.2.2. Selecting the best view consists of computing the score of all possible combinations of c users out of $3c$ users. Because the corresponding running time is $O\left(\binom{3c}{c}\right)$, our protocol uses a heuristic (Algorithm 4.2) as an approximation. It incrementally builds a view of size c from an empty view by adding at each step the user that provides the best view score. This heuristic has complexity $O(c^2)$ and turns out to be very appropriate in our evaluations.

Algorithm 4.1 GNet protocol (u gossips with v)

```
1:  loop                                              ▷ Do one cycle every T time units
2:      wait T time units
3:      v ← oldest entry in GNet_u
4:      send GNet_u ∪ {makeEntry(u)} to v             ▷ Exchange the GNET information
5:      receive GNet_v from v
6:      GNet_u ← ComputeView(GNet_u ∪ GNet_v ∪ RPS_u)          ▷ Update the view
7:      for entry ∈ GNet_u do          ▷ Fetch profiles of the users that stay in GNET K cycles
8:          if entry ∈ GNet_u during K cycles & entry.profile = ∅ then
9:              entry.profile ←fetchProfile(entry.user)
10:         end if
11:     end for
12: end loop
```

Algorithm 4.2 Scoring heuristic

```
1: input candidateSet                                          ▷ Set of entries
2: input viewSize                                              ▷ Maximum size of the view
3: if |candidateSet| ≤ viewSize then                           ▷ Trivial case
4:     return candidateSet
5: end if
6: bestView ← ∅
7: for setSize = 1 to viewSize do                              ▷ Incrementally build a view
8:     bestScore ← −∞
9:     bestCandidate ← ∅
10:    for candidate ∈ candidateSet do                         ▷ Consider all remaining users
11:        candidateView ← bestView ∪ {candidate}
12:        viewScore ← SetScore(candidateView)
13:        if viewScore > bestScore then
14:            bestCandidate ← candidate
15:        end if
16:    end for
17:                ▷ Keep the entry that got the best score and remove it from the candidates
18:    bestView ← bestView ∪ {bestCandidate}
19:    candidateSet ← candidateSet − {bestCandidate}
20: end for
21: return bestView
```

4.2.4 Gossiping digests

In order to keep the bandwidth consumption of GMIN within reasonable bounds, users do not send their full profiles during gossip. Instead, they exchange a Bloom filter [14]: a compact representation of the set of items in the profile. The Bloom filter provides a reasonably good approximation of a user's profile that can be used to compute GMIN's metric with a negligible error margin.

The Bloom filter, as depicted in Figure 4.2, is an array of bits representing a set. When an item is added to the set, h hash functions are used on the item to obtain h positions in the array: these are set to 1. In order to query the presence of an item in the set, one uses the same hash functions and checks if all the bits at the h indexes are set to 1. In the case of several additions to the Bloom filter, the request can return true even if the item was never added to the set. The rate of false positive depends on the size of the set, h, and the number of items added to the Bloom filter. However, a Bloom filter never returns a false negative.

In GMIN, the Bloom filter is used as a first approximation of a user's profile. If a user remains in the GNET for K gossip rounds, it is considered as a good candidate and the entire profile is requested ($K = 5$ in our experiments), as explained in Algorithm 4.1 lines 8–10. Once the full profile of the user has been retrieved, it is used to compute an exact similarity score. This prevents the expensive transfers of useless entire profiles of users that will reveal distant according to the GMIN metric. For example, the profile of a Delicious user is on average 12.9KB large, while the corresponding Bloom filter is only 603B. This leads to a 20-fold improvement in bandwidth usage, as discussed in Section 4.3.4.

Since a Bloom filter can return false positive results, some users considered as acquaintances

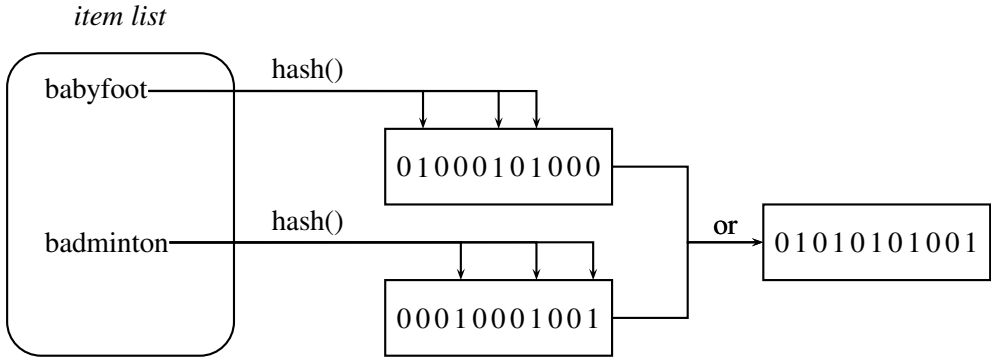


Figure 4.2: The Bloom filter of a profile

through their Bloom filters can be discarded when the exact score is computed with the profile. Nevertheless, a user that should be in the GNET cannot be discarded due to a Bloom filter approximation. Hence, an entry for a user in the GNET contains either the full profile of the user or a Bloom filter of the profile with a counter incremented at each cycle. When the counter reaches the value of K, it triggers the retrieval of the corresponding full profile.

4.2.5 Preserving anonymity

Protocol modifications

Anonymity is crucial in applications where personal interests may be exposed. The decentralized nature of GMIN somehow inherently preserves some level of anonymity, as compared to a central entity which would control and store all personal data. We go one step further by observing that, while personalized applications may benefit from the profile information contained in GNETS, they do not need to need know which users are associated with which profiles. What matters is that a user can leverage other users' information, but she doesn't need to know who provided this information. This observation underlies our *gossip-on-behalf* approach: each user u is associated with a proxy p that gossips on her behalf. Since P2P networks are subject to churn, it is possible that p disconnects, either willingly or accidentally, while u is still online. In order to avoid losing the clustering information, p periodically sends snapshots of $GNet_u$ to u. Thus, u can resume the gossip protocol on a new proxy without losing any information, the convergence state remains the same. This *anonymity by proxy* setup is achieved by an encrypted two-hop communication path similar to the ones used in onion routing [80]. Therefore p receives u's profile and its updates but does not know u's identity while the users relaying the communication cannot decrypt the profile. GMIN relies on the Byzantine resilience properties of the RPS protocol to prevent colluders from manipulating the selection of the proxy and the relays. Furthermore, we assume that the system is protected against Sybil attacks through a certificate mechanism, a challenge system at join time, or a detection algorithm [89]. Finally, we also consider that it is a user's responsibility to avoid adding very sensitive information to her profile. In that case, the profile alone would be sufficient to find the identity of the user as it was the case in the famous anonymized AOL query-log dataset.

Since GMIN's privacy guaranties are purely related to the strength of onion routing, we do not detail the attacks scenario in this thesis. The reader may refer to [79] for additional information on

this matter. GMIN ensures anonymity deterministically against single adversary users and with high probability against small colluding groups.

Anonymity overhead

The use of onion routing in GMIN generates some additional messages which increase the network cost of the protocol. However, this overhead is not comparable to the one usually observed in onion routing. Indeed, in onion routing, all messages are sent from the user u to the destinations through several other users. In GMIN, p runs the gossip protocols (GNET and RPS) on u's behalf, without communicating with u at each step of the gossip. Hence, there is absolutely no overhead on the gossip, besides the snapshots that p regularly sends to u. The proxy p sends the profiles of the users in $GNet_u$ once they have been downloaded after K cycles. Compared to a setup without anonymity, the overhead on the profile transmission is directly proportional to the length of the onion routing path. Since GMIN waits for the GNET to stabilize before downloading profiles, the total number of profiles downloaded by u is limited and close to c.

To summarize, GMIN's anonymity protocol generates some overhead when a new user joins the system and her GNET converges. She downloads profiles to acquire information and this data is routed through several users. During the rest of the execution, the only data transmitted through the onion routing is related to profile updates and snapshots. This information is much smaller than full profiles, so the anonymity protocol remains lightweight.

4.3 GMIN Evaluation

We report on the effectiveness and cost of GMIN by means of both simulation ($100,000$ users) and PlanetLab deployment (446 users). The former provides insights about GMIN's behavior with a very large number of participants while the latter measures the inherent overhead of GMIN in a real distributed setting.

4.3.1 Setting and methodology

We evaluated the quality of GMIN's GNET, its convergence speed, as well as its bandwidth consumption and behavior under churn. We considered various Web datasets:

- a trace crawled from Delicious (social URL bookmarking) in January 2009
- a trace from CiteULike (reference management system) available on October 9, 2008
- a LastFM (music recommender system) trace crawled in Spring 2008 and composed of the 50 most listened-to artists for each user
- an eDonkey (P2P file-sharing) trace [42].

Table 4.1 provides the figures for each trace. While the simulation lets us experiment GMIN on a large number of users, PlanetLab is much more limited. We have to select a subset of our dataset in order to create profiles for the users on PlanetLab. Sampling the users at random is not a good option as it emphasis the sparsity of the dataset. Users do not form communities anymore and cannot find enough acquaintances to create their GNETS. Thus, in order to preserve the density of the dataset,

	Delicious	CiteULike	LastFM	eDonkey
Number of users	$130k$	$34k$	$1,219k$	$187k$
Number of items	$9,107k$	$1,134k$	$964k$	$9,694k$
Number of tags	$2,214k$	$237k$		
Average profile size	224	39	50	142
Number of users	$50k$	$10k$	$100k$	$100k$
Recall $b = 0$	12.7%	33.6%	49.6%	30.9%
Recall GMIN	21.6%	46.3%	57.6%	43.4%

Table 4.1: Dataset properties

we first create a GMIN network in the simulator using the whole dataset, and then sample a subset of this network through snowball sampling, as performed on the Flickr dataset (Section 2.3.1). The only difference is that instead of following friendship links, we use the GNETS to discover new users.

We configured GMIN's GNETS size to 10 and the gossip cycle length to 10 seconds. We deliberately select a small *GNet* size in order to stress the system and highlight the impact of the selection of each acquaintance.

In this section, we evaluate the quality of a user's GNET through its ability to provide the user with interesting items. This measures the basic usefulness of GMIN's GNETS in search applications and recommender systems. We remove a subset (10%) of the items in the profile of each user, which we call the *hidden interests* of the user. GMIN uses the remaining part of the profile to build the GNETS. We express the quality of a GNET in terms of *recall*, i.e. the proportion of hidden interests of a user u that are present in the profile of at least one user in u's GNET. The better the GNET, the more hidden interests it contains. Each hidden interest is present in at least one profile within the full network: the maximum recall is always 1. Recall is enough to evaluate quality here because u contacts a fixed number of users. We further evaluate precision in Section 4.4.

We evaluate the overall quality by computing the recall for the whole system: we divide the sum of the number of hidden interests retrieved by GMIN for each user and divide it by the sum of the number of hidden interests of each user.

4.3.2 Metric (the quality of GNets)

Our item set cosine similarity (multi-interest) metric enables GMIN to cover the diversity of users' interests in a large scale system by assessing the quality of the whole GNET at once, instead of rating profiles individually. Parameter b in the definition of $SetScore$ (introduced in Section 4.2.2) represents the weight of minor interests with respect to major ones. We evaluate the impact of b using the hidden interests test described above. Intuitively, too small a value of b achieves poor results because the GNET fails to provide items related to minor interests. On the other hand, if b is too high, users focus too much on the fairness of the interest distribution and select profiles with too little in common with the user to provide relevant items. Figure 4.3 presents the normalized recall achieved by GNETS on the datasets with increasing values of b. The score is normalized to take as a reference the case when $b = 0$, equivalent to individual rating. As expected, the performance initially increases with increasing values of b, but it decreases for greater values of b. Multi-interest improves recall from 17% (LastFM) to 69% (Delicious). For all datasets, GMIN significantly improves the quality of GNETS over traditional clustering algorithms illustrated by the $b = 0$ configurations. The

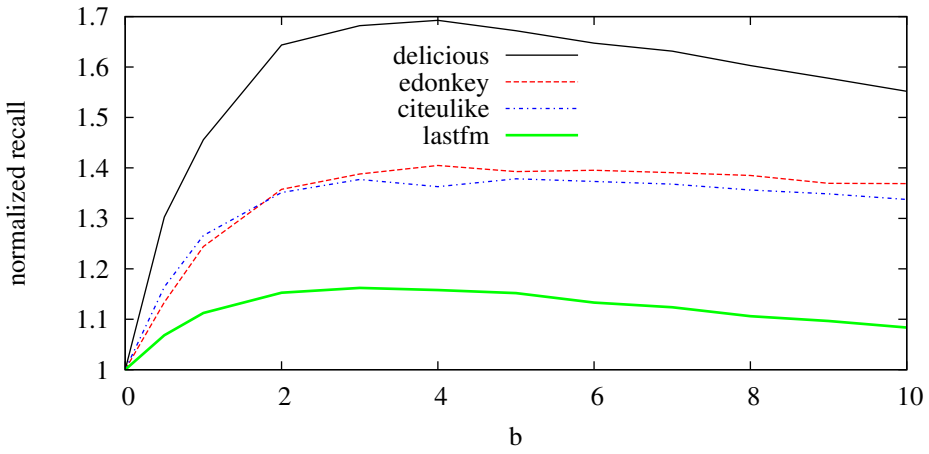


Figure 4.3: Impact of b on normalized recall

exact recall values are given in Table 4.1. We also observe that the improvement of multi-interest has more impact when the base recall is low. Further experiments reveal that this is because GMIN is particularly good at finding rare items. Finally, we observe that the maximum performance is not obtained for a specific value of b, but for a full range of values, $b \in [2, 6]$, across all datasets. This demonstrates GMIN's effectiveness on a wide variety of datasets without requiring prior fine tuning of parameters.

