

*Personalized Web Search by Gossiping with
Unknown Social Acquaintances*

Marin Bertier — Davide Frey — Rachid Guerraoui

Anne-Marie Kermarrec — Vincent Leroy

N° **6878**

Mars 2009

Thème COM



*Rapport
de recherche*

Personalized Web Search by Gossiping with Unknown Social Acquaintances

Marin Bertier^{*†}, Davide Frey^{*}, Rachid Guerraoui[‡]
Anne-Marie Kermarrec^{*}, Vincent Leroy^{*†}

Thème COM — Systèmes communicants
Équipe-Projet ASAP

Rapport de recherche n° 6878 — Mars 2009 — 30 pages

Abstract: Social networking and collaborative tagging have taken off at an unexpected scale and speed. Huge opportunities to significantly boost the search experience are out there in the Web but the amount of information to be dissected is seemingly Herculean. Moreover, users might be reluctant to publicize their profiles in order to facilitate the navigation of other users.

We present GOSSPLE, the first decentralized system to personalize the user search experience by expanding queries with information derived from anonymous social acquaintances. Underlying GOSSPLE lies the intuition that, while social networks can provide you with news from your old buddies, you can learn a lot more from people you do not know, but with whom you share many interests. Considering a collaborative tagging system with active participants annotating content, GOSSPLE manages each user profile and dynamically creates her personalized "social" network by gossiping and computing a distance between users, without revealing which profile is associated with which user. Using the information in this personalized social network, each user extracts knowledge about the relations between tags which she locally leverages to improve her own search experience through a personalized query expansion mechanism.

We evaluate GOSSPLE on traces crawled from CiteUlike and Delicious, with 33,834 and 20,000 users. We do so in a real distributed system of 170 PlanetLab nodes as well as by simulating a large-scale system involving thousands of peers. In short, we show that by sharing their tagging behaviors with small numbers of neighbors, users benefit from personalized and efficient query expansion, increasing the number of query results (recall) while significantly improving on quality (precision).

Key-words: peer-to-peer, overlay, query expansion, search engine, personalization

This work is supported by the ERC Starting Grant 204742

* ASAP research group

† INSA Rennes

‡ EPFL, Switzerland

Recherche web personnalisée grâce à la découverte de voisins sémantiques

Résumé : Les réseaux sociaux et les sites web collaboratifs ont connu un essor incroyable. Ils offrent des opportunités d'améliorer la recherche d'information sur Internet, mais la quantité de données à traiter est gigantesque. De plus, les utilisateurs sont peu enclins à publier leurs profils personnels pour faciliter la navigation des autres utilisateurs.

Nous présentons GOSSPLE, le premier système décentralisé qui personnalise les recherches en ajoutant à ses requêtes des termes appropriés issus de voisins sémantiques anonymes et détectés automatiquement. GOSSPLE repose sur l'idée que les réseaux sociaux permettent de rester en contact avec des amis, mais qu'on apprend beaucoup plus des personnes que l'on ne connaît pas mais qui ont les mêmes centres d'intérêt. Dans le cadre d'un système collaboratif dans lequel les utilisateurs annotent (taguent) du contenu, GOSSPLE gère chaque profil d'utilisateur et crée dynamiquement son réseau "social" personnalisé grâce à des algorithmes épidémiques et calcule la distance entre les utilisateurs sans révéler les associations entre les profils et les utilisateurs. En s'appuyant sur les informations du réseau social personnalisé, chaque utilisateur découvre les relations entre les tags et les utilise pour améliorer ses requêtes à travers un mécanisme d'expansion de requêtes personnalisé.

Nous évaluons GOSSPLE sur des traces des sites web Delicious et CiteULike qui contiennent respectivement les profils de 33.834 et 20.000 utilisateurs. Nos expériences reposent sur un déploiement sur un système réel de 170 machines PlanetLab ainsi que sur des simulations de réseaux large échelle comprenant des milliers d'utilisateurs. Nous montrons qu'en partageant leurs informations avec un petit nombre de voisins sémantiques, les utilisateurs bénéficient d'une expansion de requêtes personnalisée efficace, qui augmente le nombre de résultats tout en augmentant significativement la qualité.

Mots-clés : pair-à-pair, overlay, expansion de requête, moteur de recherche, personnalisation

1 Introduction

The Web has turned from a read-only infrastructure with passive participants into a read-write platform with active players. The content of the Web is no longer generated only by experts but pretty much by everyone (YouTube, Flickr, Last.fm, delicious, etc). Like any popular revolution, this goes through democratizing the language: instead of subject indexing with a controlled vocabulary, i.e., *ontology*, freely chosen keywords are used to *tag* billions of items. The user-generated taxonomy is called *folksonomy* (folk + taxonomy) and is used to label and share newly user-generated content (*e.g.* photographs), or to collaboratively label existing content (*e.g.* Web sites, books, or blog entries). As a result of this growing set of user-generated content, a gold mine of information is available on the Web and it is hard to believe that any search request of a given user does not have a perfectly matching reply somewhere in that mine.

Part of the appeal of a folksonomy is its inherent subversiveness: folksonomies can be seen as a rejection of the traditional search engine status quo with its orthodoxy of ontologies. Clearly, the success of collaborative networks is related to the freedom left to the users to tag items at will. This freedom can also however be viewed as a drawback. The facts that tags are informally defined, and continually changing makes it very difficult for the tagging behavior of a user to make any sense for another one, nor does it prevent junk tagging and synonyms, which introduce significant noise in the search process. In short, matching a specific query might rapidly reveal like looking for a needle in a haystack.

We believe that salvation can only come from pushing the revolution further. Basically, we argue for a fully user centric approach where every participant stores and controls not only her own items and tagging behavior, but also dynamically maintains a personalized perspective of the portion of the network that is relevant to her own search. This could then be effectively used to expand users' queries with appropriate tags to significantly enhance the search experience.

Existing works have already addressed some user-centric search exploiting the shared interests in social networks to enhance Web search [2, 18, 4]. Such works solve homonyms issues by providing different result sets to *apple* search whether users mean *fruit* or *computer*. Yet, they do not solve more complex user-centric queries. To give an idea underlying our approach, consider the following (real) example. After living for several years in the UK, John is back to Lyon in France and, to maintain his kids' skills in English, is looking for an English speaking baby-sitter who would be willing to trade baby-sitting hours against accommodation. Given the high number of students in Lyon, there is no doubt that such an offer would be of interest to many English speaking students. John's Google request *English baby-sitter Lyon* does not return anything interesting for baby-sitter is immediately associated with daycare or local (French) baby-sitting companies.

His Facebook buddies in Lyon or in the UK cannot really help either for none has ever looked for an English speaking baby sitter in Lyon. Consider now Alice living in Bordeaux, after several years in the US, and who is looking for a similar deal with her kids. Alice is lucky to discover that teaching assistants in primary schools are a very good match as they have the same working hours as kids; they do have a salary but would enjoy living within a family. Now if Alice associates

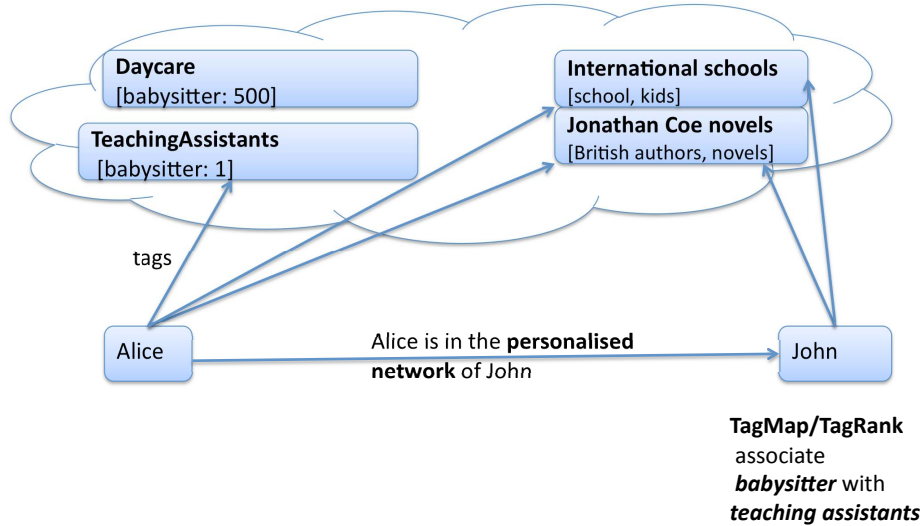


Figure 1: Alice and John example

baby-sitter with *teaching assistant* in her search request, she does indeed find very good candidates. Clearly, if John could reuse Alice's discovery, he would also find good candidates in Lyon. Nevertheless, Alice and John do not know each other nor do they live in the same area, nor even have similar jobs. Yet, their past history made clear their links through the fact that they both lived in English speaking countries and both have kids around the same age and do need baby-sitters. They share some interest for example in international schools and British novels. This is expressed through their tagging behavior as shown in Figure 1. Should a system be able to make the connection between Alice and John, the association between tags *teaching assistant* and *baby-sitter* could be made to expand John's query accordingly and render his request solvable by any search engine.

