Exploiting Diversification in Gossip-Based Recommendation

Maximilien Servajean, Esther Pacitti, Miguel Liroz-Gistau, Sihem Amer-Yahia, Amr El Abbadi

Exploiting Diversification in Gossip-Based Recommendation*

Maximilien Servajean[1], Esther Pacitti[1], Miguel Liroz-Gistau[2], Sihem Amer-Yahia[3], and Amr El Abbadi[4]

[1] INRIA & LIRMM, University of Montpellier, France
{servajean,pacitti}@lirmm.fr
[2] INRIA & LIRMM, Montpellier, France
miguel.liroz_gistau@inria.fr
[3] CNRS, LIG
sihem.amer-yahia@imag.fr
[4] Dpt of Computer Science, University of California at Santa Barbara
amr@cs.ucsb.edu

Abstract. In the context of Web 2.0, the users become massive producers of diverse data that can be stored in a large variety of systems. The fact that the users' data spaces are distributed in many different systems makes data sharing difficult. In this context of large scale distribution of users and data, a general solution to data sharing is offered by distributed search and recommendation. In particular, gossip-based approaches provide scalability, dynamicity, autonomy and decentralized control. Generally, in gossip-based search and recommendation, each user constructs a cluster of "relevant" users that will be employed in the processing of queries. However, considering only relevance introduces a significant amount of redundancy among users. As a result, when a query is submitted, as the user profiles in each user's cluster are quite similar, the probability of retrieving the same set of relevant items increases, and recall results are limited. In this paper, we propose a gossip-based search and recommendation approach that is based on a new clustering score, called *usefulness*, that combines relevance and diversity, and we present the corresponding new gossip-based clustering algorithm. We validate our proposal with an experimental evaluation using three datasets based on *MovieLens*, *Flickr* and *LastFM*. Compared with state of the art solutions, we obtain major gains with a three order of magnitude recall improvement when using the notion of *usefulness* regardless of the relevance score between two users used.

1 Introduction

In the context of Web 2.0, users become massive producers of diverse data (*e.g.* photos, videos, scientific data) that can be stored in a large variety of systems (*e.g.* DropBox, Facebook, Flickr, Google+, local computer or smartphone). Users

* Work conducted within the Institut de Biologie Computationnelle and partially funded by the labex NUMEV and the CNRS project Mastodons.

are often willing to share their data with other users in a community of interest. However, the fact that their data spaces are distributed in many different systems makes data sharing especially difficult. For instance, an artist photographer who wants to share her pictures within an online community of photographers may have to log in several different Web applications such as deviantArt, Facebook or Flickr, each with a different interface and account. Similarly, a scientist who needs to search for scientific datasets within an online community of scientists will be faced with the problem that the relevant data is typically distributed in many different labs' servers or scientists' local computers. Furthermore, since this data is hidden to web crawlers, traditional search engines become useless. In order to mitigate this problem, some Web applications allow grouping several accounts and data from different systems (*e.g.* Facebook enables to regroup DropBox and blogs into a single Facebook account). However, they are limited to a few well-known systems.

In this context of large scale distribution of users and data, a general solution to data sharing is offered by distributed search and recommendation [1, 2]. In this paper, we adopt a peer-to-peer gossip-based approach, because it provides important properties such as scalability, dynamicity, autonomy and decentralized control. Within an online community, each user u is associated to a virtual data space that contains all the data items (stored in different systems) it shares. Given u and a keyword query q, the goal of our search and recommendation approach is to recommend to u items that are relevant with respect to q and that are shared by other users, regardless of the systems that store the items. Then, a recommended item is simply a reference that can be used to retrieve the actual data item. In other words, we combine search and recommendation in the sense that a user u searches relevant items among those recommended by users similar to u.

Distributed search and recommendation has received considerable attention [1–4]. However, one open problem is the ability to attain high recall results. A query is generally forwarded only to a subset of users who will be employed to process queries and return recommendations. To compute this subset of users, many solutions cluster relevant user profiles implicitly using gossip protocols. Gossip protocols are known to be highly resilient, scalable and converge quickly [5], which makes them a good alternative for distributed search and recommendation. A *User Network* (*U-Net* in the following) refers to the cluster of relevant users, a user u is aware of by gossiping, using a score (*e.g.* similarity between u and the users in *U-Net*). At each gossip round, the most relevant users are kept in *U-Net*. Since *U-Net* is used to guide recommendations given a keyword query, the relevance score used in the clustering process plays a very important role to increase the number of relevant items retrieved with respect to the whole set of items (*i.e.* recall), known as the global corpus.