4.3.3 Convergence time

We evaluate GMIN's ability to build and maintain a useful GNET for each user. First, we consider the time required to build a network of GMIN acquaintances from empty GNETS. Then we consider the maintenance of this network by evaluating convergence in a dynamic scenario where users join an existing stable network.

Bootstrapping

We consider a simulation of $50,000$ users and a real-world PlanetLab deployment with 446 users on 223 machines. Figure 4.4 plots the hidden interests' recall (as defined in Section 4.3.2) during the construction of the GMIN network. It is normalized by the value obtained by GMIN at a fully converged state.

As expected, the multi-interest clustering ($b = 4$) constantly outperforms the standard one ($b = 0$), although the former requires slightly longer to reach a stable state. The difference is minimal, GMIN reaches 90% of its potential after only 14 gossip cycles in simulation in our Delicious traces for instance. This convergence is extremely fast given the size of the network ($50,000$) and the small size of the GNET and RPS (10). The measures conducted on LastFM assess the scalability of the protocol: for twice as large a network, only 3 more cycles are needed to reach the same convergence state. The PlanetLab deployment confirms the simulation results. The smaller scale of the experiments causes GMIN's GNETS to reach 90% of their potential after an average of 12 cycles and stabilize after 30. This confirms the scalability of our protocol by demonstrating convergence times that grow very slowly with the size of the network.

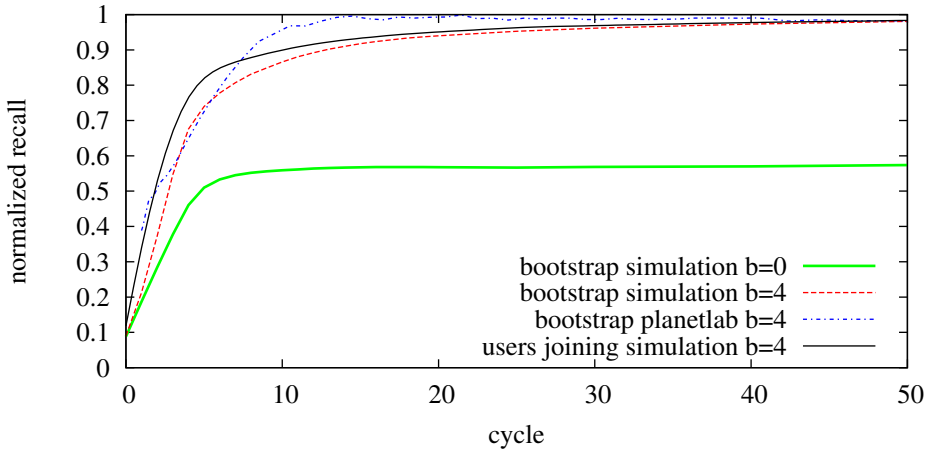


Figure 4.4: Recall during churn

Maintenance

Bootstrapping represents a worst-case scenario for the convergence of our protocol. It is, in fact, a one-time event in the life of a GMIN network. During normal operation, the system experiences instead perturbations that cause it to deviate from a stable state in which each user has converged to her best possible GNET. Examples of such perturbation include the presence of users that join and leave the network (churn), but also variations in the interests of users. When a user adds or removes items from her profiles, her optimal GNET can be modified.

To evaluate the impact of these perturbations, we consider a scenario where 1% of new users join an existing GMIN network at each gossip cycle. We measure hidden interests recall of these new users to see how many gossip cycles are needed for them to reach a quality of GNET equivalent to the one of the previously converged users (i.e. 1 on the normalized score). Joining an existing network is indeed faster than bootstrapping as 9 cycles are enough to reach a 90%-quality GNET. Clearly, this represents an upper bound on the time required for convergence in the presence of dynamic profiles or users that leave the network. In both of these cases, users that are required to converge only need to partially reconstruct their GNETS as they already have some good neighbors to rely on. Moreover, the removal of disconnected users from the network is automatically handled by the clustering protocol through the selection of the oldest peer from the view during gossip exchanges as discussed in detail in [84].

4.3.4 Bandwidth

The bandwidth consumption of GMIN when the GNETS are built from scratch is depicted in Figure 4.5. This cost is the result of:

- the exchange of profile digests upon gossip
- the download of the full profiles of users in the GNET
- the extra communications required to maintain anonymity.

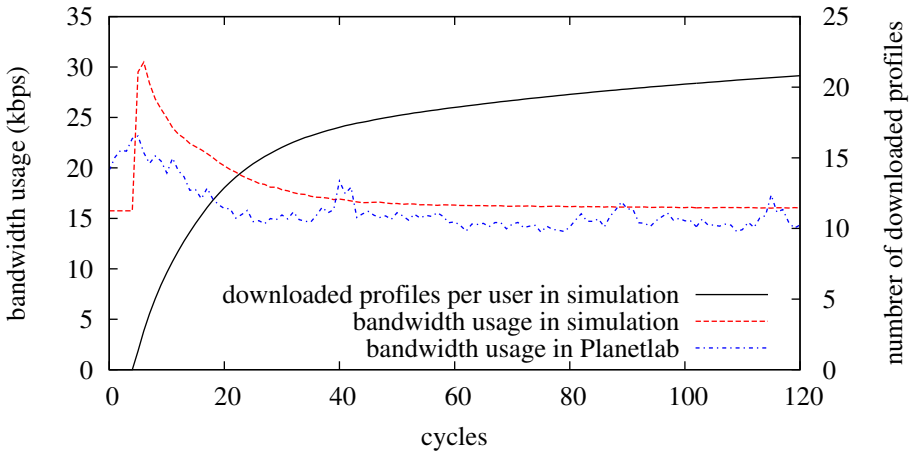


Figure 4.5: Bandwidth usage at cold-start

Each user gossips through the RPS and the GNET protocol every 10 seconds. Each message contains respectively 5 and 10 digests. This results in a maximum bandwidth consumption of 30Kbps in the most demanding situation, that is during the burst at the beginning of the experiment. This is because no profile information has yet been downloaded[5]. As soon as the GNETS start to converge, the rate of exchange of full profiles decreases, as shown by the line depicting the total number of downloaded profiles per user. This causes bandwidth consumption to decrease to the fixed cost of gossiping digests, 15Kbps, a clearly irrelevant value for almost any user on the network today.

If GMIN did not use Bloom filters, on the other hand, profile information would be exchanged continuously, thereby increasing bandwidth consumption. In the considered data trace, replacing Bloom filters with full profiles in gossip messages makes the cost 20 times larger. Finally, we observe that, while GMIN's anonymity protocol continuously sends keep-alive messages, the only ones that impact bandwidth are those sent when new profile information needs to be exchanged.

4.4 GQE mechanism

Most modern search engines modify the queries that users submit in order to improve their efficiency. One of these modifications is query expansion. The search engine adds keywords to the query in order to improve the recall and find more relevant items. For instance, adding synonyms for the important keyword in the query is crucial to identify important documents that would otherwise be ignored. It also improves the ranking of some documents that use diverse keywords to express the same idea. Modifying the query of a user is a difficult task as a the system could potentially misunderstand the user's request and add wrong keywords. In that case, the results would be polluted by documents the user is not interested in. As explained in Section 4.5, most of the existing work on query expansion is not fully user-centric. In this section, we explore possibilities to use GMIN in the context of a fully decentralized and personalized query expansion system, GQE. We consider the case of a collaborative tagging system (folksonomy), such as Delicious or CiteULike, where every user is associated with a tagging profile. As we show, GQE significantly improves the recall [17] and

[5]The is burst is slightly longer on PlanetLab due to the lack of synchronization between the machines.

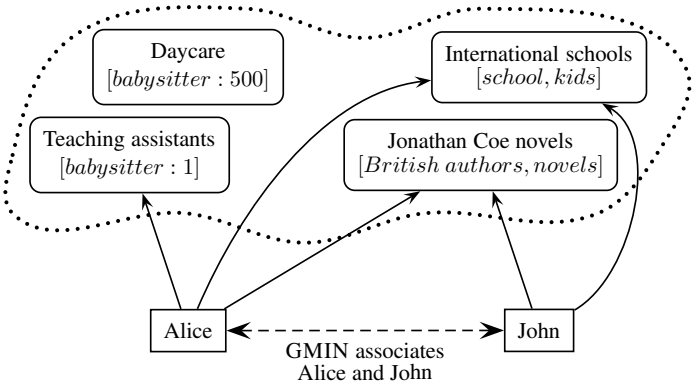


Figure 4.6: Associating John and Alice enables John to leverage the unusual association between the keywords teaching assistants and baby-sitter

precision [17] of the results, with respect to the state-of-the-art centralized personalized approach, namely Social Ranking [92].

Insight

One of the applications of personalized query expansion is solving problems such as homonymy. For instance, consider the "apple" query. GQE can leverage the user's past behavior and interests to predict if the user is looking for results about fruits or about computers. Such homonymy cases are quite frequent (jaguar, latex ...), and the user can usually solve them quite simply by adding a few keywords to make her query more precise. However, in some cases, a user's needs don't match the ones observed in the majority of the system, and the user doesn't have the knowledge required to improve her query and obtain the results she need. In that case, GQE will nevertheless manage to leverage the specificity of the user to automatically improve her query.

Consider the following real example. After living for several years in the UK, John is back to Lyon in France. To maintain his kids' skills in English, he is looking for an English speaking baby-sitter who would be willing to trade baby-sitting hours against accommodation. There is no doubt that such an offer would be of interest to many of the young foreigners living in a big city such as Lyon. Yet, John's Google request *"English baby-sitter Lyon"* does not provide anything interesting for the term *baby-sitter* is mainly associated with *daycare* or local (French) baby-sitting companies.

None of John's Facebook buddies in Lyon or the UK can help either as none has ever looked for an English speaking baby-sitter in Lyon. Yet, Alice living in Bordeaux after several years in the US, and who was looking for a similar deal with her kids, has been lucky enough to discover that teaching assistants in primary schools are a very good match.

Clearly, John could leverage Alice's discovery if only he knew about it. Should a system be able to capture the affinity between Alice and John, through their *common interest* in international schools and British novels for example (Figure 4.6), John could leverage Alice's discovery.

Indeed, consider a collaborative URL tagging system (e.g. Delicious) through which Alice has annotated the *teaching assistant* URL with *baby-sitter* (Figure 4.6). Assume both Alice and John have expressed their interest in international schools and British novels by tagging related URLs. A personalized search could leverage these affinities to return Alice's *teaching assistant* URL

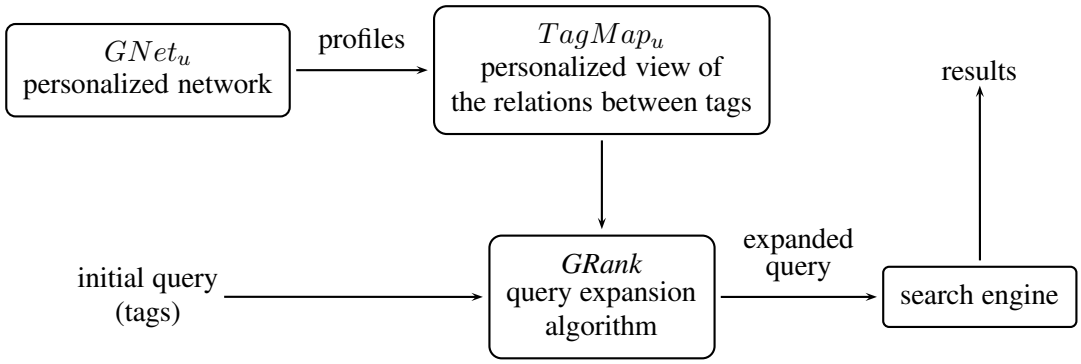


Figure 4.7: Expanding queries with GQE

first instead of the millions of URLs of standard (non English speaking) baby-sitter related URLs. Likewise, a personalized query expansion could expand John's query appropriately with tags derived from Alice's activities on the Web and make it easy to solve by any reasonable search engine. The crucial aspect here is the unusual (personal) association between baby-sitter and teaching assistant, which is relevant to a niche community (the one gathering Alice and John) while baby-sitter is dominantly associated with (non English speaking) daycare. Discovering such specific affinity is very rewarding for the users, yet challenging to automatically capture.

4.4.1 Overview

We use GMIN's GNETS to compute a data structure we call TAGMAP (Figure 4.7), a personalized view of the relations between all the tags in the user's profile and in her GNET. This is updated periodically to reflect the changes in the GNET. A query from the user is then expanded using the TAGMAP of that user through a centrality algorithm we call GRANK, which we derived from the seminal PageRank [17] algorithm. While PageRank computes the relative importance of Web pages (eigenvector centrality [1]), GRANK computes the relative importance of tags for a given user: we refer to this notion as the tag centrality in the sequel. GRANK estimates the relevance of each tag in the TAGMAP with respect to the query and assigns a score to the tag. We then recursively refine the correlation between tags by computing their distance using random walks, along the lines of [35].

4.4.2 TagMap

The GNET of a user u contains a set of profiles covering its interests. These profiles are used to compute the TAGMAP of u, namely a matrix $TagMap_u$, where $TagMap_u[a, b]$ is a score that reflects the distance between tags a and b as seen by the user u. We denote by IS_u the information space of the user u, namely its profile and the profiles in her GNET; TIS_u and IIS_u denote the set of all the tags and items in IS_u. The TAGMAP uses item-based cosine similarity to compute a score between the tags. The information needed to fill the TAGMAP is, for each tag in TIS_u, the number of occurrences of the use of that tag per item, i.e., for all $t \in TIS_u$, a vector $V_{u,t}$ of dimension $|IIS_u|$ is maintained such that $V_{u,t}[i] = x$, where x is the number of times the item i has been tagged with t in IS_u. More precisely, the similarity between two tags is computed as follows:

$$TagMap_u[a, b] = cos(V_{u,a}, V_{u,b}) \ .$$

	Music	BritPop	Bach	Oasis
Music	1	0.7	0.1	0
BritPop		1	0	0.7
Bach			1	0
Oasis				1

Table 4.2: Example of a TagMap

Table 4.2 depicts an example of a user's TAGMAP. In this TAGMAP, the score between *Music* and *BritPop* is 0.7, i.e. $TagMap_u[Music, BritPop] = 0.7$.