The fundamental observation here is that, whereas old buddies do not bring much to the search, profiles of unknown users who share similar interests can reveal very effective in expanding users' queries. Putting this idea to work is however challenging and the difficulties are of two different natures. Indeed one needs to expand queries with appropriate tags that help retrieve the appropriate matching results (completeness) but these results should not be drown with tons of useless information (accuracy). At first glance, optimizing that trade-off might go through maintaining a huge amount of personalized information about proximities between tags. This information grows exponentially with the size of the system and the success of social tagging might simply kill the underlying infrastructure. On the other hand, discovering social acquaintances might be hampered by the resistance of users to publicize their tagging behavior. In fact, the seemingly eagerness of companies to benefit from user-generated content might already dissuade users from generating new content and expressing their tagging behavior in an explicit manner.

The motivation of this work is to take up those challenges. In short, GOSSPLE *automatically* infers *personalized* connections between users and provides them

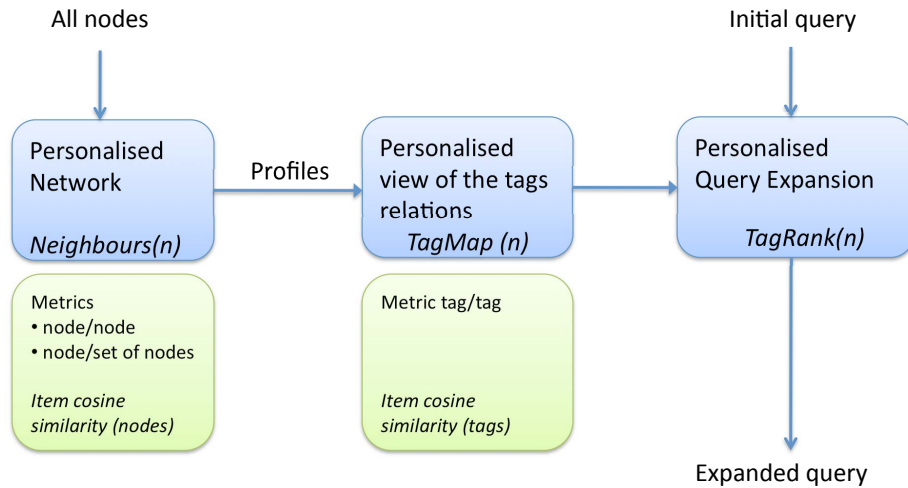


Figure 2: Personalized query expansion in GOSSPLE

with *semantically* related tags as companions to their queries. This is achieved in a fully *decentralized* manner, where user machines (nodes) continuously gossip digests of their tagging behavior in an *anonymous* fashion and locally compute adequate data structures to represent a personalized proximity between tags.

We report on our evaluation of GOSSPLE with a CiteULike trace, involving 33,834 users and a 20,000 user Delicious trace. We do so in a real distributed system of 170 PlanetLab nodes as well as through a simulation of a large scale system involving thousands of nodes. In short, we show that, with little information stored at every node, GOSSPLE enables to retrieve items that cannot be retrieved with state of the art search systems (completeness or recall) while at the same time improving accuracy (precision). More specifically, a query expansion of size 20 retrieves 40% of unsuccessful original queries while improving the precision of 58% of the originally successful queries (against resp. 36% and 24% for competitors) on a delicious trace.

2 GOSSPLE in a nutshell

Overview The goal of GOSSPLE is to expand any query launched by a user with appropriate tags, in order to provide better results than those obtained with the original query, using the same search engine. In short, the query expansion has to be (a) *complete* so that relevant results are added to the result set and (b) *accurate* so that the number of irrelevant results is kept low. These aspects are commonly referred to as *recall* and *precision* [7], respectively.

The idea underlying GOSSPLE is to expand the query of a user by leveraging the tagging profile of her social acquaintances. These are discovered in a dynamic and decentralized manner while preserving the anonymity of the user's profile.

In GOSSPLE, a query is a set of tags and the system seeks to complete this set in order to obtain a complete and yet accurate search. This is achieved by

having every node periodically compute a personalized view of the proximity between tags. Three steps are involved, as depicted in Figure 2:

1. The creation of a personalized network of acquaintances
2. The computation of a personalized view of relations between tags, captured in a data structure (called *TagMap*)
3. The computation of the relative centrality of the tags for a given tag ¹, captured in a data structure we call the *TagRank*. Interestingly the two data structures reveal very complementary: as conveyed by our experiments, the TagMap promotes completeness (recall) whereas the TagRank promotes accuracy (precision).

We give in the following a high-level description of how GOSSPLE achieves these steps.

User profile For the simplicity of presentation, we assume that each user is associated to one machine and we talk about a *node* to denote the pair (user, machine) ². We consider a system composed of a set of nodes N . Nodes tag items from a set I with tags from a set of tags T . The resulting information space (IS) is a set of triplets of the form (n, i, t) , where n is a node name, i an item and t a tag. The *profile* of a node (user) n_k , denoted $profile(n_k)$, is the set of triplets $(n_k, i, t) \in IS$.

Personalized network To identify the relevant acquaintances of a node, we rely on the *item cosine similarity* to compute the distance between nodes. Intuitively, the more two nodes have tagged the same items (not necessarily with the same tags), the closer they are. We adapt this metric to consider several nodes at a time in order to ensure that each node has enough acquaintances in its network to cover its range of interests. The personalized network of a given node n , denoted $Neighbors(n)$, contains the set of the c close nodes to n together with their profile. The size of such a network is typically several orders of magnitude smaller than the entire network: in our experiments, 20 neighbors are enough to achieve effective query expansion in a 33,834 user system.

Gossiping profiles The personalized network is periodically maintained by a decentralized protocol that gossip profiles while hiding their owners. Basically, each node is connected to a random subset of the nodes, provided by a random peer sampling (RPS) protocol [16]. This protocol, which aims at ensuring that the network remains connected, that new nodes are smoothly incorporated to the network while the faulty (or leaving) nodes are gracefully removed, is in turn used by the nodes to gossip their profiles and compute their personalized network. Instead of exchanging the full profiles, which would be too expensive, nodes exchange an approximate digest of their profiles where tags are removed and item lists are encoded via Bloom filters. (In our experiments, the size of such a digest is around 1Kb and nodes gossip messages of size 20Kb - with a

¹While PageRank computes the relative importance of Web pages (the eigenvector centrality) of Web pages, TagRank computes the importance of tags with respect to another tag on a given node, we refer to this notion as the tag centrality in the sequel.

²This can trivially be extended to multiple users per machine

personalized network with 20 acquaintances.) Every node has a proxy which acts on its behalf when gossiping profiles and with whom it shares an encryption key. The node communicates with its proxy through a relay which does not know the key but hides to the proxy the association between the node and its profile. The relay and the proxy are both randomly determined using the random peer sampling protocol. As we show through our experiments, the overhead of ensuring anonymity with this technique reveals very negligible.

TagMap The profiles of the nodes from the personalized network of a node are in turn locally used to build its *TagMap*: a data structure that represents a personalized view of the proximity between pairs of tags. Basically, the TagMap is a matrix $TagMap_n$, where $TagMap_n[t_i, t_j]$ is a value (called score in the sequel) that reflects the distance between tags t_i and t_j as seen by the node n . We use here the item cosine similarity metric applied to tags and the value is computed whenever $Neighbors(n)$ contains a new meaningful information (in a sense we define later).

TagRank The TagMap is itself locally used to compute another data structure, which we call the *TagRank*, and which is then directly used to expand a query with its q closest tags. (In our experiments, we evaluate values of q ranging from 1 to 50). The TagRank can be viewed as an accurate view of the TagMap in the sense that it conveys the proximity between tags using their relative *centrality* (in the sense of eigenvector centrality [1] applied to the local TagMap and from the perspective of a given tag). The algorithm used here is an adaptation of the celebrated PageRank algorithm (at the heart of Google) [7]: instead of building a graph that represents the relative popularity of Web pages, we build a graph of tags with edges weighted by their score derived from the *TagMap*. We then obtain the TagRank which we recursively refine by computing the distance between tags using random walks, along the lines of [7]. To expand a query, we derive from the TagRank a list of tags which are then weighted and submitted to the underlying search engine, together with the tags from the original query.

Limitations As we pointed out, the objective of GOSSPLE is to expand queries by adding new tags. GOSSPLE does not seek to correct errors in queries and replace misspelled tags. As we discuss in the related work section, some complementary approaches can be combined with GOSSPLE for that purpose.

Also, GOSSPLE assumes a system with nodes that might indeed crash or leave the network without any warning but no Byzantine fault-tolerance is considered. Indeed, GOSSPLE naturally copes with certain forms of free-riding and even malicious attitudes. Nodes do need to participate in the gossiping in order to be themselves visible and receive profile information. Furthermore, as we show in our evaluation section, the impact of arbitrary tagging or even, individual malicious tagging is very limited. Yet, malicious nodes can hamper the peer sampling protocol and malicious coalitions can corrupt the personalized networks. Combinations of Byzantine resilient random peer sampling protocols like Brahms [6], together with mechanisms to detect or reduce the effect of malicious tagging behavior would need to be considered for that purpose.

3 Personalized network

Social networks commonly refer to networks composed of explicitly declared friends (e.g., FaceBook). As we argued, and as reported in [4], as far as queries are concerned, explicit friends might not be the best to help the search process. In GOSSPLE, the social network used to expand queries is instead composed of implicit friends, i.e., (possibly unknown) social acquaintances exhibiting similar interests.