Relevance scores (*e.g.* Jaccard, overlap) define how well a user profile v meets the needs of another user u. Most of the existing solutions exploit different kinds of relevance scores to increase recall [2–4, 6, 7]. But recall results remain limited.

The reason why recall remains limited is because using relevance as the clustering score introduces a significant amount of redundant user profiles in *U-Net*. As a result, when a query is submitted, since many user profiles in *U-Net* are quite similar (*i.e.* redundant), and these users are chosen to provide recommendations to answer the query, the probability of retrieving the same set of relevant items increases and recall results remain low. In *Information Retrieval*, usefulness is used as a way to overcome redundancy between the items of a result list by combining relevance with diversity [8, 9]. In our context, we claim that usefulness can be used when clustering user profiles in *U-Net*, instead of just relevance. This way, a more diverse set of results will be returned from different users and the recall would be enhanced.

In this paper, we propose a gossip-based search and recommendation approach based on a new clustering score, called usefulness, that combines relevance and diversity. As we show experimentally, this new score is able to increase significantly the quality of the recommendations returned by the system. However, existing peer-to-peer clustering algorithms are no longer suitable since they are optimized for relevance only. Therefore we also propose a new clustering algorithm especially conceived for usefulness.

In summary, we make the following contributions:

1. We show that *usefulness* is a good way to increase recall and that it should be expressed as a known probabilistic diversification score [8, 9].
2. We propose a clustering algorithm that maintains a useful *U-Net* over a gossip overlay using the usefulness score.
3. We validate our approach with an experimental evaluation using three different datasets: *MovieLens*, *Flickr* and *LastFM*. We observe that diversification enables a huge increase of recall regardless of the relevance score used. Compared with state of the art solutions, we obtain an excellent gain with recall results up to three times better when using the notion of usefulness.

This paper is organized as follows. Section 2 provides some basic concepts and gives the problem definition. In Section 3, we describe our new clustering score and present in details the new clustering algorithm that maintains *U-Net*. In Section 4, we provide an experimental evaluation. In Section 5, we compare our contributions with related work. Finally, Section 6 concludes and provides directions for future work.

2 Basic Concepts and Problem Definition

In this section, we introduce the background necessary to understand the problem we address. In our distributed search and recommendation approach, whenever a user u submits a query q, the system sends q to a subset of users that we call *U-Net*, and who will return their relevant results to u and will also recursively forward the query to the users in their *U-Net* until the *TTL* is reached. To build *U-Net*, we use a two steps approach. First, based on *random gossiping* each user u is aware of other peers available on the network. Second, by means of a *clustering* algorithm, u chooses among these users the best ones to answer u's queries and keep them in *U-Net*.

More precisely, our peer-to-peer model is expressed based on a graph $G = (U, I, E)$, where $U = \{u_1, ..., u_n\}$ is the set of users distributed over the network, $I = \{i_1, ..., i_m\}$ the set of shared data items (in the following, an item refers to a data item), and $E = \{e_1, ..., e_k\}$ the set of directed edges among users and between users and items. This model is very generic. In our case, users are independent nodes in the network. A node can be a physical computer or a virtual node in a server.

Definition 1 (*U-Net*). *Given a user u, its User Network, or U-Net, refers to the cluster of relevant users u is aware of. There is an edge $e(u, v)$ in the graph between u and a user v, if v is in u's U-Net.*

With random gossiping [5], each user keeps locally a random view of its dynamic acquaintances (or view entries). Each view entry corresponds to a user profile. Periodically, each user chooses randomly a contact (view entry) to gossip with. The two involved users then exchange a subset of each others view (*i.e.* user profiles), and update their view state. Then, after each gossip exchange, the random view is used to update the *U-Net* if more relevant profiles are found in the updated view. We use *Jaccard* as the relevance score to select the best users:

$$\mathbf{Jaccard}(\mathbf{u}, \mathbf{v}) = |I_u \cap I_v| \ / \ |I_u \cup I_v| \tag{1}$$

Where I_u and I_v are the items shared by user u and v respectively.