4.4.3 GRank

The number of tags q added to the initial query is a parameter of the query expansion algorithm. In the two approaches we present in this section, the query expansion algorithm assigns scores to tags and a query is expanded with the q tags scoring the highest. The TAGMAP contains a personalized view of the scores between pairs of tags. This information can be directly used to expand queries as in [92]. This approach, called *Direct Read (DR)*, simply computes for each tag sums the TAGMAP scores with respect to all the tags in the query:

$$DRscore_u(a) = \sum_{t \in query} TagMap[t, a]$$

While this approach seems intuitive, it gives bad results when the item sparsity of the system is high. This is very likely to happen for niche content: with a very large number of items, the relationships between tags might not always be directly visible in the TAGMAP. To illustrate the issue, consider the TAGMAP presented in Table 4.2 and a query on *Music* with $q = 2$. The TAGMAP exhibits a high score between *Music* and *BritPop* (based on a given set of items). In addition, there is a low score between *Music* and *Bach* suggesting that the user is more interested in *BritPop* than in *Bach*. However *BritPop* and *Oasis* have also a high score in the same TAGMAP (gathered from a different set of items), DR will never associate *Music* and *Oasis* whilst this association seems relevant for that user. In fact, DR would instead expand the query with *Bach*, increasing the result set size and reducing the precision of the search (Figure 4.8).

Our approach is based on the observation that, by iterating on the set of newly added tags, more relevant tags can be selected for the query. We capture this idea through an algorithm inspired by *PageRank* [17], which we call GRANK. GRANK is very similar to the rooted PageRank already used for link prediction in Section 2.5.1. In short, GRANK runs a rooted PageRank on the tag graph extracted from the TAGMAP and assigns the set of roots to the tags in the query. More specifically, considering the weighted graph provided by the TAGMAP, all the tags in TIS_u are vertices and, for each non-null score in the TAGMAP, we add an edge weighted by the score. These scores affect the transition probabilities when generating the query expansion. As opposed to PageRank, where each link is chosen with equal probability, in TAGMAP, the transition probability (TRP) from one tag to another depends on the edge weight provided by the TAGMAP:

$$TransitionProbability_u(a, b) = \frac{TagMap_u[a, b]}{\sum_{t \in TIS_u} TagMap_u[a, t]}$$

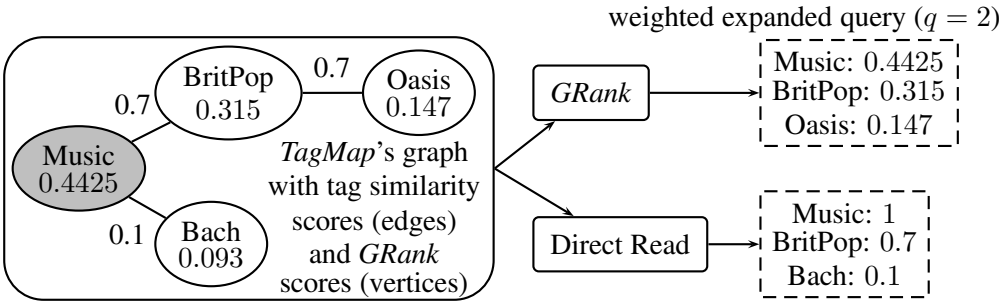


Figure 4.8: Example of query expansion

To limit the computation time of a score (which can be prohibitive in a large graph), instead of running an instance of GRANK per query, we divide the computation per tag in the query, forming partial scores. They are approximated through random walks [35] and cached for further use whenever the same tag is used in a new query. Still, a centralized server computing GRANK for all users would not scale. It can only be applied in the context of GQE where each user provides her processing power to compute its own GRANK.

Both *DR* and GRANK assign scores to the tags in the query and the ones added by the query-expansion. These tags are transmitted to the search engine in order to weights in the query evaluation.

4.4.4 Evaluation

We evaluated GQE's ability to achieve a complete and accurate query expansion. First, we describe our experimental setup. Then, we report on the evaluation of GQE on Delicious and CiteULike traces. Finally, we consider synthetic traces modeling the behavior of a mad tagger trying to disrupt GQE's operation.

Evaluation setup

Workload Each user is associated with a profile from a real trace (Delicious or CiteULike). The profiles drive the generation of requests: a user u generates a query for each item $i \in I_u$ such that at least two users have i in their profiles. The initial query consists of the tags used by u to describe item i since they are also likely to be the ones u would use to search for i. We evaluate GQE using the generated queries. Given a query from user u on an item i, we first remove i from u's profile so that u's *GNet* and *TagMap* are not built with it, then we determine if i is indeed an answer to the expanded query.

Search engine Although any search engine could be used to process a query expanded by *GRank*, for the sake of comparison, we consider the search engine and ranking method used in [92]. The search engine runs on the set of items available in the real trace and takes into account the weights assigned by the query expansion mechanisms. An item is in the result set if it has been tagged at least once with one of the tags of the query. To rank the items, the score of an item is the sum, for each tag in the query, of the number of users who made an association between the item and the tag, multiplied by the weight of the tag.

Evaluation criteria The query expansion mechanism is evaluated along the two classical and complementary metrics: recall and precision. Recall expresses the ability of the query expansion mechanism to generate a query that includes the relevant item in the result set. In these experiments, we are interested in the item i for which the query has been generated. A query succeeds when i is in the result set: recall is 1 if i is in the result set, 0 otherwise. Note that the size of the result set increases with the length of the query expansion. Since each document that has been tagged at least once with a tag in the query is added to the dataset, the more the query is expanded, the bigger the result set is. The popularity of the tags added to the query also has a huge impact on the number of items retrieved. While precise tags such as "BritPop" only add a few items to the result set, very generic tags like "Music" are used on millions of items. This potentially reduces the visibility of the item the user is looking for. To balance this effect, we also assess the quality of the query expansion by evaluating *precision*. We rely on the observation that although a Web search can find billions of related Web pages[6], this does not impair the user's experience as long as the page she wishes to find is among the top ones. Therefore, we consider the absolute rank (based on the score provided by the search engine) of the items in the result set as a metric for precision: namely, the precision is defined as the difference between the rank with query expansion and the rank with the initial query.

Recall is evaluated on the queries that do not succeed without query expansion. This comprises 53% of the queries in the CiteULike trace and 25% in the Delicious one. These results show that a significant proportion of items have been tagged by several users with no tags in common and highlights the importance of query expansion. Precision is evaluated on the queries that are successful even without query expansion since they provide a reference rank for the item.

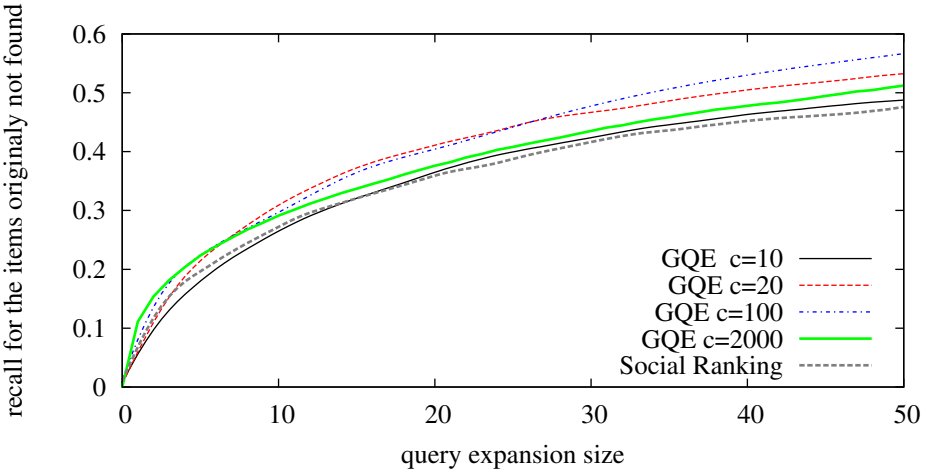
Impact of GQE's personalization

Our first results, in Figure 4.9, isolate the impact of personalization through a *TagMap* based on a small GNET with respect to a global *TagMap* based on all users' profiles. Figure 4.9a shows the extra recall obtained with query expansion sizes from 0 to 50 on the Delicious traces with *GNet* sizes from 10 to $2,000$. We compare these results with those of Social Raking, i.e. the case where *GNet* contains all the other users. The figure presents the proportion of items that were found with the query expansion out of the ones that were not found without. For example, the point $(20, 0.37)$ on the GQE 10-neighbor curve says that 37% of the requests not satisfied without query expansion are satisfied with a 20-tag expansion based on a *GNet* of 10 users.

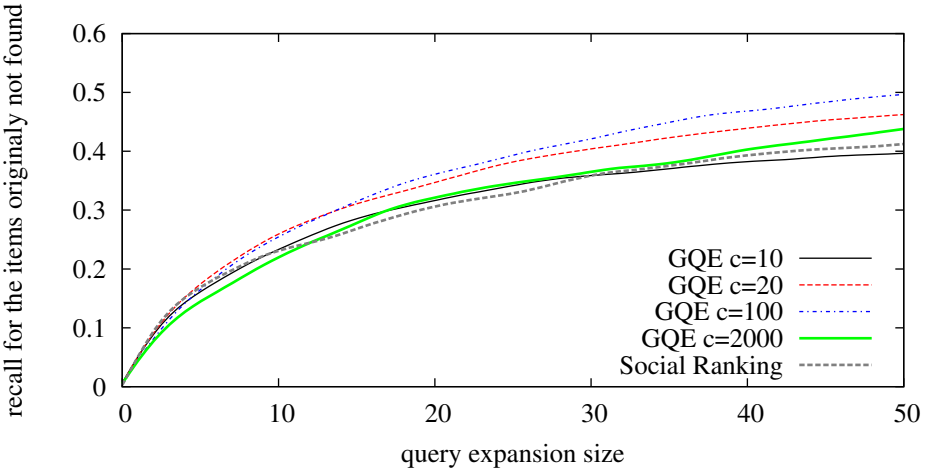
The figure highlights the benefit of personalization over a *TagMap* that would involve all users such as Social Ranking. Even though increasing the size of the *GNet* up to 100 has a positive impact on recall, larger sizes degrade performance. With a 30-tag query expansion, a 10-user *GNet* has a recall of 43%; recall goes up to 47% in a 100-user network and drops to 42% in the case of Social Ranking. As the number of user profiles integrated in the *TagMap* increases, relevant tags are gradually *swallowed* by less relevant tags or popular ones.

The experiments conducted on the CiteULike dataset, presented in Figure4.9b, lead to very same conclusions. The striking similarity between the two figures clearly demonstrates the benefits of personalization, since GQE is able to bring similar performance on two different datasets. This is clearly an illustration of how the baby-sitter/teaching assistant association in the example presented in Section 4.4 could be diluted if we had considered all users.

[6]The Yahoo! search engine returns 9 billions results to the "music" query.



(a) Delicious



(b) CiteULike

Figure 4.9: Extra recall (Delicious)

Impact of GQE's centrality

We evaluate here the impact of computing the relative importance of tags (*GRank*) over the DR query expansion. As mentioned previously, personalization is mostly responsible for increasing recall. However, computing the relative importance of tags (i.e., tag centrality) significantly improves the precision of the results. This is clearly illustrated in Figure 4.10, which compares our the results of:

- non personalized DR query expansion (Social Ranking), Figure 4.10a
- GQE DR query expansion, Figure 4.10b
- GQE GRANK query expansion, Figure 4.10c.

For the three configurations, increasing the query expansion size improves the recall for the items which were not initially found. Yet, this is accompanied by a significant drop in the precision for

71% of the items originally found when using the DR query expansion. *GRank* however with a 20-, resp. 50-tag query expansion, increases the recall of items not found initially up to 40%, resp. 56%, while increasing the ranking of approximately $58,5\%$ resp. 40% of the items which were returned to the initial query. This is a clear illustration of the impact of *GRank* on the precision. Interestingly enough, GRANK improves the precision for approximately 50% of items with a query expansion of 0. This is due to the fact that GRANK weights the tags in the query with respect to their importance in the context of the TAGMAP of the user issuing the query.

To summarize, while the direct use of the *TagMap* improves the recall over existing approaches, mostly through personalization, the precision is significantly improved through the centrality provided by *GRank*. The latter is of utmost importance as the better precision enables us to expand queries with more tags.