In this section we present the gossip protocol through which we discover and maintain these implicit friends to form the node's personalized network. We first present the metrics used to assess behavioral affinities in Section 3.1. The actual gossip protocol is then presented in Section 3.2. In Section 3.3, we describe how we form the personalized network while hiding the associations between users and their profiles.

3.1 Rating users: items cosine similarity

Discovering the social, yet unknown, connections, typically nodes sharing interest, requires every node to compute a distance with other nodes based on their tagging behavior. This is not entirely trivial if we seek for a sensible metric that encompass the multiple interests of a node. For the sake of clarity of the presentation, we first present how nodes can be rated individually then we describe how we rate a set of nodes to cover multiple interests in GOSSPLE.

Rating individuals The most natural metric to compute a distance between individual nodes is the overlap between tagged items. This simple metric suffers from several drawbacks: nodes having a very high tagging activity exhibit a high overlap with any other node, while this does not reflect any specific interest. In addition, the distance between nodes with a relevant metric should not only increase when interests are similar, but it should also decrease if many other interests are not shared. To address this concern, we compute the distance between individual nodes using the widely used *cosine similarity* between items³: the cosine similarity consistently revealed the most satisfactory. We discuss metrics in the Section 6). This can be interpreted as a normalized overlap. Items are represented as vectors in a multidimensional space, the number of dimension being $|I|$. More precisely, the cosine between two vectors of items is defined as follows:

$$\cos(\vec{v}_1, \vec{v}_2) = \frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \times \|\vec{v}_2\|}$$

The item cosine between two nodes is defined as follows:

$$ItemCos(n_1, n_2) = \frac{|Item(\{n_1\}) \cap Item(\{n_2\})|}{\sqrt{|Item(\{n_1\})| \times |Item(\{n_2\})|}}$$

This provides a distance between two nodes and increases when interests are shared and decreases when they are not.

³Note that we conducted experiments with other metrics such as *co-occurrence* [22]

Rating a set of nodes Individual rating enables a node to select its closest nodes according to their tagging behavior. Yet, users may have several areas of interest, which are not related to each other. Consider a node expressing an interest in a new topic. Individual rating does not capture such emerging interest until this interest represents an important proportion of the profile. Rating only individuals may lead to consider only the dominant interest, ignoring minor, potentially important, ones. This is even more an issue for emerging interests: a node discovering a new topic is particularly keen to expand a query, being not familiar with the tags related to that topic yet. In order to provide a node with a set of nodes reflecting together its multiple interests, the personalized network as a whole should be rated. Note that this need is even more stringent for small sizes of personalized networks.

Rating a set of nodes is a trade-off between the extent to which nodes in the set share interests with a given node n , and how close the distribution of the interests in the set (sum of the interests of the users in the set) is close to the one of n . Following the cosine similarity between item vectors, a score computed on node n for a set of nodes is computed as follows:

$$SetItemVect(set) = \sum_{p \in set} \frac{(ItemVect(p) \otimes ItemVect(n))}{\|ItemVect(p)\|}$$

Where \otimes is the term to term multiplication of the vectors, $(a, b) \otimes (c, d) = (a*c, b*d)$. $SetItemVect(set)$ represents the vector of items for the set of nodes in $Neighbors(n)$.

$$SetScore(n, set) = SetItemVector(set) \cdot ItemVect(n) \times \cos(SetItemVector(set), ItemVect(n))^b$$

As in the individual node rating scheme, each node's contribution to items of interest for n is penalized with the norm of the vector in order to avoid taking into account non shared interests. This way, a node tagging many items of no interest for n is not be favored over others. Then, we only look at the items of interest for n . The score of the set ($SetScore$) reflects both the amount of items of interests and their distribution (cosine). b represents the trade-off between the amount of shared interests and a fair distribution that does not favor any item. It is important to notice that for $b = 0$ (therefore no cosine impact), the distribution is not considered. Then, the highest score of a set is exactly the same as the one obtained from the individual rating, the score of a set being equal to the sum of the item cosine of each node in the set.

3.2 Multi-interest clustering protocol

The personalized network of a node n is called $Neighbors(n)$ and is composed of c entries where c is a parameter of the system representing the trade-off between the amount of available information and the personalization degree of this information (in other words, its quality). Each entry contains a node's IP address, a timestamp and its profile. Note that the profile can be either a full profile or a digest of the profile as detailed below.

We assume that each node runs a gossip-based random peer sampling protocol (RPS) [16], providing a node with a random, continuously changing, subset of the network. In a gossip protocol, each node maintains a *view* of a subset of

the network. A gossip protocol consists in a periodic exchange of information between pairs of nodes, characterized by three functions:

- The *PeerSelection* function selects a gossip target
- The *DataExchange* function selects the data to send during the gossip operation
- The *DataProcessing* function processes the data exchanged during the gossip operation.

The clustering protocol builds the personalized network of a node and is implemented on top of the RPS protocol. The RPS protocol provides the bootstrap for the clustering protocol and ensures that new nodes may be discovered when maintaining the personalized network.

During the clustering protocol, each node n encounters another node q has to evaluate its distance to node q in order to assess q 's eligibility to belong to the personalized network of n . In order to reduce message size during the protocol, nodes do not exchange their full profile upon gossip. Instead, they exchange a profile digest, which is an approximation of their full profile. A profile digest is a compact representation of a full profile and is composed of a Bloom filter representing a hash of the items vectors. The Bloom filter provides a reasonably good approximation of the user profile that can be used to compute the item cosine similarity with a negligible error margin. If the value of the cosine between the node's vector and the one inferred from the Bloom filter, nodes are considered close enough. More precisely, if a node remains as an entry in $Neighbors(n)$ for 5 gossip rounds, it is considered as a good candidate and the entire profiles are exchanged. Otherwise, there is no further exchange. This prevents the useless and expensive transfers of entire profiles. Therefore, an entry for a node in the personalized network contains either the full profile or a digest profile with a counter.

For the sake of presentation clarity, we first present a clustering protocol to build and maintain the personalized network composed of the c closest nodes according to the individual user metric. Then we present our actual multi-interest protocol.

Clustering protocol A simple personalized network is composed of the c nodes that best represent its profile according to the metric presented in Section 3.1. To build the personalized network, each node runs a gossip-based clustering protocol. Starting from a random sample (provided by the underlying RPS), the network is refined as follow. The *PeerSelection* function selects g , the oldest (based on the node timestamp) node in its clustering view. The *DataExchange* function sends the c nodes of $Neighbors(n)$ to the gossip target and waits for the c nodes of $Neighbors(g)$. The *DataProcessing* function merges the RPS's view of n , $Neighbors(n)$ and $Neighbors(g)$ and selects the c closest according to the metric described above (item cosine similarity) to form the new $Neighbors(n)$. This process is iterated and converges in a few cycles [15]. The pseudo-code is presented in Algorithm 1.

Multi-interest clustering protocol The multi-interest clustering protocol is similar to the clustering protocol described above, except for the computation

```

Active thread
Do once for each  $T$  time units
begin
     $p = \text{PeerSelection}(\text{view}_n)$ 
    Send  $\text{DataExchange}(\text{view}_n)$  to  $g$ 
    Receive  $\text{view}_g$  from  $g$ 
     $\text{state} = \text{DataProcessing}(\text{view}_g)$ 
    increment timestamp of  $\text{view}_n$ 
end

Passive thread
Do forever
begin
    Receive  $\text{view}_g$  from  $g$ 
    Send  $\text{DataExchange}(\text{view}_g)$  to  $g$ 
     $\text{state} = \text{DataProcessing}(\text{view}_g)$ 
end
    
```

Algorithm 1: Simple Clustering Gossip Protocol

Hook	Action taken
<i>PeerSelection()</i>	Select Q , the node with the highest age.
<i>DataExchange()</i>	Select the c nodes of $\text{Neighbors}(n)$.
<i>DataProcessing()</i> (simple clustering protocol)	Select nodes closest to n from RPS's view of n , $\text{Neighbors}(n)$ and $\text{Neighbors}(g)$.
<i>DataProcessing()</i> (multi-interest clustering protocol)	Select the view with the highest score from nodes in RPS's view of n , $\text{Neighbors}(n)$ and $\text{Neighbors}(g)$ using Algorithm 2.

Table 1: Function definitions for a node n gossiping with a node g

```

Input: set of nodes  $\text{candidateSet}$ 
Output: a view of size  $\text{viewSize}$ 
 $\text{bestView} = \{\}$ 
for  $\text{setSize}$  from 1 to  $\text{viewSize}$  do
    foreach  $\text{candidate} \in \text{candidateSet}$  do
         $\text{candidateView} = \text{bestView} \cup \{\text{candidate}\}$ 
         $\text{viewScore} = \text{SetScore}(\text{candidateView})$ 
    end
     $\text{bestCandidate} = \text{candidate}$  that got the highest  $\text{viewScore}$ 
     $\text{bestView} = \text{bestView} \cup \{\text{bestCandidate}\}$ 
end
Result:  $\text{bestView}$ 
Algorithm 2: Multi-interest clustering view selection algorithm
    
```

of $\text{Neighbors}(n)$ which evaluates a set of nodes instead of each node independently. The metric to assess the relevance of a set of nodes is the one described above. It is obviously too cumbersome to compute the score of any possible combination of nodes as this is NP hard. We rely on a heuristic to address this issue. We use an algorithm that incrementally builds the view. Starting from an empty view, it builds the best view of size one, and from that view the best view of size two, until it reaches the required view size. The pseudo code is presented in Algorithm 2.