Here, we use the vector space model to represent items and user profiles [10]. Specifically, each item *it* is modelled as a sparse vector containing only the weights of the keywords $k_1, ..., k_z$ in it. The weight of each keyword is computed using $tf \times idf$. Distributed $tf \times idf$ can be easily implemented using gossip protocols. Indeed, the first part of the score, denoted *tf*, can be computed locally, and the second part, denoted *idf*, only needs average information (*e.g.* average number of items per user) to be computed. These averages can be easily computed using gossip protocols [11]. Due to lack of space, we do not develop this protocol in this paper.

Each *user profile* is defined based on the items the user shares, I_u (*i.e.* content based recommendation). We choose a relevance score (*i.e. Jaccard*) that works well with content-based recommendation, but other relevance measures and profiles definition methods could be used as well.

As mentioned before, whenever a user u submits a keywords query $q = k_1, ..., k_w$, the query is redirected to all users in the participating users' *U-Net* recursively, until a predefine upper threshold, TTL (*i.e. Time-To-Live*). Whenever a user v receives a query, it computes its *top-k* most relevant items with respect to the query using a specific relevance score (*e.g.* Jaccard). Then, v returns them to u. A recommended item is defined by its identifier, its $tf \times idf$ vector, v's identifier and v's profile. Once u receives the set of recommended items from $v_1, ..., v_n$ with respect to its query q, it ranks them based on their relevance with respect to the query:

$$Rec_q = rank(rec_q^1(it_1, ...) \cup ... \cup rec_q^n(it_p, ...)) \tag{2}$$

Where $rec_q^1(it_1, ...)$ is a recommendation (*i.e.* a set of recommended items) coming from a user v_1.

To evaluate the quality of search and recommendation, we use the *recall* measure [12]. Recall captures the fraction of items that have been successfully recommended: $recall = |Iret_q| \ / \ |Irel_q|$, where $Iret_q \in I$ refers to the relevant items recommended with respect to a query q, and $Irel_q \in I$ refers to all the relevant items with respect to query q.

Problem Definition: Given a user $u \in U$, a query q, I in G, and a gossip based overlay, the goal is to maximize the number of relevant items with respect to q returned to u while minimizing TTL.

3 Diversified Clustering and Algorithm

In this section, we show that usefulness is an excellent way to increase recall results of gossip-based recommendation, and can be used as a clustering score. In section 3.1, we formally show that to increase recall, usefulness should take into account relevance and diversity. Next, Section 3.2 presents the *Useful U-Net* clustering algorithm deployed over a gossip-based overlay.

3.1 Usefulness Score

The usefulness score should be designed such that it maximizes the probability that a user u can retrieve relevant items given a random query q, known as the coverage probability. In other words, u's neighbors $v_1, ..., v_n \in G$ should be chosen such that the number of relevant items (with respect to the queries u will submit) that can be accessed through them is maximized.

Let Q be the set of all possible queries (all the combinations of terms), and $P(Q_v)$ the probability that a user v can return at least one relevant item given a random query $q \in Q$. In the following, we first define the coverage with respect to *U-Net*$_u = \{v_1, ..., v_n\}$. Then, based on coverage, we express the usefulness of a user v with respect to the other users in u's *U-Net*.

Definition 2 (Coverage). *Given Q and U-Net$_u = \{v_1, ..., v_n\}$, the users in u's U-Net. The coverage is the probability that at least one of the user in u's U-Net can return at least one item given a random query $q \in Q$. Coverage is denoted $P(Q_{v_1} \cup Q_{v_2} \cup ... \cup Q_{v_n})$.*

The user profiles $v_1, ..., v_n$ must be selected such that the coverage probability is maximized. Formula 3 develops the coverage probability with respect to every user in u's *U-Net*.

$$P(Q_{v_1} \cup ... \cup Q_{v_n}) = \sum_{j \in 1,...,n} (P(Q_{v_j}) - P(Q_{v_j} \cap (Q_{v_{j+1}} \cup ... \cup Q_{v_n}))) \quad (3)$$

$P(Q_{v_j}) - P(Q_{v_j} \cap (Q_{v_{j+1}} \cup ... \cup Q_{v_n}))$ represents the coverage added by user v_j with respect to the users $v_{j+1}, ..., v_n$. As a consequence, when $j = n$, only $P(Q_{v_j})$ is considered as there is no more user profiles to compare with.

In the following, we define the usefulness of a user profile v_i with respect to the coverage probability.