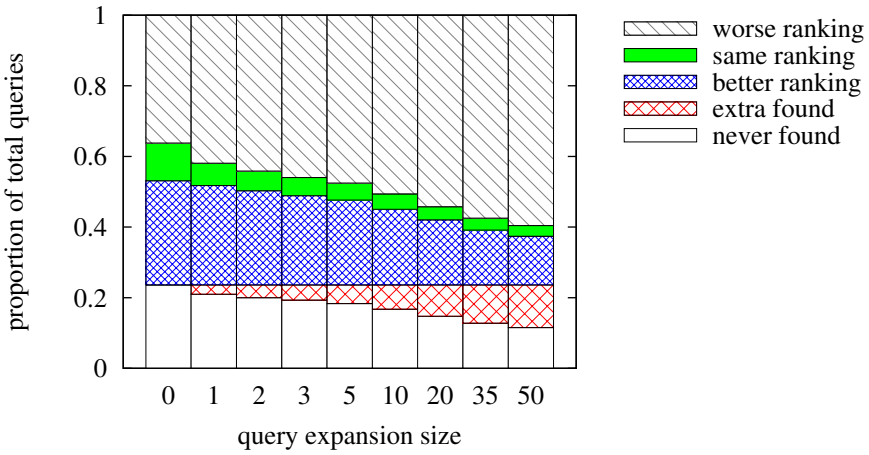
Synthetic traces

We also assess the resilience of GQE to attacks through synthetic traces. We consider the case of a user trying to force an association between two tags. We call this situation a GQE *bombing*, in reference to the *Google bombing* attacks [9]. As GQE benefits from a personalized query expansion mechanism, an attacker willing to manipulate the results of a user first has to modify her TAGMAP and therefore manipulate her GNET. As presented in Section 4.2.5, we assume that an attacker cannot run a Sybil attack to overwhelm the system with infected users. If the attacker tries to target many users by adding very diverse items to its profile, the attack fails. The GNET protocol favors precise overlapping of interests and therefore no user adds the attacker to its profile. On the contrary, if the attacker targets users in a specific community by forging a profile very similar to their interests, it is selected as an acquaintance and impacts their query expansion. But in that case, only few users are infected as the attack is specific to that community. If the attacker's objective is to promote some tags for marketing purposes, targeting a precise community can be sufficient to achieve its goal.

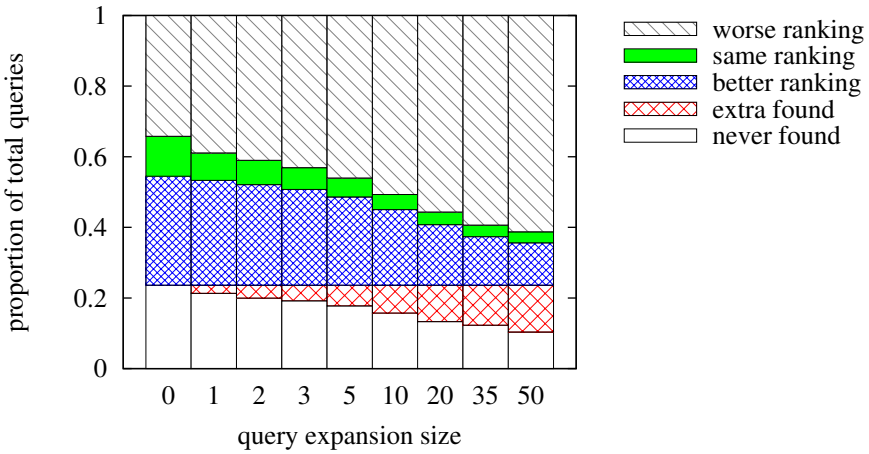
GQE is not fully protected against bombing operations. The personalization of GQE ensures that only users that exhibit similar interests can influence each-other. Nevertheless, it is still possible to target a specific community. In that case, it is difficult to automatically make a difference between an attack and a good query expansion. We do not propose any solution to this problem in this thesis. We believe that we could improve GQE's resilience to bombing attacks by making the query expansion mechanism explicit. Users would be able to blame other users for query expansion pollution and mark them as potential attackers in a reputation system. This would then be taken into account by other users in order to ignore the attackers in the future.

4.5 Existing Work

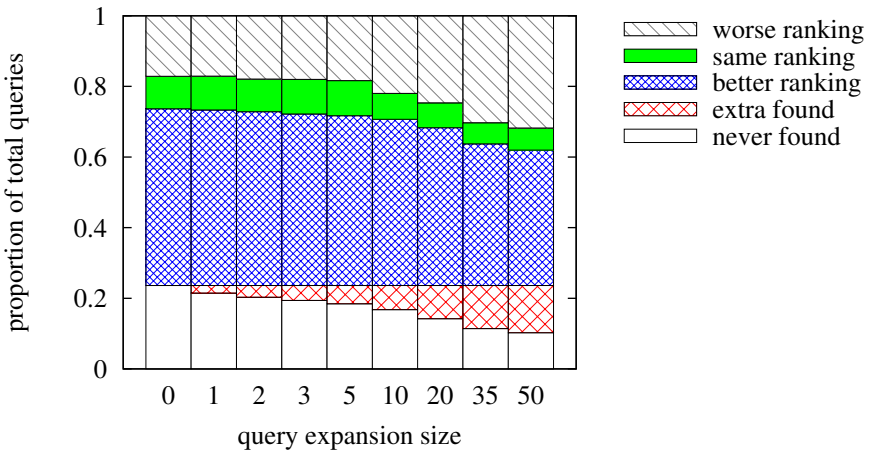
The GMIN network can be viewed as a semantic overlay devoted to boosting the search on the Web, in particular through query expansion. We overview below state-of-the-art semantic overlays. We also discuss query expansion techniques and compare them to GQE. Finally, we present state-of-the-art folksonomy metrics.



(a) Social Ranking



(b) GQE (DR)



(c) GQE (GRANK)

Figure 4.10: Overall performance (Delicious)

4.5.1 Semantic overlays

Explicit approaches

Many user-centric approaches [4, 11, 62] consider an explicit (predefined) social network, typically derived from systems such as Facebook, LinkedIn, LastFM or MySpace, to improve the search on the Web. The assumption here is that the explicit, declared, relationships between users reflect their shared interests [63] and can help their search. In many cases, however, and as pointed out in [3, 11], the information gathered from such networks turns out to be very limited in enhancing the navigation on the Web.

A couple of alternative approaches exploit the unknown acquaintances of a user to improve the search further [51]. These require the user to explicitly declare a list of topics of interests and assume a strict topic classification, rendering it impossible to dynamically detect new communities. However, it provides an interesting feature: a user interested in several topics splits her acquaintances among them with respect to her involvement. GMIN goes a step further and automatically assigns acquaintances to users solely based on their common items: GMIN's associations naturally evolve over time. It enables the system to benefit from new and useful information while sharing the acquaintances among the several interests of the users. This independence from any a priori list of topics makes the system well suited to changes. This gives GMIN much more freedom and the ability to dynamically evolve with the users without depending on a given list of topics.

Implicit approaches

Some approaches form semantic acquaintances by associating users with those that successfully answered their queries in the past [22,24,34,42,75,77]. In file sharing communities, these approaches gradually transform a random overlay into weakly structured communities that improve the success ratio of queries. Various replacement policies have been proposed to limit the number of semantic neighbors. A major drawback is the need for a warm-up phase to establish the network based on a reasonable sample of queries: the first queries are sent to random users, leading to poor results. Furthermore, because the acquaintances of a user only reflect her last few queries, queries on new topics are inefficient. GMIN actively locates the acquaintances of a user independently of her past queries but based on her (full interest) profile instead.

In order to avoid the warm-up phase, some (active) clustering approaches rely on gossip. Voulgaris and Steen [84] use the number of shared files in common as a proximity metric. While this approach improves search, it overloads generous users that share many files. Jin *et al.* [48] consider the cosine similarity of the users as a metric to penalize non-shared interests. This gives better performance than simple overlap and underlies our GMIN metric. These approaches also tend to choose uniform acquaintances that only reflect the primary interest of a user, while GMIN spreads the acquaintances among all the interests of a user. As discussed in [56], a user has usually several areas of interests, typically non-correlated: on average, a user requires three different communities (clusters) to find 25% of the data she is looking for.

Patel *et al.* [70] use gossip to select semantic acquaintances for RSS feed transmission. The selection algorithm increases the score of users that provide items not covered by other acquaintances. GMIN's multi-interest metric can be considered a generalization of this idea. Instead of giving a bonus to the first user covering an interest, GMIN increases the score for rare interests in general.

In [27], a centralized ontology-based analysis of all the items in the system assigns a type to each one of them and infers users' interests. This procedure is centralized and relies on an external source of knowledge for the classification, while GMIN is fully distributed and only uses the item interest pattern of users as a source of information. In [8], the attributes of the items generate a navigable overlay. This approach is centered on items rather than users, and does not scale with the number of items a user shares. It is also limited to items which carry a lot of metadata, like music or text files. The system proposed in [58] is fully distributed: it clusters items in order to obtain semantic coordinates. The user joins the unstructured community responsible for most of her items and obtains small-world links to other communities. Although the user has to advertise the items which are not part of its main community, this approach scales quite well with the size of the user profile. However, it suffers from the same drawback as [8] and requires items with a lot of metadata for clustering. GMIN relies on shared items to determine similarity between users. Metadata, such as tags, can also be leveraged in order to detect common interests. Nevertheless, since our goal in this chapter is to provide a query expansion functionality, clustering users using tags would have limited the number of new tags discovered, hence the use of items only.

4.5.2 Query expansion

Personalized systems

Several approaches leverage information about users' past behavior to personalize their query expansion process. In [18], the query is expanded by tags already used in previous queries. In [47], the query is expanded using the information available on the user's local computer. The tags are chosen with a local search engine and completed by user feedback. In both cases, no information is inferred from other users. We believe that the main interest of query expansion is to add tags which are not part of the user's usual vocabulary, which is why GQE learns from other users.

Global approaches

Centralized Web search engines often rely on handwritten taxonomies (Wordnet[7], Open Directory Project[8], Yahoo! Directory[9]) to expand queries. Instead, we only consider knowledge that is automatically extracted from the user's profile. [12] proposes a query-expansion framework that uses social relations to rank results. However, only the scoring model is personalized. The query-expansion mechanism exploits the co-occurrence of tags in the full document collection leading to a non-personalized query-expansion output. We now describe Social Ranking [92], which is somehow the system the most similar to our work, despite its centralization.

Social Ranking Zanardi and Capra propose Social Ranking [92], an architecture for personalized search in folksonomies. Social Ranking relies on the cosine similarity between the tags of the users to compute the similarity between users. Each tag is weighted by the number of times the user tagged an item with it. The similarity between tags is the same for all users. It consists in the cosine similarity between the items that have been tagged by these tags. Again, each item is weighted by the number of times a user in the system tagged the item with it.

[7] http://wordnet.princeton.edu/
[8] http://www.dmoz.org/
[9] http://dir.yahoo.com/

A query is first expanded, using the tag similarity measure, and then evaluated. The query expansion process assigns a weight to each tag, depending on the tag similarity, and keeps the k tags which obtained the best scores. The score of a document depends on how many user tagged the document with a tag in the expanded query. This measure is weighted using the weight of the tag, and the similarity between the user that tagged the item and the user which issues the query.

Zanardi and Capra experiment Social Ranking on a CiteULike dataset. They show that taking into account the similarity between users in the raking process increases the raking of relevant documents, especially the ones which are not popular. The query expansion process recovers additional relevant documents, and they do not observe a significant drop in precision.

In Section 4.4.4, we compare GQE against Social Ranking and show that GQE obtains significantly better results. There are two main reasons for this. Social Ranking weights the contributions of the users depending on their similarity with the user issuing the query. But these weights are not taken into account in the query expansion process: the query expansion outputs the same results for all the users. Similarly to Social Ranking, GMIN considers the similarity between users to select the GNET. However, GQE is personalized, and each user generates her own query expansion using the information in the GNET. This accounts for the subjectivity of the relations between tags and increases the recall of the search. The second reason which explains the performance difference is GRANK. The system we propose is fully distributed, therefore each user contributes to the query processing with her own machine. We leverage this processing power to compute the relative centrality of tags and obtain a more accurate weighting of the query. As we show in Section 4.4.4, this has a huge impact on the precision of the results.

4.5.3 Folksonomy metrics

Several metrics have been proposed to measure the distance between tags or items in a folksonomy. This chapter presents a framework for decentralized user-centric applications, therefore it does not focus on metric. Our preliminary experiments have shown that cosine similarity yields more effective results, so this is the one we implemented in our experiments and compared with our *set cosine similarity* metric. Nevertheless, we have also considered several other similarity measures:

Edit-distance [86] captures the relative similarity between tags. The edit-distance between two tag names equals the minimum number of insertions, deletions and substitutions needed to transform one name (string) to the other. This distance basically allows to detect misspelled words (or words with the same base). We did not use this in our context as our goal was to expand queries with new ones instead of correcting errors: this is somewhat complementary.

Co-occurrence count [65] determines the number of times two tags are used on the same items, normalized by the times the first tag is used. The use of co-occurrence count helps derive for instance the fact that the tag "tennis" is close to the tag "sport". The use of this metric in a user-centric way prevents ambiguities of tags by looking at the context within which tags are used.

Cosine similarity is an effective metric to compute distance between users [88, 94] or tags [19, 87]: we heavily relied on a variant of this metric in our system, which we adapted to encompass the multiple interests of a user. In fact, we compared this metric with co-occurrence count in a preliminary step of our work and came up with the conclusion that cosine similarity yields more effective results.

To rank the tags in our query expansion solution, we use an algorithm inspired by the rooted version of the PageRank algorithm [17] to compute the distance between tags in terms of their *relative (eigenvector) centrality*. A similar idea was used in the context of top-k item ranking in [35, 44, 45, 88].

4.6 Concluding Remarks

We presented GMIN, and Internet-scale protocol that discovers connections between users and leverages their social information to propose a personalized Web experience. GMIN is fully distributed, no central authority is involved and there is no single point of failure. With little information stored and exchanged, every GMIN user is associated with a relevant network of acquaintances. GMIN protects the privacy of the users by hiding the association between a user and her profile. The decentralized nature of GMIN makes it possible to use costly personalization algorithms. GQE computes the TAGMAP of users and relies on GRANK, a centrality measure. They both would have been computationally prohibitive in a centralized system. Interestingly, GMIN naturally copes with certain forms of free-riding: users do need to participate in the gossiping in order to be visible and receive profile information. As we have also shown, the impact of arbitrary tagging, or even individual malicious tagging, is very limited.