3.3 Anonymity: gossip on behalf

As mentioned before, anonymity is of utmost importance in applications where personal interests may be exposed. The decentralized nature of GOSSPLE inherently preserves anonymity contrary to a central entity which controls and stores every personalized data. Yet, some nodes may still want to benefit from the personalized query expansion without letting others associate their id and their profile. Interestingly enough, the personalized query expansion relies on the personalized network to gather the relevant information for a given node. Yet nodes do not need to explicitly know from which node this information comes from.

GOSSPLE leverages this property and solves anonymity through a *gossiping on behalf* approach. The intuition of this approach is to have a node gossip on behalf of another one without being able to associate such a node with its profile. Effectively, the association between a node and a profile is explicit only when nodes exchange information about their own profiles. To this end, a node n chooses a *proxy* p at random to run the gossip protocol on its behalf. Node n sends its original profile and its profile updates to p while p sends to n users' profiles to fill its personalized network. All nodes in the system, except p , ignore the association between the profile and n . Note that this solution might be sufficient most of the time as nodes might not feel their anonymity is threatened by an individual random node. Yet, GOSSPLE includes additional mechanisms so that even p cannot associate the profile with n . All communications between n and p are encrypted and go through a relay node r , also chosen at random. Thus, p never communicates directly with n and does not know its identity. Node r knows the identity of n but does not know its profile as communication is encrypted. k is a second relay, temporarily used to transmit part of the key for bootstrapping the private communication

The steps of the *gossiping on behalf* protocol are summarized below.

1. n chooses p, r, k at random. Typically the random sample provided by the RPS protocol can be used to select such nodes.
2. n generates a symmetric encryption *key*
3. n splits *key* in two parts and sends it to p , via r and k , each responsible for one part.
4. n and p communicate through r using the encryption and exchange all the information they need

Anonymity guarantees Step 3, where the key is split in two is necessary to prevent any peer, especially r from reading the private information transmitted. Since n chooses p, r, k itself and at random, there is a very low probability that the three would collude against n to break its privacy by exchanging the parts of the encryption key. Nodes r and k know the identity of n and know that p is its proxy, so they may try to gossip with r to get the unencrypted profile, which is why p should avoid gossiping with them. The only option left for r and k is to gossip with other nodes in the network at random and find one that has p in its personalized network in order to get that node's profile and get an idea of what n 's profile could be. Since we consider a large network, such a situation arises

with very low probability and will be even less likely if p, r and k are renewed regularly.

Churn resilience Since GOSSPLE assumes a peer-to-peer network, some nodes might leave the network unexpectedly. If r disconnects, then n chooses a new relay node to establish a new connection with p . If p disconnects, then n should run the protocol back from the beginning. However, to avoid losing all information about the clustering, p periodically sends snapshots of the personalized network to n so that n can resume the gossiping protocol on a new proxy without losing any information.

Anonymity overhead Choosing p, r, k has virtually no cost since these are directly extracted from the sample provided by the RPS protocol. The encryption requires a little extra computation, but the main cost of anonymity is the additional communications it generates. All profiles are forwarded to n from p through r . So, each time a new node enters the personalized network, the cost of fetching its profile is multiplied by three. Apart from that, the anonymity protocol is an addition on the top of the gossip protocol and has no impact on its performance.

4 Personalized query expansion

Queries are expanded using the information gathered from the personalized network: two major steps are involved.

The first step consists, for each node, in collecting and maintaining information about relation between tags provided by the node's profiles of its personalized network. This information is stored in a matrix, the *TagMap*, which contains the raw score, reflecting the proximity between every pairs of tags. Typically, changes in the profiles of the nodes from the personalized networks are conveyed in the *TagMap*.

The second step consists in refining the *TagMap* to explore the relations between tags. More specifically, considering a tag (in a query), the "importance" of other tags is computed by considering all tags and not only the directly related tags. This importance can be seen as the *centrality* of the tag (with respect to a given tag) for that particular node. To compute this score, we use an algorithm inspired by Google Page Rank. The resulting scores are stored in a second matrix called *TagRank*.

4.1 The TagMap: a personalized view of the relations between tags

The *TagMap* is the one of the central abstractions of GOSSPLE and captures the proximity between tags for a given node. A node n collects the profile of all nodes in $Neighbors(n)$, its personalized network. IS_n is defined as the set of triplets (x, i, t) such that $x \in \{Neighbors(n) \cup n\}$, namely the profiles of $Neighbors(n)$ and n . n computes the score of each pair of tags in $Tags(\{IS_n\})$, which is the set of tags involved in IS_n .

Item cosine similarity The information needed to fill the TagMap is, for each tag, the number of occurrences of the use of that tag per item, namely: For all $t \in \text{Tag}(\{IS_n\})$, a vector V_t of dimension $|I|$ is maintained such that if $V_t[\text{item}_i] = v$, where v is the number of times the item item_i has been tagged with t in IS_n . The TagMap is then filled as follow:

$$\text{TagMap}[t_i, t_j] = \cos(\vec{V}_{t_i}, \vec{V}_{t_j})$$

The main difference between this use of cosine similarity and the one applied to rate users is that values are no longer binary. Table 4.1 depicts an example of a node's TagMap. In this example, $\text{score}(\text{Musique}, \text{BritPop}) = 0.9$. This captures the distance between those two tags for that particular node.

	Music	BritPop	Classic	Bach	Oasis
Music	1	0.9	0.3	0.3	0
BritPop		1	0	0	0.41
Classic			1	1	0
Bach				1	0
Oasis					1

Table 2: A TagMap Example

TagMap maintenance The TagMap entry for a pair of tags reflects their distance with respect to the set of items that have been tagged by n and its personalized network. Consequently the TagMap should be updated whenever the personal network changes. In our current design, the TagMap is updated when the personalized network is considered to have significantly changed, *i.e.* half of the nodes in the personalized network either have changed or have updated their profile. Note that updates affect only the entries related to tags affected by the changes in the personalized network.

4.2 TagRank: computing tag centrality

The TagMap represents the personalized scores between any pairs of tags. This information can be directly used to expand the user queries. We call this query expansion approach *Direct Read (DR)*. This approach, used in [28] for example, expands a query with the tags having the highest scores with all tags of the initial query.

More precisely:

$$\text{score}_{DR}(t_i) = \sum_{t \in \text{Initial}_{query}} \text{TagMap}[t, t_i]$$

The number of tags q added to the initial query is a parameter of the query expansion algorithm. With DR, a query is expanded with the q tags scoring the highest. This is an issue for the items with a high sparsity: with a very large number of items, the relationships between tags are sometimes hidden and are difficult discovered. More specifically, they might not always be directly visible in the TagMap.

Consider the TagMap presented in Table 4.1 and a query on *Music* with $q = 2$. The TagMap exhibits a high score between *Music* and *BritPop* (based on a 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 *Music* 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* whereas it seems relevant for that user. Moreover, DR instead expands the query with *Bach*, thus increasing the result set size and reducing the precision of the search.

By iterating on the set of added tags, more relevant tags can be added to the query. To this end, we designed an algorithm called TagRank, inspired from the PageRank [7] algorithm underlying in Google. While PageRank is used to access the relative importance of Web pages, we propose TagRank, which, based on the same intuition, computes the relative importance of tags for a given tag and a given node, providing a notion of tag centrality in this context.

The PageRank (PR) score of a page p_i is given as

$$(1 - d) + d \sum_{p_j \in M_{P_i}} \frac{PR(p_j)}{L(p_j)}$$

where M_{P_i} is the set of pages pointing to p_i , $L(p_j)$ is the number of links from p_j and d is a damping factor, representing the probability that nodes will continue clicking from the current page. Intuitively in PageRank, a random surfer walks in a graph of Web pages. The importance of each page is the probability of the surfer to be on that page at any time. At each step of the walk, the surfer either follows a link on the page (with probability d) or moves to a page chosen uniformly at random on the whole graph (with probability $(1 - d)$)⁴.

TagRank adapts this algorithm to compute the relative importance of tags with respect to any given a tag (typically inside a query). The TagMap is represented as a graph in which all the tags are vertices. One of the main differences with TagRank is that the edges are weighed with the score between tags for that given node. Vertices (tags) are connected by weighted edges so that $weight(t_i, t_j) = TagMap(t_i, t_j)$ and $weight(t_i, t_i) = 1$ ⁵. In TagRank, the transition probability (TRP) from one tag to another depends on the edge weight:

$$TRP(t_1, t_2) = \frac{TagMap[t_1, t_2]}{\sum_{t \in T} TagMap[t_1, t]}$$

While the probability of clicking from a Web page is the same for all links, in TagRank this is guided by the transition probability.