Definition 3 (Usefulness). *Given u's U-Net, the usefulness of a user profile v_j is the probability that it can return relevant items for a random query q, that*

could not be returned by other users in u's U-Net. In other words, it is defined as follows:

$$usefulness(v_j|v_{j+1},...,v_n) = P(Q_{v_j}) - P(Q_{v_j} \cap (Q_{v_{j+1}} \cup ... \cup Q_{v_n})) \quad (4)$$

Formula 4 shows that the usefulness score should consider relevance $P(Q_{v_j})$ and take into account $P(Q_{v_j} \cap (Q_{v_{j+1}} \cup ... \cup Q_{v_n}))$ which corresponds to the redundancy of user profile v_j with respect to the other user profiles $v_{j+1}, ..., v_n$.

In the following, we show that $usefulness(v_j|v_{j+1},...,v_n)$ can be expressed into a known probabilistic diversification model [8, 9]. In Formula 5 we first integrate usefulness (the right hand side of Formula 4) into a conditional probability.

$$\begin{aligned} P(Q_{v_j}) - P(Q_{v_j} \cap (Q_{v_{j+1}} \cup ... \cup Q_{v_n})) &= P(Q_{v_j}) \times (1 - P(Q_{v_{j+1}} \cup ... \cup Q_{v_n}|Q_{v_j})) \\ &= P(Q_{v_j}) \times P(\bar{Q}_{v_{j+1}} \cap ... \cap \bar{Q}_{v_n}|Q_{v_j}) \end{aligned} \quad (5)$$

Similar to [8, 9, 13], we assume that the redundancy of a user profile v_1 with another user profile v_2 is independent of its redundancy with other users and we derive Formula 6.

$$P(Q_{v_j}) \times P(\bar{Q}_{v_{j+1}} \cap ... \cap \bar{Q}_{v_n}|Q_{v_j}) = P(Q_{v_j}) \times \prod_{i \in j+1,...,n} (1 - P(Q_{v_i}|Q_{v_j})) \quad (6)$$

Finally, we observe that the usefulness of a user profile is clearly similar to the probabilistic diversification problem used in [8, 9] and presented in Formula 7.

$$usefulness(v_j|v_{j+1},...,v_n) = rel(v_j) \times \prod_{i \in j+1,...,n} (1 - red(v_j, v_i)) \quad (7)$$

where $rel(v_j) = P(Q_{v_j})$ is the relevance of user profile v_j and $red(v_j, v_i) = P(Q_{v_i}|Q_{v_j})$ is the redundancy of user profile v_j with respect to the other user profile v_i.

3.2 Useful U-Net Clustering

We now present in details our clustering algorithm that maintains a useful *U-Net* over a random gossip overlay using the usefulness score.

Given the set of users in the random view, the goal of the clustering algorithm is to compute the usefulness of each user found in the view, with respect to those that were previously added to the *U-Net*, taking into account relevance and diversity, as defined in Equation 7, and to update the *U-Net* as consequence.

Based on random gossiping [5], each user u maintains a set of random view entries corresponding to the users profile u is aware of. Periodically, users gossip, and exchange a random subset of their views entries. After the random gossip merging phase, the clustering algorithm, which corresponds to the *Useful U-Net* Algorithm depicted in Algorithm 1, is triggered. In fact, taking into account the previous gossip exchange, the algorithm selects the most useful users from the random view considering the useful users previously selected (*i.e.* from the previous gossip rounds) in the *U-Net*. The algorithm uses three main data structures: random view, *U-Net*, and the candidate list. The random view and the *U-Net* are initialized when u joins the network, and continuously updated as a result of random gossip. The candidates list contains the user profiles that will potentially

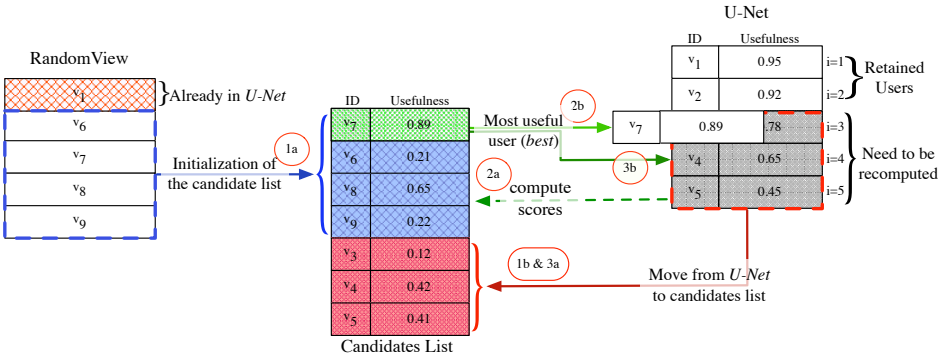


Fig. 1: An example of the execution of Useful-Unet.

be added to the *U-Net* and is initialized each time the clustering algorithm is triggered.