Yet, GMIN has some limitations and many extensions might need to be considered. Our gossip-on-behalf approach is simple and lightweight, but users may require additional security even by enduring a higher network cost. It would be interesting to devise schemes where extra costs are only paid by users that demand more guarantees. It would also be interesting to explore the benefits of a social network of explicit friends. For instance, GMIN could take such links into account as a ground knowledge for establishing the personalized network of a user and automatically add new implicit semantic acquaintances. This would pose non-trivial anonymity challenges.

CHAPTER 5

CONCLUSION

Context

Personalized Web applications have a central place in the Internet, as they efficiently guide the user to find relevant content among an increasingly large number of websites. Still, the current implementation of these services raises scalability and privacy problems. To benefit from personalized recommendations, users have to express their preferences and provide private information. This data is processed on data centers owned by private companies. Generating personalized recommendation for all users is a very computationally intensive process. While new paradigms have been developed to face the growing need for processing power, maintaining such infrastructure is extremely costly. Hence, the recommendations are approximated in order to scale with the demand. To maintain these data centers, private companies need to find business opportunities and use this data for commercial purposes, such as personalized advertising. This exposes the privacy of the users. As they become aware of how much private information private companies store, users may become reluctant to use these personalized services.

Summary of the contributions

In this thesis, we argue that social applications should be deployed on peer-to-peer (P2P) networks. In a P2P architecture, users contribute to the processing of the information and remain responsible for their own private data. While P2P improves the scalability and privacy, it also creates new challenges. As the data is now distributed among all the participants, it is necessary to design efficient algorithms to locate and retrieve it.

We present three main contributions. In the first one, we evaluate the privacy risk of social information. In the second one, we propose a link prediction algorithm for P2P social networks. Finally, the third contribution is a P2P platform that leverages social information for personalized Web search.

Link prediction from social information Link prediction consists in predicting new edges in a social network solely using the existing ones. We define the *cold start link prediction* problem, which consists in predicting the edges in a social network when the social network is hidden. However, some social information about the users is available. We propose an original solution to solve this problem. It relies on a two-phase approach based on a probabilistic graph. We apply it to a dataset from Flickr, using the group memberships of the users as social information. In the first phase, we use several features to compare the group memberships of pairs of users and assign a probability to the existence of a link between them. This generates a probabilistic graph which contains an edge for all pairs of users having at least one group in common. The second phase of our approach consists in adapting classic link prediction algorithms to probabilistic graphs in order to refine the probabilities and increase the coverage of the prediction. Our experiments show that, even though the group memberships are weakly correlated with the social network, we manage to predict the social network of the users with a reasonable accuracy. This work highlights the privacy problems caused by the centralization of social information.

Decentralized link prediction for social networks We propose SOSC, a distributed algorithm to predict new links in a social network. Recommending friends is a crucial functionality in social network, hence this is a preliminary step towards P2P social networks. Inspired from graph drawing algorithms, SOCS relies on gossip protocols to perform a distributed force-based embedding of the social graph. Each vertex is assigned social coordinates, and can compute its social distance with respect to any other vertex. We show that SOCS is adapted to P2P networks, as it converges quickly and is not affected by churn. Furthermore, SOCS exhibits a low network cost. We also show that SOCS is able to accurately predict links in a social network by leveraging its community structure. Despite its constraints related to the decentralization, SOCS outperforms some centralized and costly link prediction algorithms.

Distributed personalized Web search In the context of social information websites, we propose GMIN, a gossip algorithm that provides each user with a list of similar anonymous acquaintances. Contrary to traditional clustering algorithms, GMIN was designed to take into account all the interests of the users, even the minor ones. We experimentally show that this enables GMIN to answer a wide range of queries. We go one step further by introducing GQE, a query expansion mechanism that leverages the neighbors identified by GMIN. Query expansion automatically adds keywords to a query, in order to improve the quality of the results. We argue that the relation between tags can be subjective, hence GQE performs a personalized query expansion. We experimentally show that this improves the recall of the queries. In addition, as GQE is computed in a P2P context, it benefits from more computing power than the centralized server. Consequently, we improve GQE by computing the relative centrality between the tags in order to weight them. We show that this significantly improves the precision of the results.

Perspectives

In this thesis, we present three different contributions. For each of them, we provide, at the end of the chapter, a few indications on possible future work. In addition, we now present two main projects

that we foresee could lead to interesting results. We also mention Diaspora, a P2P social platform that we could use to implement our protocols.

Multi-purpose coordinates system In Chapter 3, we propose SOCS, a distributed graph embedding protocol that assigns social coordinates to vertices in a social network. These coordinates are then leveraged to predict new links. We believe that SOCS could be adapted to other classes of recommender systems. For instance, many websites propose a rating mechanism, in which users can express their opinion about items by assigning a score. This is the case for example in the Netflix [1] dataset: users rate movies. It would be interesting to adapt SOCS to bipartite graphs in order to represent the relations between users and items. The main advantage is that users and items would be placed in the same coordinates space, so it would be possible to obtain a distance between a user and an item, but also between two users and two items.

The force model we currently use applies attractions between the entities that are connected and repulsion between all. In order to take into account the rating of users, we would have to propose a new force model that applies attractions between users and items they like, and repulsions between users and the items they dislike. The new force model would also apply weak repulsions between two users, two items and a user and an item she has not rated. Such a setup is very challenging as we must ensure that the coordinates converge to a stable state. While conceiving is P2P network in which each user is represented by a machine is quite straightforward, the placement of the items is more problematic. Several users are interested by the same item, but the system should provide unique coordinates for each item. Thus, a unique machine should be responsible for each item, and its state should be replicated as the coordinates of the item should not be lost when the user disconnects. One possibility to obtain an automatic placement of items would be relying on a Distributed Hash Table (DHT). However, this would give the responsibility of an item to a user which is not necessarily interested in it. A better solution would take into account the coordinates so that users would be responsible for items in their social neighborhood. This would optimize the computation of the repulsions since only one instance of the clustering protocol would be necessary: the user and the item would have the same neighbors hence the same repulsions.

Enhanced personalization control In Chapter 4, we present a framework for personalized Web search using social information. In particular, we show that a personalized query expansion process can significantly improve the quality of the results proposed to a user. Nevertheless, in some cases, the user may be interested in obtaining an answer which differs from what her community is usually interested in. This is important, as too much personalization could lead to missing information from people with different points of view.

One possible evolution of our protocol would involve routing the users' queries on the GMIN overlay. Hence, it would be possible to aggregate results along the path of the query, from the user to different communities. The answers found close to the user would be very personalized, while more distant results would come from different communities and therefore be more general. The diffusion of the query is a difficult problem. For scalability reasons, it is not possible to flood the query on all links of the network. We believe a more controlled diffusion scheme is necessary, potentially using the interests of the users to directly target users able to provide answers.

[1] http://www.netflix.com/

Implementation for end-users In this thesis, we propose several protocols to personalize the Web experience of Internet users. These protocols are experimented using real datasets, through a simulator and on a distributed experimentation platform (Planetlab). Still, none of these experiments involves real users actively using the system. The Diaspora [2] project aims at building a decentralized social network. It has started very recently and is available under an open-source license. In the future, we could use it as a development platform to implement some of the algorithms described in this thesis and obtain feedback from the users.

[2] http://www.joindiaspora.com/

CHAPTER 6

FRENCH SUMMARY

Introduction

L'émergence du "Web 2.0" a fondamentalement changé le comportement des utilisateurs d'Internet. Le Web, qui ne disposait auparavant que de contenu statique, est devenu une plate-forme collaborative. Les utilisateurs sont maintenant actifs, et contribuent au contenu des sites Internet en exprimant leurs opinions et en partageant de l'information.

Réseaux sociaux Les réseaux sociaux, comme Facebook, ont été créés pour permettre aux utilisateurs de garder le contact avec leurs amis. Sur ces plates-formes, les utilisateurs peuvent publier de l'information pour des amis qu'ils ont explicitement désignés. Il est également possible, par exemple, de partager des photos et d'associer un ami avec une photo sur laquelle il apparaît. Facebook est un moyen efficace de contacter rapidement des milliers de personnes pour organiser des évènements.

Les réseaux sociaux ont développé des outils pour permettre à d'autres sites Internet d'accéder au profil d'un utilisateur afin de personnaliser ses pages Web en fonction de ses amis ou de la ville où il habite par exemple. Ainsi, un utilisateur peut recommander une page Web à ses amis. Le site Web peut aussi détecter que deux amis sont intéressés par la même information, ce qui leur donne l'occasion d'en discuter. Alors même que le contenu d'Internet ne cesse de grandir, l'utilisation des réseaux sociaux peut être un moyen efficace de fournir aux utilisateurs un contenu personnalisé et plus intéressant.

Information sociale Contrairement aux réseaux sociaux, qui sont plutôt destinés aux personnes qui se connaissent réellement, de nombreux sites Internet ont été créés pour tirer partie d'informations personnelles. Ils ne s'appuient pas sur un réseau social explicite, mais permettent aux utilisateurs de partager leurs préférences. Par exemple, LastFM enregistre la musique que les utilisateurs écoutent, et Delicious permet aux utilisateurs de partager leurs listes de sites Web préférés. Ces sites Internet agrègent l'information fournie par chaque utilisateur afin d'en extraire des connaissances, comme les

tendances dans le monde de la musique, ou les sujets à la mode sur Internet. En retour, les utilisateurs bénéficient de recommandations personnalisées.

Certains sites Internet reposent sur les utilisateurs pour annoter le contenu qu'ils proposent. Youtube et Flickr permettent aux utilisateurs de partager des vidéos et des photos. Ces contenus sont très difficiles à indexer pour les moteurs de recherche, contrairement au texte. Les utilisateurs peuvent annoter le contenu du site avec des mots clés, ou *tags*, de leur choix, afin de faciliter la diffusion de leur propre contenu, ou d'améliorer l'efficacité du site. Cela permet aux moteurs de recherche de découvrir ces documents plus facilement, puisque les tags peuvent être utilisés pour en identifier le contenu. L'étude des activités des utilisateurs permet aussi de découvrir des liens entre différents contenus et de connaître leur popularité.

Vie privée Les plates-formes sociales sont extrêmement importantes pour la personnalisation de la navigation Internet. Cependant, elles posent d'importants problèmes au sujet du respect de la vie privée. Les interactions sont devenues tellement complexes que les utilisateurs ne savent souvent plus quelle information est enregistrée et qui peut y accéder. Ce problème vient principalement des entreprises privées qui gèrent ces plates-formes. Les utilisateurs ont besoin de protéger leur vie privée, mais ces entreprises ont besoin de rentabiliser leur infrastructure. Il est bien connu que les amis s'influencent entre eux, par conséquent, les réseaux sociaux peuvent être utilisés à des fins commerciales. Par exemple, Facebook a développé une fonctionnalité qui prévient automatiquement les amis d'un utilisateur lorsque celui-ci fait un achat en ligne sur Amazon. Cette application met clairement en danger la vie privée des utilisateurs, et elle a été retirée suite à leurs réactions. De même, il est courant que des informations personnelles soient vendues à des fins de publicité ciblée. Par conséquent, les utilisateurs pourraient devenir réticents à partager leurs données personnelles. C'est un problème important pour les plates-formes sociales, qui doivent trouver un équilibre entre l'information qu'elles exposent et la qualité des services qu'elles proposent.

Une navigation personnalisée d'Internet s'appuie sur des données personnelles, que ça soit à travers les réseaux sociaux ou à travers les informations sociales. La personnalisation peut fortement améliorer la navigation des utilisateurs mais ceux-ci pourraient se sentir menacés par les pratiques des entreprises privées et rejeter ces plates-formes.

Coût et limitations Le nombre d'utilisateurs de plates-formes sociales ne cesse d'augmenter. Par conséquent, il devient de plus en plus difficile de faire face à la quantité d'information à traiter. Facebook affiche plus de 500 millions d'utilisateurs, et LastFM 40 millions. L'extraction de données personnalisées nécessite des centres de calcul extrêmement coûteux, que ce soit en matériel ou en énergie. Ces centres de calcul sont souvent composés de milliers de machines "classiques", et de nouveaux paradigmes de calcul distribué ont été créés pour les exploiter.

Toutefois, le calcul de recommandations personnalisées reste extrêmement coûteux. Pour faire face à la demande, les services de recommandation regroupent les utilisateurs en communautés d'intérêt et recommandent le contenu pour toute une communauté. Ainsi, les recommandations sont approximées, ce qui réduit leur qualité. Si les plates-formes sociales disposaient de d'avantage de puissance de calcul, elles pourraient utiliser des algorithmes plus sophistiqués et améliorer la qualité de leur service.

Le calcul de recommandations personnalisées est très coûteux et nécessite des infrastructures dédiées. Le niveaux de personnalisation offert aux utilisateurs est limité par la puissance des centres de calcul.

Réseaux pair-à-pair (P2P) Dans cette thèse, nous défendons l'idée que les plates-formes sociales devraient être déployées sur des réseaux P2P. Ces applications sont centrées sur l'utilisateur, par conséquent, elles peuvent s'envisager facilement dans un contexte P2P : chaque utilisateur participe au réseau avec son propre ordinateur.

Les réseaux P2P sont complètement décentralisés, et chaque participant contribue à l'opération d'un service. Ce modèle est souvent opposé au modèle client/serveur, pour lequel une machine centrale, le serveur, répond aux demandes de tous les clients. Par conséquent, les réseaux P2P sont reconnus pour leurs propriétés de passage à l'échelle. La décentralisation de l'architecture augmente également sa fiabilité puisque le service est répliqué sur des milliers de machines partout dans le monde.