The original PageRank algorithm computes a score for each vertex, but that score only depends on the structure of the graph, not on a node query. Like in personalized versions of PageRank, we modify the set of vertices the *surfer* can move to at random and limit it to the tags of the query. Therefore, the score computed is biased by the query, the query tags being the ones that spread importance into the graph. Those scores are stored in TagRank so that $TagRank[t_i, t_j]$ provides the importance (weight) of t_j when t_i is in the query. Since each weight is a probability, they sum up to one for a given tag. Expanding a query with the TagRank consists then in adding to the original query, the q tags scoring the highest, considering all tags of the initial query.

Maintenance of TagRank Calculating exact TagRank scores in a large graph can be a long process. We use an algorithm from [12] in order to provide a

⁴Typically $d = 0.85$ and we use this same value

⁵This is directly inferred from the metric based on cosine of vectors of items.

	$ U $	$ I $	$ T $	$ IS $
Delicious	20,000	1,144,000	315,626	6,471,684
CiteULike	33,834	1,134,167	237,450	4,064,310

Table 3: Physiognomy of CiteULike and Delicious traces

more efficient approach with a reasonable approximation. The partial scores are approximated through random walks and stored in TagRank. at each update of the TagMap is too expensive as $TagRank[t]$ depends on every tag associated directly or not with t . Instead, $TagRank[t]$ is computed

- each time a query is generated with the tag t if scores for that tags are not yet available (the results are stored in TagRank for further use)
- in the background whenever there are some free cycles

5 Evaluation

In this section, we report on the simulations on a 20,000 to 33,834 node system and a deployment of GOSSPLE on a 170 PlanetLab node testbed of a 340 users. Simulations enable to evaluate the quality of the personalized query expansion on large sets of users and compare it against state of the art approaches. Implementation demonstrates the practicality of the approach and evaluates its cost in a real distributed setting.

In short, our evaluation results demonstrate the following points:

- The personalization of the query expansion improves up to 37% completeness (recall) over state of the art (non personalized) approaches
- Computing tag centrality (TagRank) improves significantly the accuracy (precision) of the result over state of the art approaches, up to 71%
- Effective and anonymous personalization can be achieved with a small number of nodes sharing similar interests through a multi-interest gossip-based clustering.

GOSSPLE is evaluated on two real system workloads: a 33,834 user trace available from CiteULike (managing and discovering scholarly references) and a 20,000 user trace crawled from the social bookmarking system delicious. We used the CiteULike dataset of the 2008-10-09, available from the CiteULike Web site and a delicious trace crawled in January 2009. The figures of the traces are summarized in Table 3. We build a profile for each user $u \in U$ for each trace.

In these traces, we considered only items tagged by at least two users namely 91005 items (present 262609 times in profiles) in the CiteULike trace and 210859 items (present 1023109 times in profiles) in the delicious one.

In addition, we generated two synthetic traces to illustrate how GOSSPLE addresses our babysitter example (in the introduction) and the presence of a mad tagger. The evaluation of the properties and costs of the clustering protocol is discussed in Section 3. Section 5.2 dissects the GOSSPLE query expansion features and evaluates the quality of the mechanism.

5.1 Personalized network evaluation

We evaluate GOSSPLE's ability to build a personal network by aggregating users with similar tagging behaviors. First we evaluate GOSSPLE in a large-scale scenario consisting of 20000 simulated nodes. Then we move to a real deployment consisting of 340 GOSSPLE instances distributed over 170 PlanetLab machines. In both cases, we measure the amount of cycles (or time) required by GOSSPLE to converge to a stable network in which each node is equipped with the personal network that can offer it the best performance in terms of query expansion.

Results for our simulation experiments are depicted in Figure 3(a). At each simulation cycle we evaluate the ratio between the score of each personal network and that of the ideal network computed by an offline algorithm with global knowledge of all user profiles. We then average the resulting ratios, providing a value for each simulation cycle. The three lines in the picture show that the clustering protocol is able to converge very quickly to the correct set of personal networks even in the presence of approximate profiles. The time required to achieve convergence obviously increases with the size of the network, but 200 cycles are sufficient to achieve a score ratio of 90% in a network of as many as 5000 users.

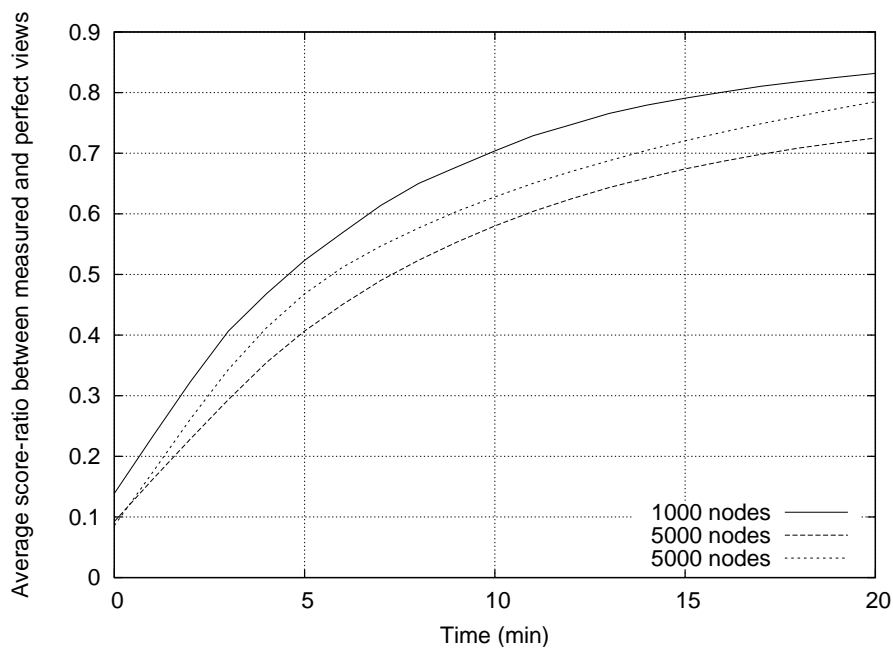
To confirm these good results, we also evaluate the protocol in a real PlanetLab testbed with nodes gossiping every 30seconds. The setting is the same as in the simulations, except that the score ratios cannot clearly be measured at each cycle, but are instead measured at periodic time intervals. The plot depicted in Figure 3(b) shows that the protocol is able to achieve a score ratio of 90% in approximately 10 minutes roughly corresponding to 20 cycles.. The slight increase in the time required to converge with respect to the simulation results discussed above may easily explained by observing the distribution of score ratios over time depicted in Figure 4.

Figure 4 shows that a large number of nodes manage to converge to their ideal personal network in a relatively short time. However, the remaining nodes take longer and a fraction of them even seems to be unable to converge. A closer analysis of the data traces obtained from the experiments revealed that these poor-performing nodes turned out to be unable to participate correctly in the clustering protocol due to particularly high computational loads during the experiments.

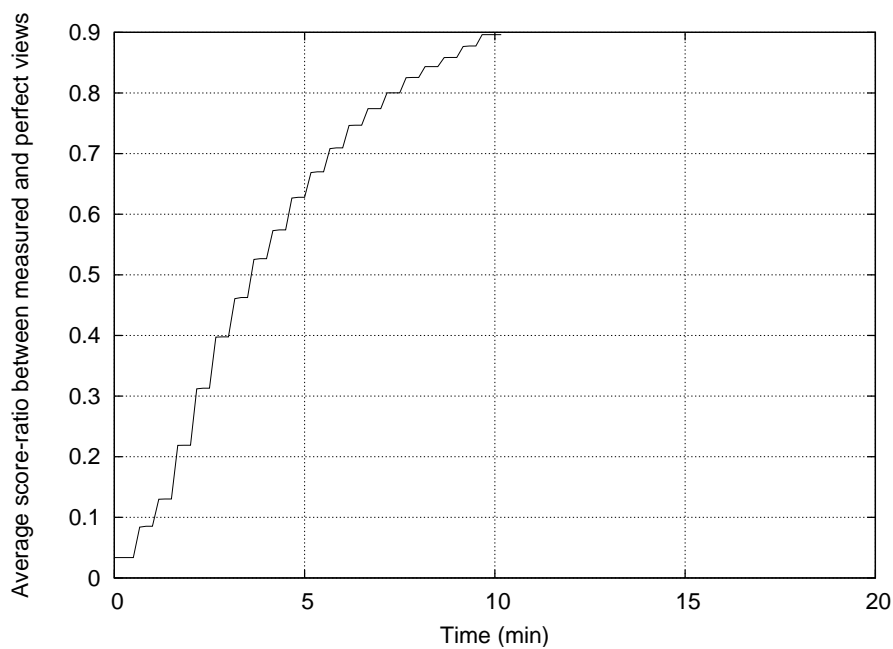
5.2 Personalized query expansion

5.2.1 Evaluation setup

Workload In the evaluation, each node is associated with a user from a real trace. For example, when considering the delicious trace, we consider a 20,000 node system where each node is associated with a delicious user. For the purpose of evaluation, each node generates requests based on its profile in the trace. To this end, the x items in a profile node, generates x requests, the goal of which is to retrieve successively each of those items. Consider a node n . A query is generated for each item in $i \in Item(\{n\})$ such as $|User(\{i\})| > 1$ (an item has to be tagged by at least 2 nodes). The query consists of the tags used to describe the item i by n ($Tag(\{n\}, \{i\})$). The rationale behind the query generation is the following: since this item has been tagged by n with those tags, these tags are assumed to be the ones n used by n to search that item. The query is then



(a) average ratios between the scores of the measured and those of ideal against the number of cycles in networks of 1000, 5000 users.