Une plate-forme sociale déployée sur un réseau P2P a deux avantages principaux. Premièrement, chaque utilisateur garde le contrôle de ses données personnelles et n'a pas besoin de les stocker sur un serveur central appartenant à une entreprise privée. Une plate-forme P2P est neutre et n'appartient à personne. Deuxièmement, le passage à l'échelle des réseaux P2P en terme de puissance de calcul et de stockage permet d'améliorer la qualité des services personnalisés. Les centres de calcul évoluent vers des architectures distribuées, nous proposons d'aller encore plus loin en intégrant l'application directement sur les machines des utilisateurs. De cette façon, il n'est plus nécessaire de gérer des centres de calcul coûteux. De plus, comme la puissance de calcul disponible sera supérieure, l'architecture P2P permettra d'offrir de nouveaux services plus performants que ceux existants dans les architectures centralisées actuelles.

Cependant, créer une plate-forme sociale décentralisée soulève aussi plusieurs difficultés. Comme chaque utilisateur gère ses propres données, le système doit pouvoir localiser efficacement l'information nécessaire, sans générer trop de trafic réseau. L'architecture doit être suffisamment efficace pour fournir un service de qualité équivalente à celle d'un système centralisé. Si la puissance de calcul des réseaux P2P peut permettre l'émergence de nouvelles fonctionnalités, il est impératif de préserver la qualité des fonctions existantes. De nombreuses architectures P2P ont été proposées pour gérer de l'information de façon distribuée. Nous utilisons des algorithmes épidémiques pour structurer le réseau et trouver l'information nécessaire.

Contributions Dans cette thèse, nous nous intéressons à la création d'une plate-forme sociale basée sur un réseau P2P. Nous sommes convaincus qu'il s'agit d'une étape importante vers un meilleur contrôle de la vie privée. Cependant, les réseaux P2P ne sont pas une réponse suffisante à ce problème. Ils doivent être utilisés en conjonction avec d'autres méthodes, comme "l'onion routing", pour mieux protéger les utilisateurs. Le but de cette thèse est de proposer des algorithmes adaptés aux conditions des réseaux P2P pour exploiter les systèmes sociaux. Par conséquent, nous nous intéresserons principalement aux mécanismes permettant de localiser et d'obtenir l'information nécessaire à la personnalisation. Nous mentionnerons des mécanismes de protection de la vie privée, mais ne prétendons pas résoudre ce problème.

La première contribution que nous présentons montre comment l'information sociale peut être utilisée pour prédire un réseau social tenu secret. Cela met en évidence les risques des solutions centralisées et justifie le déploiement d'architectures P2P. La seconde contribution s'intéresse au problème de la recommandation d'amis dans le contexte d'un réseau social décentralisé. Enfin, la troisième contribution est une architecture P2P pour un système de recherche personnalisé.

Prédiction d'un réseau social à partir des informations sociales des utilisateurs

Contexte

La prédiction de liens, définie par Liben-Nowell et Kleinberg, est un problème classique en fouille de données. L'objectif est, à partir d'un graphe social existant, de deviner quelles sont les nouvelles connexions qui vont apparaître dans ce graphe dans un futur proche. Pour cela, les approches existantes s'appuient sur les caractéristiques du graphe, telles que la distance entre deux sommets, ou le nombre de voisins en commun. La prédiction de liens montre donc à quel point l'évolution d'un graphe est conditionnée par son état actuel, par opposition à une évolution aléatoire par exemple.

Dans notre cas, nous considérons que le graphe social est tenu secret, car les utilisateurs ne souhaitent pas révéler ces informations sur leur vie privée. Par contre, une entreprise, qui possède des données sur les activités de ces utilisateurs, souhaite reconstruire ce graphe. L'objectif est multiple. La connaissance du graphe social des utilisateurs permet d'optimiser des stratégies marketing par l'adoption de techniques dites virales. Une bonne prédiction du réseau social permettrait également à l'entreprise, si elle le souhaite, de développer son propre réseau social explicite sur son site Internet. En effet, cela permettra de suggérer des connexions aux utilisateurs et donc de créer le réseau beaucoup plus rapidement. Comme cette entreprise n'a aucune connaissance sur l'état actuel du graphe social, elle ne peut pas utiliser les méthodes traditionnelles de prédiction de liens. Par contre, elle dispose des données sur les activités des utilisateurs pour essayer d'inférer un graphe social de départ sur lequel travailler. Notre objectif, dans cette section, est d'étudier dans quelle mesure cette entreprise peut obtenir une reconstruction précise du graphe social uniquement à partir de cette information auxiliaire. En d'autres termes, nous cherchons à savoir à quel point les informations sur les activités des utilisateurs mettent en évidence des liens sociaux qu'ils souhaiteraient garder secrets.

Contributions

Nous proposons une approche en deux phases pour résoudre le problème de la prédiction de liens quand le graphe social est initialement vide. Dans un premier temps, nous utilisons l'information auxiliaire dont l'entreprise dispose pour inférer un graphe social probabiliste. Cela signifie que nous élaborons des mesures qui nous permettent, à partir des données sur les activités des utilisateurs, d'estimer la probabilité que deux utilisateurs soient liés dans le graphe social caché. Puis, dans un second temps, nous adaptons les algorithmes classiques de prédiction de liens pour affiner ces probabilités et découvrir de nouveaux liens. Nous mettons en pratique notre méthode sur des données issues du site Internet Flickr. Sur cette plate-forme de partage de photos, les utilisateurs peuvent désigner quels sont leurs amis. Cette information constitue le graphe social caché que nous souhaitons prédire. Ils peuvent également rejoindre des groupes d'intérêt, comme *"Nikon Selfportraits"*, ou *"Cat and Dog: not Cat or Dog"*. Nous utilisons la connaissance des membres des groupes comme information auxiliaire pour démarrer la prédiction de liens. Les groupes d'intérêt des utilisateurs sont, en pratique, assez peu révélateurs des liens sociaux. Cependant, ils ont l'avantage d'être présents sur de nombreux sites communautaires. Par exemple, sur un forum de discussion, on peut considérer qu'un sujet constitue un groupe. Ainsi, l'approche que nous présentons est tout à fait généralisable à d'autres jeux de données.

Construction du graphe probabiliste

Pour construire une première approximation du réseau social, nous utilisons un graphe probabiliste. Nous nous appuyons donc sur la connaissance des groupes d'utilisateurs pour estimer leur probabilité d'être connectés dans le graphe. Nous étudions quatre types de mesures :

- L'*activité des utilisateurs* : plus un utilisateur rejoint de groupes, plus il est actif sur le site Internet. Par conséquent, il a également plus de chances d'être bien connecté dans le réseau social.
- Le *nombre de groupes en commun* : si deux utilisateurs ont beaucoup de groupes d'intérêt en commun, alors ils ont plus de chances d'être connectés dans le réseau social.
- La *taille des groupes en commun* : si deux utilisateurs sont membres d'un même petit groupe, alors ils ont plus de chances de se connaître que si se groupe contient beaucoup d'utilisateurs.
- La *proximité temporelle* : si la date à laquelle un utilisateur rejoint un groupe est proche de celle à laquelle un autre utilisateur rejoint ce groupe, alors il est possible qu'ils se connaissent et se soient influencés.

Nous construisons plusieurs mesures basées sur chacune de ces observations, puis nous combinons les meilleures d'entre elles en un seul prédicteur. Ce prédicteur nous permet de construire le graphe social probabiliste : tous les utilisateurs qui ont au moins un groupe en commun sont reliés par un arc pondéré par leur probabilité d'être connectés. Par contre, ce prédicteur ne permet pas d'évaluer les connexions entre deux utilisateurs qui n'ont aucun groupe en commun. C'est pourquoi la seconde étape de notre approche s'intéresse aux chemins dans le graphe probabiliste, afin de pouvoir prédire un lien entre tous les utilisateurs.

Prédiction de liens dans un graphe probabiliste

Nous utilisons les chemins reliant les utilisateurs dans le graphe probabiliste pour affiner les probabilités et prédire des liens entre des utilisateurs situés à plusieurs sauts de distance dans ce graphe. Pour cela, nous nous appuyons sur trois mesures classiques de la prédiction de lien, et les adaptons au graphes probabilistes.

- Le *nombre de voisins en commun* : plus deux utilisateurs ont des voisins en commun, plus ils ont de chance d'être eux-mêmes reliés dans le graphe social.
- *Katz* : cette mesure fait la somme du nombre de chemins entre les utilisateurs. Chaque chemin est pondéré en fonction de sa longueur, les chemins courts étant de meilleurs indicateurs de lien social.
- *La centralité relative (rooted PageRank)* : pour chaque utilisateur, on effectue des marches aléatoires sur le graphe. Plus la marche a de chances de passer par un utilisateur donné, plus ils ont de chances d'être connectés.

Nos expériences montrent que la prise en compte des probabilités améliore grandement la précision de la prédiction. Par conséquent, notre approche de la prédiction de liens est tout à fait adaptée à une connaissance imparfaite du graphe social.

Conclusion

Nous avons proposé un nouveau problème, qui consiste à prédire les liens d'un graphe social quand celui-ci est totalement inconnu. Notre approche modélise le comportement d'un adversaire qui souhaiterait deviner une information tenue secrète par les utilisateurs d'un site Internet. Nous avons montré que, bien que l'information des groupes d'utilisateurs soit une source d'information peu fiable, il est possible de prédire de façon relativement précise les liens entre utilisateurs. Nous pensons que notre algorithme pourrait être utilisé pour évaluer l'efficacité de techniques d'anonymisation de données : si la prédiction de liens est précise, alors les données sont sensibles.

Prédiction de liens distribuée pour les réseaux sociaux

Contexte

De nombreuses applications distribuées pourraient bénéficier d'un système de recommandation décentralisé. Par exemple, une plate-forme de partage de musique pourrait recommander des artistes aux utilisateurs. Un réseau social, comme Diaspora[1], doit pouvoir recommander de nouveaux amis aux utilisateurs. Pour des raisons de performances et de respect de la vie privée des utilisateurs, ces applications sont distribuées. Pour préserver ces propriétés, il est important que le système de recommandation qui leur est associé le soit également. Une grande majorité des travaux dans ce domaine est réalisée dans un environnement centralisé, et ne peut donc pas être adaptée simplement pour les réseaux distribués.

Dans cette section, nous proposons une approche distribuée au problème de prédiction de lien décrit dans la section précédente. Cependant, dans ce cas, nous considérons que le réseau social est connu. Chaque pair du système peut communiquer avec ses voisins dans le réseau social. Par contre, il ne connaît que ses propres voisins.

Contributions

Nous proposons un algorithme décentralisé, nommé SoCS, basé sur un plongement du graphe social dans un espace de coordonnées. SOCS s'appuie sur des techniques de dessin de graphe: chaque pair calcule des coordonnées dans l'espace social. Grâce à ces coordonnées, il est ensuite capable de calculer une distance sociale vis à vis des autres utilisateurs et donc de prédire s'ils sont proches ou non. SOCS préserve la structure communautaire du graphe social, ce qui le rend efficace dans le cadre de la prédiction de liens. Il s'appuie sur des protocoles épidémiques largement étudiés dans le domaine du P2P, et résiste parfaitement aux dynamiques observées dans les systèmes distribués. De plus, nous montrons qu'en adaptant SOCS aux spécificités des systèmes distribués, nous améliorerons également la précision des coordonnées sociales. Nous évaluons SOCS sur trois types de graphes différents :

- un réseau de collaborations entre terroristes
- un réseau de collaborations scientifiques (DBLP)
- un graphe petit monde synthétisé

[1] `http://www.joindiaspora.com`

Modèles de forces pour le dessin de graphes

Le plongement de graphe consiste à attribuer à chaque sommet du graphe des coordonnées. Souvent, l'objectif d'un plongement est de faire correspondre la distance entre ces coordonnées à la distance du plus court chemin dans le graphe. Notre objectif avec SoCS est précisément l'inverse. Nous sommes intéressés par les différences entre la distance dans le graphe et la distance entre les coordonnées (appelée distorsion), car c'est ce qui va nous indiquer la séparation entre les communautés d'utilisateurs.

De nombreux travaux cherchent à dessiner des graphes de façon à ce qu'ils soient facilement compréhensibles. Il s'agit d'un cas particulier de plongement de graphe, dans lequel l'espace de coordonnées a deux ou trois dimensions. Ces méthodes ont fréquemment recours à des modèles de forces pour placer les sommets. Deux types de forces sont appliquées sur le graphe:

- Les sommets sont attirés par leurs voisins dans le graphe. Cela signifie que les voisins du graphe doivent être dessinés côte à côte. Cette force permet de préserver les structures de communautés dans la représentation. Mais si elle était utilisée seule, tous les sommets seraient dessinés précisément au même endroit.

- Les sommets sont repoussés par tous les autres sommets. Cette force permet d'écarter les sommets dans la représentation, et indique que deux sommets qui ne sont pas bien connectés dans le graphe doivent être distants.

Ces algorithmes ont été largement étudiés, et de nombreux modèles de forces ont été proposés. Nous nous intéressons particulièrement à deux d'entre eux. Le modèle LinLog s'appuie sur des propriétés théoriques et a été conçu pour représenter les communautés. Le modèle HC s'inspire des modèles physiques des particules chargées et des ressorts.

Contrairement au cas du dessin de graphe, SoCS n'est pas limité à trois dimensions. Toutefois, afin de favoriser la distorsion et de rendre les coordonnées compactes lors de leur transmission sur le réseau, nous travaillons dans des espaces de taille réduite, par exemple dix dimensions.

SoCS

SoCS s'inspire des algorithmes de dessin de graphe. À chaque étape du protocole, chaque pair calcule les forces d'attraction et de répulsion qui lui sont appliquées, et ajuste sa position.

Les attractions sont calculées par rapport à un nombre réduit de voisins du graphe et ne posent par conséquent pas de difficulté. Par contre, il n'est pas envisageable de communiquer avec tous les pairs du réseau pour calculer les forces de répulsion : le coût réseau serait bien trop élevé. SoCS s'appuie sur une propriété importante des modèles de forces que nous considérons : les forces de répulsion diminuent rapidement avec la distance. Par conséquent, SoCS ignore tout simplement les répulsions des pairs distants dans l'espace social, en considérant que leur impact est négligeable. Il suffit alors de maintenir une liste des r voisins les plus proches dans l'espace social en utilisant des algorithmes épidémiques de "clustering" pour pouvoir calculer les forces avec un coût raisonnable.

Une autre différence importante avec les algorithmes de dessin de graphe vient du fait que le réseau P2P est maintenu de façon permanente. Dans un dessin de graphe, les sommets sont initialisés à des positions aléatoires. Par contre, dans SoCS, le graphe social évolue au cours de la vie du réseau, et des pairs peuvent rejoindre le réseau. Dans ce cas, un pair initialise ses coordonnées au centre de gravité des coordonnées de ses voisins du graphe. Cela lui permet de commencer à

une position proche de l'idéal, et de converger plus rapidement, en évitant d'être bloqué dans un minimum d'énergie local.

Évaluation

Dans un premier temps, nous évaluons SOCS dans le cadre d'une application de prédiction de liens. Pour cela, nous considérons deux graphes, un réseau terroriste de 152 sommets et un réseau de scientifiques de 211018 sommets. Nous comparons les performances de SOCS avec celles de trois algorithmes couramment utilisés dans la prédiction de liens :

- le plus court chemin dans le graphe (SP)
- le nombre de voisins en commun (CN)
- les "hierarchical random graphs" (HRG), proposés par Clauset, Moore et Newman

Les performances de SOCS sont supérieures à celles de SP et HRG, mais inférieures à celles de CN. Nos résultats confirment les travaux existants, qui indiquent qu'une approche aussi simple que CN est souvent plus efficace que des algorithmes plus élaborés. Toutefois, il est important de noter que dans SoCS, les pairs n'échangent jamais leurs listes de voisins. SOCS ne calcule pas non plus de plus court chemin dans un graphe. Cette opération est particulièrement coûteuse dans un environnement décentralisé. Enfin, il est très intéressant de remarquer que SOCS est plus efficace lorsque r, le nombre de répulsions prises en compte, est petit. Ce paramètre est très important puisqu'il permet à la fois de réduire le coût réseau du protocole, mais aussi d'améliorer la précision des résultats. Dans nos expériences, HC se montre plus précis que LinLog.

Ensuite, nous générons des graphes petits mondes suivant le modèle de Kleinberg afin de vérifier l'adéquation entre la distance sociale théorique et celle obtenue par SoCS. Nos expériences montrent que lorsque r est petit, la distance vers les voisins proches est mieux respectée. Ces résultats confirment ceux obtenus dans le cadre de la prédiction de liens : il est beaucoup plus important de bien positionner les voisins proches pour avoir des recommandations précises. Par conséquent, en limitant le nombre de répulsions, SOCS élimine le bruit causé par les pairs lointains et améliore les résultats. Nous évaluons également SOCS dans le cadre d'un réseau dynamique. Nos mesures indiquent que SOCS converge plus rapidement avec le modèle HC et lorsque r est petit.

Conclusion

Nous avons décrit SoCS, un algorithme de plongement de graphe qui apporte une approche complètement décentralisée au problème de prédiction de liens. Plutôt que de simplement décentraliser un algorithme existant, SOCS apporte une nouvelle approche originale à la prédiction de liens. À notre connaissance, il s'agit du seul algorithme de prédiction de liens basé sur des modèles de forces. Les performances de SOCS sont comparables aux algorithmes existants, et nos résultats montrent que SOCS est particulièrement adapté à un environnement P2P, puisqu'il converge vite et que son coût est très réduit.

Moteur de recherche personnalisé et décentralisé

Contexte

Les sites Internet modernes, comme Flickr, Delicious et CiteULike, s'appuient sur des données "sociales" renseignées par les utilisateurs. Ce modèle de fonctionnement, appelé "folksonomy", offre aux utilisateurs une liberté complète pour décrire les éléments du site Internet. Les utilisateurs "taguent" les éléments du site avec un vocabulaire qui leur est propre. Cette liberté est une source de richesse, puisque chaque utilisateur peut enrichir le site Web avec sa connaissance, mais elle est également un frein à la navigation. L'information est non structurée (contrairement à une taxonomie ou ontologie), si bien qu'il est très difficile de trouver une réponse satisfaisante à une recherche.

Contributions

Nous proposons GQE, un système décentralisé qui permet une recherche d'information personnalisée dans une folksonomie. GQE s'appuie sur GMIN, un protocole P2P qui crée un réseau sémantique d'utilisateurs partageant des centres d'intérêt. Contrairement à un réseau social au sens habituel, les voisins sémantiques permettent de découvrir d'autre tags et éléments, et ainsi d'avoir une recherche plus efficace. GMIN s'attache particulièrement à couvrir tous les centres d'intérêt d'un utilisateur, de façon à pouvoir satisfaire tous les types de requêtes. Il est également économe en ce qui concerne l'utilisation du réseau. Les profils des utilisateurs, contenant l'information sur leurs intérêts peuvent être particulièrement volumineux, c'est pourquoi GMIN utilise des résumés de ces profils sous la forme de filtres de Bloom. Ainsi, l'information complète n'est échangée que lorsque les utilisateurs sont certains qu'ils bénéficieront de cet échange. GMIN est un protocole complètement distribué, qui permet d'offrir aux utilisateurs un anonymat dont ils ne pourraient pas bénéficier dans un système centralisé. Enfin, nous proposons également un système d'extension de requêtes personnalisé de façon à améliorer les résultats d'une recherche.

Sélection des voisins

GMIN génère, pour chaque utilisateur, un ensemble de voisins sémantiques appelé GNET. Pour cela, GMIN doit identifier les utilisateurs ayant des intérêts communs. Cela nécessite la création d'une métrique qui permet de détecter la similarité entre des profils d'utilisateurs. Les métriques classiques basées sur l'utilisation de tags en commun, ou le nombre de centres d'intérêt en commun, ne sont pas satisfaisantes. En effet, comme GMIN est un système décentralisé et personnalisé, la taille du GNET doit rester faible. Cela pose problème lorsqu'un utilisateur a plusieurs centres d'intérêt différents. Sélectionner les utilisateurs ayant le plus d'intérêts en commun pousse à ignorer les intérêts mineurs et ceux-ci sont sous-représentés dans le GNET. Par conséquent, nous proposons une nouvelle façon de sélectionner les voisins sémantiques. Chaque utilisateur tient compte des voisins précédemment sélectionnés, de façon à s'assurer que la représentation de ses intérêts sera équilibrée. Si un intérêt est sous-représenté, alors un voisin partageant cet intérêt sera favorisé pour faire partie du GNET.

La construction du GNET se fait par algorithmes épidémiques. Les utilisateurs échangent l'information sur leurs voisins de façon à continuellement améliorer leur GNET. Comme le profil d'un utilisateur est volumineux, ce protocole aurait un coût réseau très élevé si les profils étaient envoyés à chaque échange. Dans GMIN, chaque utilisateur construit un résumé de son profil sous la forme d'un filtre de Bloom des éléments qui l'ont intéressé. Ce filtre de Bloom est beaucoup moins

volumineux que le profil entier et fournit une approximation précise du contenu du profil. Lorsqu'un voisin est considéré comme faisant partie du GNET pendant une période suffisamment longue, alors son profil est téléchargé de façon à bénéficier de sa connaissance.

GMIN est un protocole décentralisé qui permet un anonymat des utilisateurs. En effet, les utilisateurs ne souhaitent pas forcément pouvoir être associés au contenu de leur profil. Pour cela, nous modifions la procédure épidémique de création du GNET en introduisant la présence d'un intermédiaire. Chaque utilisateur sélectionne au hasard un intermédiaire qui assure la construction du GNET à sa place. De cette façon, les personnes ajoutées au GNET ne connaissent pas l'utilisateur mais son intermédiaire. L'intermédiaire lui-même ne communique pas directement avec l'utilisateur, leurs communications sont cryptées, et passent par plusieurs pairs relais, de la même façon que les systèmes comme Tor protègent l'anonymat des utilisateurs. Nous optimisons également les transferts réseau de façon à limiter le volume des informations transmises par les relais.

Nous évaluons la procédure de création du GNET par des simulations, ainsi qu'un déploiement sur la plate-forme de test Planetlab. Pour évaluer la qualité du GNET, nous mesurons sa capacité à fournir à un utilisateur des éléments qui l'intéressent. Nos mesures, effectuées sur des données réelles, indiquent que la métrique utilisée par GMIN peut améliorer les résultats de 17% à 69% suivant les données utilisées. Ces expériences prouvent qu'en prenant en compte l'ensemble des intérêts d'un utilisateur, GMIN améliore grandement la proportion de requêtes qu'il est capable de satisfaire. Nos expériences en simulation et sur Planetlab indiquent également que GMIN converge rapidement (20 cycles d'échange), et consomme peu de bande passante (20kbps au maximum).

Extension de requêtes

Les moteurs de recherche internet bénéficient tous d'un système d'extension de requête. Lorsqu'une requête est envoyée, elle est complétée par l'ajout de synonymes, de façon à fournir plus de résultats à l'utilisateur. Ce mécanisme est, dans la majorité des cas, le même pour tous les utilisateurs. Nous proposons GQE, un mécanisme d'extension de requêtes personnalisé basé sur l'information contenue dans le GNET. Grâce au profils contenus dans son GNET, chaque utilisateur génère sa propre vision des relations entre les tags, et enregistre cette information dans une structure nommée TAGMAP. Nos expériences indiquent qu'utiliser une vision personnalisée des relations entre tags, par opposition à une vision globale, est suffisant pour améliorer la qualité de l'extension de requêtes. Nos expériences montrent qu'avec GQE, les utilisateurs trouvent les réponses à leurs recherches plus facilement. Mais GQE, a également un autre avantage. Comme notre système est décentralisé, chaque utilisateur peut, avec son ordinateur, contribuer aux calculs liés à sa recherche. Par conséquent, GQE, peut utiliser des mécanismes plus coûteux en calculs qu'un moteur de recherche classique. Nous utilisons à nouveau des mesures de centralité relative. Nos expériences montrent que son utilisation permet d'assigner des poids aux éléments de la requête, et ceux-ci, une fois pris en compte par le moteur de recherche, améliorent la précision des résultats. Cette démarche est uniquement possible dans le cadre de GQE, un moteur de recherche centralisé ne serait pas capable d'effectuer tous les calculs nécessaires pour de nombreux utilisateurs.

Conclusion

Avec GMIN et GQE, nous proposons une nouvelle approche à la navigation Internet. Notre système est décentralisé et met l'utilisateur au coeur du processus de recherche. Ce mécanisme permet

d'améliorer la pertinence des résultats qui lui sont proposés, et aussi de protéger sa vie privée en masquant l'association entre son identité et son profil.

Conclusion et perspectives

Les plates-formes sociales permettent de personnaliser la navigation Internet. Cependant, les plates-formes actuelles posent de sérieux problèmes de respect de la vie privée et de passage à l'échelle. Nous étudions ces aspects à travers trois contributions différentes ;

- Nous proposons une approche générique qui permet, à partir d'information sociale, de prédire un réseau social gardé secret pour des raisons de vie privée. Notre méthode peut être utilisée pour évaluer à quel point des données sont sensibles. Nos résultats mettent en évidence le danger que représentent les plates-formes centralisées pour la vie privée des utilisateurs.
- Nous proposons SOCS un algorithme décentralisé qui permet de prédire des liens dans un graphe social. Notre approche s'inspire des travaux de dessins de graphe et peut être utilisée dans le cadre d'un réseau social décentralisé.
- Nous proposons GMIN, un protocole épidémique P2P qui permet de regrouper les utilisateurs ayant des intérêts en commun. Cette information est ensuite utilisée par GQE, un système d'expansion de requêtes qui personnalise les recherches des utilisateurs et améliore la qualité des résultats. Notre système protège l'identité des utilisateurs en masquant l'association entre un profil et l'identité de l'utilisateur.

Dans nos travaux futurs, nous envisageons d'adapter SOCS aux graphes bipartites qui modélisent les interactions entre les utilisateurs et les contenus. Ainsi, il serait possible d'obtenir des coordonnées pour des utilisateurs et des films dans un même espace. Il serait également intéressant de modifier GQE afin de pouvoir paramétrer le niveau de personnalisation souhaité par l'utilisateur. Enfin, nous envisageons d'implémenter nos algorithmes au sein de diaspora, une plate-forme de réseaux sociaux P2P récemment créée.

BIBLIOGRAPHY

[1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, pages 280–290, 2003.

[2] L.A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[3] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. 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, 2007.

[4] S. Amer-Yahia, M. Benedikt, L. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. In *Proceedings of the 34st International Conference on Very Large Databases (VLDB '08)*, pages 710–721, 2008.

[5] Y. Aumann and Y. Rabani. An O (log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998.

[6] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*, pages 181–190, 2007.

[7] X. Bai, M. Bertier, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. Gossiping personalized queries. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT '10)*, pages 87–98, 2010.

[8] F. Banaei-Kashani and C. Shahabi. Swam: a family of access methods for similarity-search in peer-to-peer data networks. In *Proceedings of the 2004 ACM Conference on Information and Knowledge Management (CIKM '04)*, pages 304–313, 2004.

[9] J. Bar-Ilan. Google bombing from a time perspective. *Journal of Computer-Mediated Communication*, 12(3):910–938, 2007.

[10] J. Baumes, M. Goldberg, M. Hayvanovych, M. Magdon-Ismail, W. Wallace, and M. Zaki. Finding hidden group structure in a stream of communications. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics, (ISI '06)*, pages 201–212, 2006.

[11] M. Bender, T. Crecelius, M. Kacimi, S. Miche, J. Xavier Parreira, and G. Weikum. Peer-to-peer information search: Semantic, social, or spiritual? *TCDE*, 30(2):51–60, 2007.