(b) average ratios between the scores of the measured and those of ideal against time

Figure 3: Convergence speed

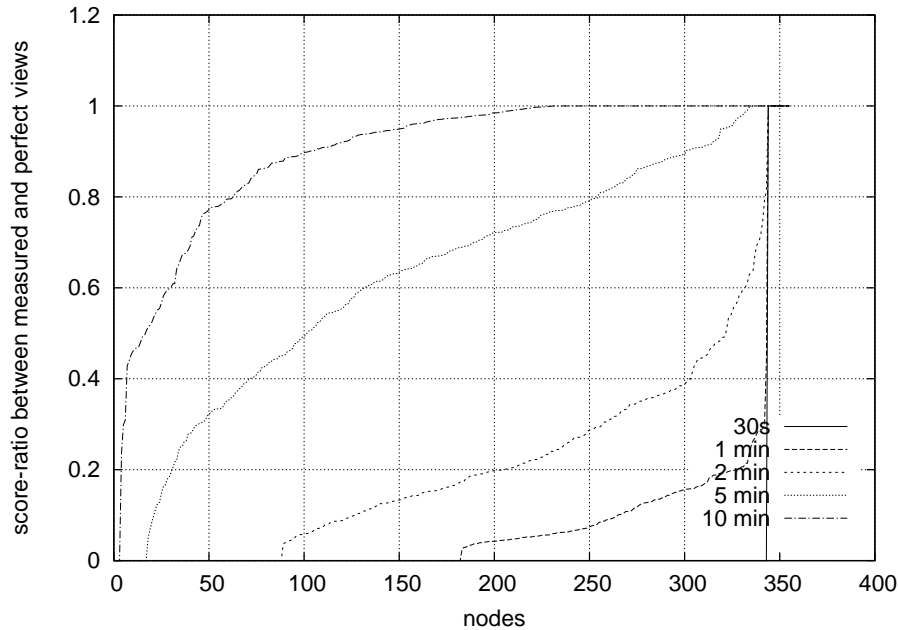


Figure 4: Distribution of ratios between the scores of measured and ideal views at several stages the clustering process.

expanded through GOSSPLE. This item is considered as the expected answer to that expanded query. Obviously, before launching a query from n on that item, the item is removed from n 's profile so that n 's TagMap is not built with this item. More formally n 's TagMap and n 's personalized network are generated at each query considering the information space defined as follow $(IS - (u, i, t), t \in Tag(\{u\}, \{i\}))$.

Search engine We previously mentioned that any search engine could be used to process an expanded query. For the sake of comparison, we choose the search engine and the ranking method used in [28]. An item is in the result set if it has been tagged at least once with one of the tags of the query, and its score reflects how many times it was tagged by a tag in the query. Each tagging is weighted by the weight the query expansion mechanism assigns to the tag.

Metrics The query expansion mechanism is evaluated along the following metrics: *recall* and *precision*. *Recall* expresses the ability of the query expansion mechanism to generate a query that includes in the result set the relevant item. In this set of experiments, we are interested in one item, the one for which the query has been generated. A query succeed when i is in the result set: recall is 1 if the item is in the result set, 0 otherwise. Queries are expanded by adding tags to the query. Consequently, the size result set increases with the length of the query expansion, reducing the visibility of the relevant item. To balance this effect, we also evaluate *precision* to access the quality of the query expansion. In this paper, we consider the (absolute) rank of the items in the result set as

a metric for precision. To this end, the precision is defined as the difference between the rank with and without the query expansion.

The recall evaluation is conducted only on the queries that do not succeed without the query expansion. That concerns 53% of the queries in the CiteULike trace and 25% in the delicious one. These results are interesting by themselves and show that a significant proportion of items have been tagged by at least two users with no tags in common.

Precision however is evaluated on the whole set of queries as the query expansion might have some impact on the visibility of the items.

Experimental settings We follow an incremental approach in our experiments to dissect the effects of each component of the query expansion, namely the impact of the personalization (TagMap and size of the personalized network) and the tag centrality (TagRank). We run the following experiments on the same input trace:

- *Impact of the personalization:* We show that reducing the size of the personalized network improves the quality of the query expansion.
- *Impact of the multi-interest personalization:* We show that the use of the multi-interest clustering protocol (MC) aiming at favouring the minor interests presented in Section 3 improves the quality of the query expansion. This is compared against a clustering protocol (individual rating) called *Simple clustering (SC)* in the sequel.
- *Impact of TagRank:* We evaluate the impact of TagRank over *Direct Read (DR)*.

Note that we do not report on precision when evaluating the personalization and the multi-interest clustering protocol as it was hardly affected. Conversely TagRank improves precision but hardly recall so we present only precision results when evaluating TagRank. The query expansion approach presented in [28] represents our baseline for comparison. Direct Read corresponds to the Social Ranking algorithm proposed in [28], however in [28], the Direct Read is run on a global TagMap, involving the whole set of users. When considering large size personalized network (from 1,000), we assume that this represents a reasonable approximation of a global TagMap. ⁶

5.2.2 Impact of personalization

Figures 5(a) and 5(b) present the extra recall obtained by using the query expansion on the delicious and CiteULike traces respectively. The goal of these experiments is to isolate the impact of the personalization. To this end, a simple clustering is used, providing each node with a personalized network composed of the closest nodes according to the individual rating (SC). This experiment has been conducted with different size of personalized networks: from 10 to 1,000 in CiteULike and 10 to 5,000 in delicious. The figures present the proportion of items that were found with the query expansion out of the ones that were not found without. The size of the query expansion varies from 0 to 50. For

⁶We believe this a fair comparison with Social Ranking as increasing the size of the personalized network affects negatively the quality of the results.

example the point ($x = 20, y = 0.33$) on the 10 neighbours curve on Figure 5(a) says that 33% of requests that were not satisfied without query expansion, are satisfied with a 20 tag query expansion in a system with a personalized network of 10 nodes.

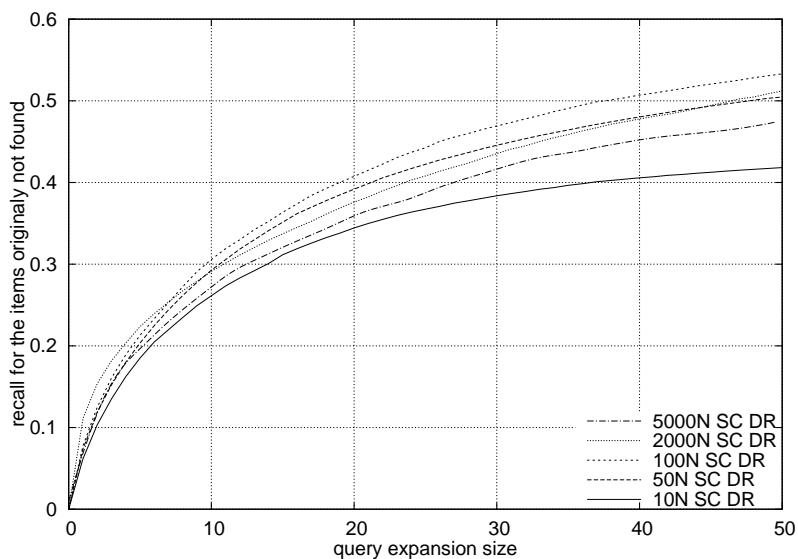
On these figures, we clearly observe the benefit of the personalization. Even though the size of the personalized network has an impact on the recall up to 100, increasing further the size, results in a degraded recall. On the delicious trace (Figure 5(a)), with a 30 tag query expansion, a 10 node personalized network has a recall of 35%, the recall goes up to 46% in a 100 node network and drops down to 38% in a 5,000 network. We can easily extrapolate that incorporating more profiles will degrade the recall further. As the number of node profiles integrated in the TagMap increases, relevant tags are gradually *swallowed* by less relevant tags or popular ones. This is clearly an illustration of the baby-sitting example presented in 1. This trend is more pronounced as the length of the query expansion increases. This is even more striking in the CiteULike experiment where a 10 node personalized network outperforms the 1000 node network for query expansion up to 35 tags. Note that in CiteULike, personalization with a 100 node personalised network exhibit a 50% recall.

5.2.3 Impact of multi-interest personalization

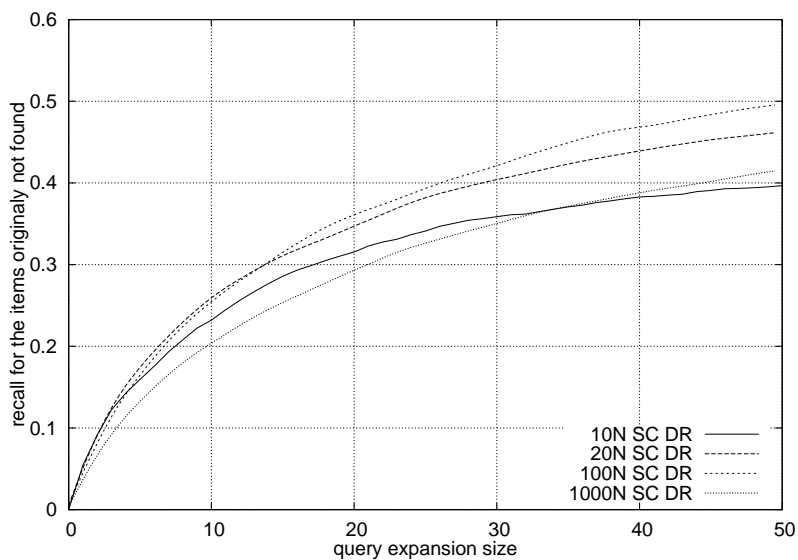
This set of experiments isolates the impact of the multi-interest clustering. Figure 6 presents a similar experiment comparing a simple clustering and the GOSSPLE multi-interest clustering protocol. Based on the previous experiments, we considered personalized network up to a 100 nodes only. We observe that the MC provides consistently a better recall than the SC. For space reason we do not provide the CiteULike results from which we can draw the exact same conclusions.

5.2.4 Impact of TagRank

Finally, we evaluate the impact of TagRank over the DR query expansion. As mentioned previously, the personalization is mostly responsible for increasing recall. However, TagRank significantly improves the precision of the results. This is clearly illustrated on Figure 7(b). The counterpart for the non personalized (we consider a 2000 node network as non personalized) DR query expansion is presented on Figure 7(a). We do not display the personalized (100 node personalized network) DR configuration for the two histograms are extremely similar. This is due to the fact that the precision is mostly impacted by TagRank. Those figures also illustrate the benefit of fully fledged GOSSPLE, all features included. The gain of GOSSPLE is clear while comparing the results. All items are considered (included those which were found without a query expansion). On Figure 7(a), as the query expansion size increases, the recall for items which were not found initially, improves. With a 20 tag query expansion, we observe a 37% recall of the items which were not originally found. Yet, this is accompanied by a significant drop in the precision for 71% of the items originally found. GOSSPLE 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%, items which were already found with the initial query. This is a clear illustration of the impact of GOSSPLE on the precision.



(a) Delicious



(b) CiteULike

Figure 5: Extra recall provided by the query expansion (SC, DR query expansion)

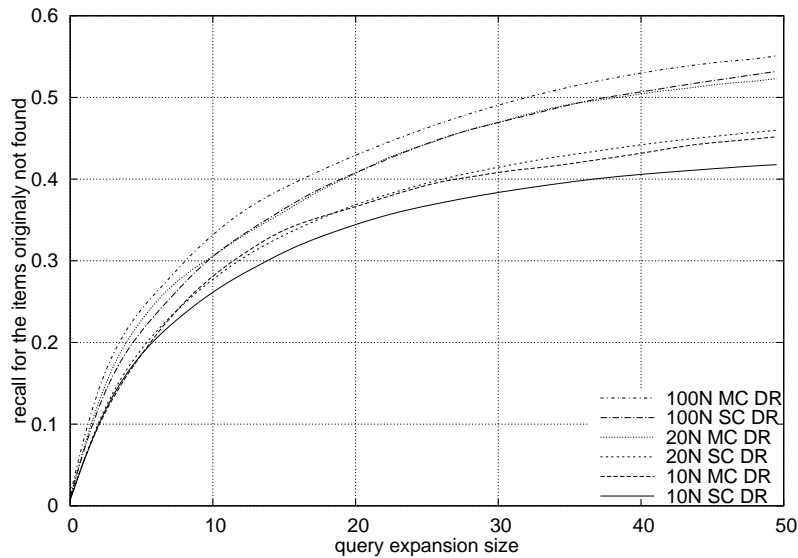


Figure 6: Extra recall provided by the query expansion for the Delicious trace using multi interest clustering (MC and DR query expansion).

Interestingly enough, GOSSPLE improves the precision for approximately 50% of items with a query expansion of 0. This is due to the way TagRank affects the tag's weights reflecting their importance.

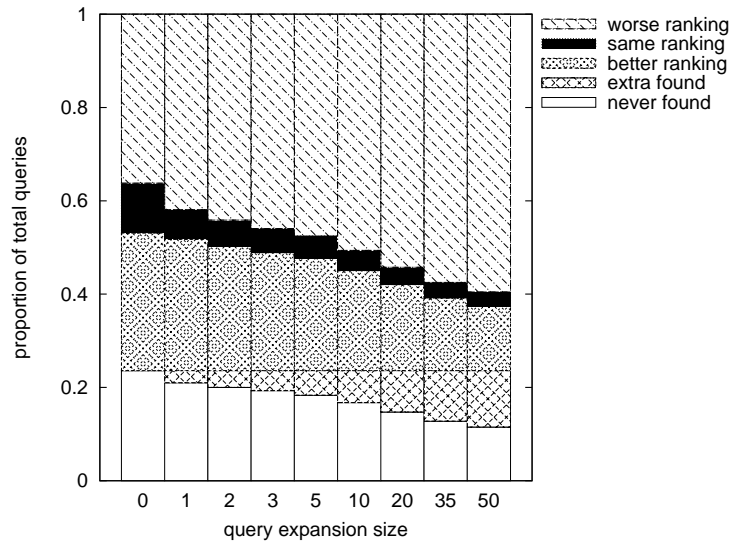
To summarize, while GOSSPLE improves the recall over existing approaches, mostly through personalization, the precision is significantly improved through TagRank. This is of the utmost importance as the better precision enables to expand queries with more tags.

5.2.5 Synthetic traces and discussion

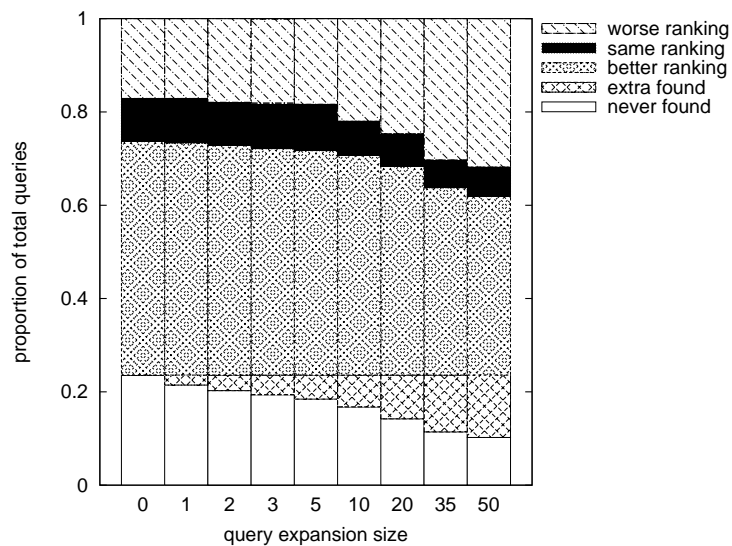
In order to assess the benefit of GOSSPLE on very specific cases, which are extremely hard to derive from the large real traces, we generated two synthetic traces. The first trace was generated to demonstrate that GOSSPLE indeed addresses the baby-sitter example, used as the motivator. The second trace shows that the personalization of GOSSPLE limits the impact of a user trying to force association between tags.

Baby-sitter example As stated in the introduction, tackling the Alice and John baby-sitter example is both challenging and promising. Addressing this issue requires to capture a seldom, but extremely relevant (to Alice), tag association while such tags are each associated by a majority of users with other tags (such as Daycare for baby-sitter as depicted in Figure 1)

To illustrate this example, we generated a 500 user trace, where a large majority of users (490) with fairly similar profiles. Among them, 98 have associated tags *Baby-sitter* and *Daycare*. The 10 remaining ones, have a similar



(a) Non personalized DR query expansion



(b) GOSPLE

Figure 7: Overall performance (recall and precision) of the query expansion on Delicious

profile and only one of them (Alice) has tagged an item with both *Baby-sitter* and *TeachingAssistant*. We run GOSSPLE on that trace with a 10 node personalized network (TagMap & Tagrank) against DR and a 499 node network so that the TagMap takes into account all profiles). Each user generated a query on the tag *baby-sitter*. We run the query for each user in the trace. As expected the 10 users in GOSSPLE expanded the query with *teachingAssistant* while the 490 other users expanded the query with *Daycare*. In the other configurations, all users expanded the query with *Daycare*. Obviously, the trace was made to illustrate our purpose; a slightly larger personalized network would have provided the right results. From this example, we can see that more than 20 neighbours would have been enough to hide the right association. This provides interesting insight for future work where the gaps in distance between a user and the nodes in its personalized network could be taken into account while filling up the TagMap

GOSSPLE Bombing Google bombing consists for a user or a set of users, to force Google to associate a keyword with an item to influence search⁷. Applied to GOSSPLE, bombing would consist in forcing the association between two tags to influence query expansions.

To evaluate GOSSPLE bombing, we added to the delicious trace an extra user forcing a tag association. We chose a tag moderately used in the delicious trace: *ResearchEngine* (used 1,800 times). This tag is used 1,000 with the tag *Search*. Our extra user tagged 10,000 items with *ResearchEngine* and *Gossple* in an attempt to force this association. Each user generated a query on the tag *ResearchEngine* in the two same configurations as above. Results show that such a user has no impact in GOSSPLE as long as the size of the personalized networks remain low (around 20). Above this limit, some users having few neighbors in the network due to data sparsity incorporate the extra user in their personal network and get polluted. In a system with global tags association, all users get directly polluted by the association, while in GOSSPLE only a few users suffer from that attack.