[12] M. Bender, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. Xavier Parreira, R. Schenkel, and G. Weikum. Exploiting social relations for query expansion and result ranking. In *ICDE Workshops*, pages 501–506, 2008.

[13] M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Proceedings of the 11th International Middleware Conference (Middleware '10)*, pages 191–211, 2010.

[14] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *CACM*, 13(7):422–426, 1970.

[15] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: byzantine resilient random membership sampling. In *Proceedings of the 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 145–154, 2008.

[16] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

[17] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[18] M. Carman, M. Baillie, and F. Crestani. Tag data and personalized information retrieval. In *Proceedings of the CIKM Workshop on Search in Social Media, (SSM '08)*, pages 27–34, 2008.

[19] C. Cattuto, D. Benz, A. Hotho, and G. Stumme. Semantic analysis of tag similarity measures in collaborative tagging systems. In *Proceedings of the 3rd Workshop on Ontology Learning and Population, (OLP3)*, pages 39–43, 2008.

[20] M. Cha, A. Mislove, and K.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, 2009.

[21] R. Chellappa and A. Jain. Markov random fields: theory and application. *Boston Academic Press*, 1993.

[22] G. Chen, C. P. Low, and Z. Yang. Enhancing search performance in unstructured p2p networks based on users' common interest. *TPDS*, 19(6):821–836, 2008.

[23] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

[24] V. Cholvi, P. Felber, and E. Biersack. Efficient search in unstructured peer-to-peer networks. In *SPAA*, pages 271–272, 2004.

[25] A. Clauset, C. Moore, and M. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

[26] B.F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. PNUTS: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment*, 1(2):1277–1288, 2008.

[27] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. In *AP2PC*, pages 1–13, 2004.

[28] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM*, pages 15–26, 2004.

[29] P. Domingos and M. Richardson. Markov logic: A unifying framework for statistical relational learning. In *Proceedings of the ICML '04 Workshop on Statistical Relational Learning and its Connections to Other Fields*, pages 49–54, 2004.

[30] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[31] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[32] N. Eagle and A. Pentland. Reality mining: Sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[33] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222, 2003.

[34] A. Eyal and A. Gal. Self organizing semantic topologies in p2p data integration systems. In *ICDE*, pages 1159–1162, 2009.

[35] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[36] P. Fraigniaud, E. Lebhar, and Z. Lotker. Recovering the long-range links in augmented graphs. In *SIROCCO*, page 118, 2008.

[37] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement, 1991.

[38] J. Gao and P.-N. Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM '06)*, pages 212–221, 2006.

[39] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Machine Learning*, 3:679–707, 2003.

[40] O. Goussevskaia, M. Kuhn, M. Lorenzi, and R. Wattenhofer. From web to map: Exploring the world of music. In *WI-IAT*, pages 242–248, 2008.

[41] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 1970.

[42] S. B. Handurukande, A.-M. Kermarrec, F. Le Fessant, L. Massoulie, and S. Patarin. Peer Sharing Behaviour in the eDonkey Network, and Implications for the Design of Server-less File Sharing Systems. In *EuroSys*, pages 359–371, 2006.

[43] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *PVLDB*, 1(1):102–114, 2008.

[44] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *ESWC*, pages 411–426, 2006.

[45] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *PKDD*, pages 506–514, 2007.

[46] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *TOCS.*, 25(3):8, 2007.

[47] H. Jie and Y. Zhang. Personalized faceted query expansion. In *SIGIR*, 2006.

[48] H. Jin, X. Ning, and H. Chen. Efficient search for peer-to-peer information retrieval using semantic small world. In *Proceedings of the 15th International Conference on World Wide Web (WWW '06)*, pages 1003–1004, 2006.

[49] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. In *STOC*, pages 561–568, 2004.

[50] A.-M. Kermarrec. Challenges in personalizing and decentralizing the Web: an overview of GOSSPLE. In *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS '09)*, pages 1–16, 2009.

[51] M. Khambatti, K. D. Ryu, and P. Dasgupta. Structuring peer-to-peer networks using interest-based communities. In *DBISP2P*, pages 48–63, 2003.

[52] J Kleinberg. The small-world phenomenon: An algorithmic perspective. In *STOC*, pages 163–170, 2000.

[53] J. Kleinberg. Complex networks and decentralized search algorithms. In *ICM*, 2006.

[54] Y. Koren. Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications*, 49(11-12):1867–1888, 2005.

[55] V. E. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2002.

[56] S. Le-Blond, J.-L. Guillaume, and M. Latapy. Clustering in p2p exchanges and consequences on performances. In *IPTPS*, pages 193–204, 2005.

[57] V. Leroy, B.B. Cambazoglu, and F. Bonchi. Cold start link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, pages 393–402, 2010.

[58] M. Li, A. Lee, W.-C. andSivasubramaniam, and J. Zhao. Ssw: A small-world-based overlay for peer-to-peer search. *TPDS*, 19(6):735–749, 2008.

[59] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *ASIS&T*, 58(7), 2007.

[60] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:577–591, 1994.

[61] M.J. McGill and G. Salton. *Introduction to modern information retrieval*. 1986.

[62] A. Mislove, K. P. Gummadi, and P. Druschel. Exploiting social networks for internet search. In *HotNets*, 2006.

[63] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, pages 29–42, 2007.

[64] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proceedings of the 13th IEEE Symp. on Security and Privacy*, pages 173–187, 2009.

[65] S. Niwa, Takuo Doi, and S. Honiden. Web page recommender system based on folksonomy mining for itng '06 submissions. In *ITNG*, pages 388–393, 2006.

[66] A. Noack. An energy model for visual graph clustering. In *Graph Drawing*, pages 425–436, 2003.

[67] A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 2009.

[68] J. O'Madadhain, J. Hutchins, and P. Smyth. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Exploration Newsletter*, 7(2):23–30, 2005.

[69] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[70] J. A. Patel, É. Rivière, I. Gupta, and A.-M. Kermarrec. Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*, 53(13):2304–2320, 2009.

[71] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-nearest neighbors in uncertain graphs. In *Proceedings of the VLDB, the 36th International Conference on Very Large Databases (PVLDB 2010)*, volume 3, 2010.

[72] F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML*, 1998.

[73] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '94)*, pages 345–354, 1994.

[74] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323, 2000.

[75] J. Sedmidubsky, S. Barton, V. Dohnal, and P. Zezula. Adaptive approximate similarity searching through metric social networks. In *Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE '08)*, pages 1424–1426, 2008.

[76] A. Singla and I. Weber. Camera brand congruence in the flickr social graph. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM '09)*, pages 252–261, 2009.

[77] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. *INFOCOM*, 3:2166–2176, 2003.

[78] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *Transactions on Networking*, 11(1):17–32, 2003.

[79] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer, 2001.

[80] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *SP*, pages 44–59, 1997.

[81] B. Taskar, M.F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Neural Information Processing Systems*, volume 15, 2003.

[82] W. M. P. Van Der Aalst, H. A. Reijers, and M. Song. Discovering social networks from event logs. *Computer Supported Cooperative Work*, 14(6):549–593, 2005.

[83] F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. In *INFOVIS*, pages 199–206, 2004.

[84] S Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In *EuroPar*, pages 1143–1152, 2005.

[85] P. M. Winkler. Proof of the squashed cube conjecture. *Combinatorica*, 3(1):135–139, 1983.

[86] B. Wong, A. Slivkins, and Sirer E. G. Approximate matching for peer-to-peer overlays with cubit. Technical report, Cornell University, Computing and Information Science Technical Report, 2008.

[87] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In *SIGIR*, pages 155–162, 2008.

[88] H. Yildirim and M. S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys*, pages 131–138, 2008.

[89] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman. Sybilguard: defending against sybil attacks via social networks. *TON*, 16(3):576–589, 2008.

[90] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, pages 609–616, 2001.

[91] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, pages 694–699, 2002.

[92] V. Zanardi and L. Capra. Social ranking: uncovering relevant content using tag-based recommender systems. In *RecSys*, pages 51–58, 2008.

[93] J. Zhang and Y. Yang. Probabilistic score estimation with piecewise logistic regression. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML '04)*, page 115, 2004.

[94] S. Zhao, N. Du, A. Nauerz, X. Zhang, Q. Yuan, and R. Fu. Improved recommendation based on collaborative tagging behaviors. In *IUI*, pages 413–416, 2008.

[95] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*, pages 531–540, 2009.

Résumé

L'émergence du "Web 2.0" a fondamentalement changé le comportement des utilisateurs d'Internet. Le Web est devenu une plates-forme collaborative. En indiquant leurs préférences et en partageant des informations privées, les utilisateurs bénéficient d'un contenu personnalisé.
Cependant, ces systèmes posent des problèmes au niveau du respect de la *vie privée* et du *passage à l'échelle*. Les plates-formes sociales divulguent ces informations personnelles dans un but commercial. De plus, ces systèmes centralisés nécessitent des serveurs très coûteux. Par conséquent, les plates-formes sociales existantes ne tirent pas complètement profit des possibilités de personnalisation.
Dans cette thèse, nous nous intéressons à la conception de réseaux sociaux et de services tirant partie des informations personnelles dans le contexte de réseaux *pair-à-pair* (P2P). Les réseaux P2P sont des architectures décentralisées, par conséquent les utilisateurs contribuent au service et gardent le contrôle de leurs donnés personnelles. Cette architecture résout donc en partie le problème du passage à l'échelle et préserve mieux la vie privée des utilisateurs. Cependant, comme les données sont distribuées, il devient plus difficile de localiser les informations intéressantes.
Dans cette thèse, nous présentons les contributions suivantes. Nous proposons une approche pour prédire un réseau social à partir des informations personnelles des utilisateurs. Notre méthode repose sur l'utilisation d'un *graphe probabiliste*. Nous évaluons ses performances sur des donnés de Flickr en utilisant les groupes d'intérêt comme information disponible. Nos résultats montrent qu'il est possible de prédire un réseau social tenu secret à partir de peu d'information, et justifient donc le passage à une solution décentralisée.
Nous proposons SoCS, un algorithme *décentralisé* de *prédiction de liens*. La recommandation de contacts est une fonctionnalité centrale des réseaux sociaux, notre algorithme est donc une première étape vers leur décentralisation. SoCS s'appuie sur des protocoles épidémiques pour réaliser un plongement de graphe distribué. Les coordonnées sociales obtenues permettent ensuite de prédire des liens entre les utilisateurs. Nous montrons que SoCS est particulièrement adapté aux systèmes distribués.
Nous proposons GMIN, une plates-forme *décentralisée* tirant profit des centres d'intérêt des utilisateurs. GMIN fournit à chaque utilisateur une liste de voisins intéressés par les mêmes sujets. Notre algorithme fait attention à prendre en compte l'ensemble des intérêts d'un utilisateur, et pas seulement les principaux. Nous utilisons ensuite ces donnés pour générer des *requêtes personnalisées* (GQE). Chaque requête est étendue de façon à prendre en compte le point de vue de l'utilisateur et de ses voisins. Nous montrons que ce mécanisme améliore la pertinence des résultats.

Mot-clés : algorithmes épidémiques, expansion de requêtes, folksonomy, information sociale, pair-à-pair, personnalisation, plongement de graphe, protection de la vie privée, réseaux sociaux, systèmes distribués

Abstract

The so-called Web 2.0 revolution has fundamentally changed the way people interact with the Internet. The Web has turned from a read-only infrastructure to a collaborative platform. By expressing their preferences and sharing private information, the users benefit from a personalized Web experience.
Yet, these systems raise several problems in terms of *privacy* and *scalability*. The social platforms use the user information for commercial needs and expose the privacy and preferences of the users. Furthermore, centralized personalized systems require costly data-centers. As a consequence, existing centralized social platforms do not exploit the full extent of the personalization possibilities.
In this thesis, we consider the design of social networks and social information services in the context of *peer-to-peer* (P2P) networks. P2P networks are decentralized architecture, thus the users participates to the service and control their own data. This greatly improves the privacy of the users and the scalability of the system. Nevertheless, building social systems in a distributed context also comes with many challenges. The information is distributed among the users and the system has be able to efficiently locate relevant data.
The contributions of this thesis are as follow.
We define the *cold start link prediction* problem, which consists in predicting the edges of a social network solely from the social information of the users. We propose a method based on a *probabilistic graph* to solve this problem. We evaluate it on a dataset from Flickr, using the group membership as social information. Our results show that the social information indeed enables a prediction of the social network. Thus, the centralization of the information threatens the privacy of the users, hence the need for decentralized systems.
We propose SoCS, a *decentralized* algorithm for *link prediction*. Recommending neighbors is a central functionality in social networks, and it is therefore crucial to propose a decentralized approach as a first step towards P2P social networks. SoCS relies on gossip protocols to perform a force-based embedding of the social networks. The social coordinates are then used to predict links among vertices. We show that SoCS is adapted to decentralized systems at it is churn resilient and has a low bandwidth consumption.
We propose GMIN, a *decentralized* platform for *personalized services* based on social information. GMIN provides each user with neighbors that share her interests. The clustering algorithm we propose takes care to encompass all the different interests of the user, and not only the main ones. We then propose a personalized *query expansion* algorithm (GQE) that leverages the GMIN neighbors. For each query, the system computes a tag centrality based on the relations between tags as seen by the user and her neighbors. We show that this improves the recall and the precision of the user's queries.

Keywords: distributed systems, foksonomy, gossip, graph embedding, peer-to-peer, personalization, privacy, query expansion, social information, social networks

N° ordre: XXXXXXXX

Institut National des Sciences Appliquées de Rennes
20, Avenue des Buttes de Coësmes . CS 14315 . F-35043 Rennes Cedex
Tel : 02 23 23 82 00 – Fax : 02 23 23 83 96