6 Related Work

Personalization User-centric approaches have been considered in several contexts, including wireless routing [23, 5], information dissemination [10], social network management [11], unwanted communication filtering [20] and research [22, 27, 25, 12, 4, 21, 18, 26, 24, 3, 13, 29] and query-expansion [28, 8, 17]. In the context of query expansion, two main approaches have been considered so far.

The first approach consists in collecting information about the past behavior of a user in order to improve her search, e.g., [8, 17]. In [8], the query is expanded by tags already used by the user in previous queries. In [17], 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 a user's feedback. In both cases, no information is inferred from other users. The second approach ([2, 18, 4]) consists in enhancing the search using an explicit (predefined) social

⁷Favorite settings for Google bombing are political elections

network, typically derived from systems a la Facebook, LinkedIn, Last.fm or MySpace. The assumption here is that the explicit, declared, relationships between users indeed reflect their shared interests [19] and can be leveraged to explore the Web. For instance, Web sites like Last.fm and MySpace propose new contents to user using her explicit social network. In most cases, and as we pointed out in the introduction, the information gathered from such networks revealed however very limited in enhancing the search [4].

To our knowledge, we are the first to build an implicit social network of users and leverage it to expand queries. Many centralized systems leverage distance between users in ranking items obtained from a query expansion [22, 27, 25, 12, 13, 29]. We extensively compare our approach with the state of the art representative of these approaches, namely Social Ranking [28].

Folksonomy metrics Several metrics have been proposed to measure the distance between tags or items in a folksonomy. These include the *co-occurrence count*, the *cosine similarity*, the *edit-distance* and *relative centrality*.

Edit-distance [25] captures the relative similarity between tags. The edit-distance between two tag names is equal to the minimum number of insertions, deletions and substitution 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, which is somehow complementary.

Co-occurrence count [22] determines the number of times two tags are used on the same items, normalized by the total number of time 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 [27, 29] or tags [9, 26]: 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 solution, we use an algorithm inspired by the celebrated page-rank algorithm [7] 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 [13, 14, 27, 12].

7 Conclusion

Collaborative social tagging schemes provide a huge potential for discovering new information through implicit connections. We presented GOSSPLE, the first distributed system to discover such connections and leverage them to expand user queries. The GOSSPLE query expansion mechanism improves the completeness of the search queries over state of the art alternatives while improving search accuracy.

GOSSPLE is decentralized and each does only need to maintain little information about its own profile as well as those of relevant acquaintances. Nodes

can join and leave the system dynamically and the profile of a node that left the system is eventually automatically removed. No central authority is involved and there is no single point of failure or information control. This we believe is the only reasonable way to personalise Internet search, in a world where users are free to express their opinion and interests. GOSSPLE promotes anonymity by decoupling the identity of the nodes from their profiles.

We foresee many extensions to GOSSPLE, including alternative applications such as recommendation and top-k processing, as well as protection mechanisms against freeriders and malicious users. We also believe that a fully decentralized and personalised search engine a la GOSSPLE, can reveal very effective.

8 Acknowledgement

We warmly thank Vivien Quéma for his help at early stages of the work.

References

- [1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *Proc. of the 12th international conference on World Wide Web (WWW '03)*, pages 280–290, New York, NY, USA, 2003. ACM.
- [2] S. Amer-Yahia, M. Benedikt, L. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. In *Proc. Very Large Data Bases (VLDB'08)*, volume 1, pages 710–721. VLDB Endowment, 2008.
- [3] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *Proceedings of the 16th international conference on World Wide Web (WWW'07)*, pages 501–510, New York, NY, USA, 2007. ACM.
- [4] M. Bender, T. Crecelius, M. Kacimi, S. Miche, J. Xavier Parreira, and G. Weikum. Peer-to-peer information search: Semantic, social, or spiritual? *Bulletin of Computer Society Technical Committee on Data Engineering*, 2007.
- [5] C. Boldrini, M. Conti, and A. Passarella. Exploiting users' social relations to forward data in opportunistic networks: the hibop solution. *Journal of Pervasive and Mobile Computing (PMC'08)*, 2008.
- [6] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: byzantine resilient random membership sampling. In *Proc. of the twenty-seventh ACM symposium on Principles of distributed computing (PODC '08)*, pages 145–154, New York, NY, USA, 2008. ACM.
- [7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Journal of Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.
- [8] M. Carman, M. Baillie, and F. Crestani. Tag data and personalized information retrieval. In *Proc. of the Workshop on Search in Social Media (SSM '08)*, pages 27–34, New York, NY, USA, 2008. ACM.

- [9] C. Cattuto, D. Benz, A. Hotho, and G. Stumme. Semantic analysis of tag similarity measures in collaborative tagging systems. In *Proc. of the 3rd Workshop on Ontology Learning and Population (OLP3)*, July 2008.
- [10] M. Cha, A. Mislove, B. Adams, and K. P. Gummadi. Characterizing social cascades in flickr. In *Proc. of the first workshop on Online social networks (WOSP '08)*, pages 13–18, New York, NY, USA, 2008. ACM.
- [11] L. A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *Proc. of 6th International Conference on Wireless On-demand Network Systems and Services (WONS 2009)*. IEEE/IFIP, Feb 2009.
- [12] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Journal of Internet Mathematics*, 2(3):333–358, 2005.
- [13] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *Proc. of the 3rd European Semantic Web Conference*, pages 411–426, Budva, Montenegro, 2006. Springer LNCS.
- [14] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, pages 506–514. Springer Verlag, 2007.
- [15] M. Jelasity and O. Babaoglu. T-Man: Gossip-Based Overlay Topology Management. In *Proc. of the 3rd International Workshop on Engineering Self-Organising Applications (ESOA '05)*, volume 3910, pages 1–15. Springer, 2005.
- [16] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Proc. of the 5th international conference on Middleware*, pages 79–98, New York, NY, USA, 2004. ACM/IFIP/USENIX.
- [17] H. Jie and Y. Zhang. Personalized faceted query expansion. In *Proc of the First SIGIR '2006 Workshop on Faceted Search*. ACM, 2006.
- [18] A. Mislove, K. P. Gummadi, and P. Druschel. Exploiting social networks for internet search. In *Proc. of the 5th Workshop on Hot Topics in Networks (HotNets '06)*. ACM, 2006.
- [19] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proc. of the 7th SIGCOMM conference on Internet measurement (IMC '07)*, pages 29–42, New York, NY, USA, 2007. ACM.
- [20] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi. Ostra: leveraging trust to thwart unwanted communication. In *Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, pages 15–30, Berkeley, CA, USA, 2008. USENIX Association.

- [21] J. Morrison. Tagging and searching: Search retrieval effectiveness of folksonomies on the world wide web. *Journal of Information Processing and Management*, In Press, Corrected Proof, 2008.
- [22] S. Niwa, T. Doi, and S. Honiden. Web page recommender system based on folksonomy mining for itng'06 submissions. In *Proc. of Third International Conference on Information Technology: New Generations (ITNG 2006)*, pages 388–393. IEEE, 2006.
- [23] N. Sastry, K. Sollins, and J. Crowcroft. Delivery properties of human social networks. In *Proc. of The 28th Conference on Computer Communications (INFOCOM)*. IEEE, April 2009.
- [24] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of the 28th annual International Conference on Research and development in information retrieval (SIGIR '05)*, pages 449–456, New York, NY, USA, 2005. ACM.
- [25] B. Wong, A. Slivkins, and S. E. G. Approximate matching for peer-to-peer overlays with cubit. Technical report, Cornell University, Computing and Information Science Technical Report, 2008.
- [26] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In *Proc. of the 31st annual international SIGIR conference on Research and development in information retrieval (SIGIR'08)*, pages 155–162, New York, NY, USA, 2008. ACM.
- [27] H. Yildirim and M. S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proc. of the 2008 conference on Recommender systems (RecSys '08)*, pages 131–138, New York, NY, USA, 2008. ACM.
- [28] V. Zanardi and L. Capra. Social ranking: uncovering relevant content using tag-based recommender systems. In *Proc. of the conference on Recommender systems (RecSys '08)*, pages 51–58, New York, NY, USA, 2008. ACM.
- [29] S. Zhao, N. Du, A. Nauerz, X. Zhang, Q. Yuan, and R. Fu. Improved recommendation based on collaborative tagging behaviors. In *Proc. of the 13th international conference on Intelligent user interfaces (IUI '08)*, pages 413–416, New York, NY, USA, 2008. ACM.

Contents

1	Introduction	3
2	GOSSPLE in a nutshell	5
3	Personalized network	8
3.1	Rating users: items cosine similarity	8
3.2	Multi-interest clustering protocol	9
3.3	Anonymity: gossip on behalf	12
4	Personalized query expansion	13
4.1	The TagMap: a personalized view of the relations between tags .	13
4.2	TagRank: computing tag centrality	14
5	Evaluation	16
5.1	Personalized network evaluation	17
5.2	Personalized query expansion	17
5.2.1	Evaluation setup	17
5.2.2	Impact of personalization	20
5.2.3	Impact of multi-interest personalization	21
5.2.4	Impact of TagRank	21
5.2.5	Synthetic traces and discussion	23
6	Related Work	25
7	Conclusion	26
8	Acknowledgement	27



Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399