In the following we present in more details the *Useful U-Net* algorithm based on the example of Figure 1. The random view entries correspond to the profiles of users v_1, v_6, v_7, v_8, v_9. The previous useful user profiles are v_1, v_2, v_3, v_4, v_5 and are stored in *U-Net*. Assuming that the algorithm is executed in u's node, the algorithm input is u's profile, its random view denoted $RandomView_u$ and its *U-Net* denoted $U\text{-}Net_u$. The data structure used for *U-Net* is an array of size N of user profiles, associated to their usefulness score and sorted in decreasing order of usefulness. The output of the algorithm is the updated *U-Net*. *Useful U-Net* algorithm has three main parts:

1. The first part (lines 1 to 6) finds the *best* useful user profile from the random view, and the position i where it should be inserted in the *U-Net* (recall that the usefulness score of a user depends on its position in the *U-Net*). As a consequence, the update of the *U-Net* will only concern the user profiles from position i to N. To find the *best* useful user from the random view, the algorithm first initializes the *candidates* list with all users in the random view except those already in the *U-Net* (line 2). In Figure 1, v_1 is already in the *U-Net*, so the candidates list is initialized with the users v_6, v_7, v_8, v_9 $(1a)$. For each position i in *U-Net*, all the usefulness scores of the candidates are computed using Formula 7 taking into account the set of users in the *U-Net* at positions $1, ..., i-1$, and compared with the usefulness score of the user profile in $U\text{-}Net_u[i]$. If the best user profile in *candidates* is more useful than $U\text{-}Net_u[i]$, then, the algorithm stops iterating (line 6). If there is more than one best user profile, the best user profile is chosen randomly with respect to the set of best user profiles. In Figure 1, v_7 is more useful than v_3 at the third position in u's *U-Net* because v_3's usefulness is 0.78 while v_7's usefulness is 0.89 $(1b)$. If there is no user profile in the candidate list whose profile score is superior to any user profile in the *U-Net*, position N is reached and the algorithm stops. Only the scores of the user profiles up to position i are definitive. Thus, in our example, the scores of v_4, v_5, v_6, v_8, v_9 are not definitive because they are either not in the *U-Net* or after i.

2. The second part (lines 7 to 10) copies and deletes the remaining user profiles (from position i to N) from the *U-Net* to the candidates ($2a$) list because their scores need to be recomputed using Formula 7 and with respect to the best user profile in *candidates* (computed in part 1). Then, the best user profile is inserted in position i. In the on-going example of Figure 1, the user profiles v_3, v_4, v_5 are copied and removed from the *U-Net* to the candidates list and user profile v_7 is added in the *U-Net* at position 3 ($2a$ and $2b$).
3. Finally, in the last part (lines 11 to 15), the algorithm iteratively computes, for each empty position i in the *U-Net* (positions emptied in part 2), the scores of the user profiles in the candidates list using Formula 7 and taking into account the set of users in the *U-Net* at positions $1, ..., i-1$ (lines 12 and 13 and step $3a$ in the figure). Then, the most useful candidate is moved to the *U-Net* at that position (line 15 and step $3b$ in the figure). The algorithm repeats these steps until all the positions in *U-Net* are filled out (line 11).

Recall that gossip protocols converge quickly [3]. As a consequence the *U-Net* will also converge quickly and, in general, tends to stabilize. Therefore, the algorithm will stop at step $1b$ more and more frequently.

Algorithm 1: Useful *U-Net*

Input: u profile, $U\text{-}Net_u$ (array[1..N]), $RandomView_u$
Output: $U\text{-}Net_u$ is updated with respect to the $RandomView$

```
1  candidates : unsorted list of user profiles;
2  candidates ← RandomView_u − U-Net_u; best ← ∅; i ← 0;
3  repeat
4  |  i++;
5  |  for each c_j ∈ candidates do score(c_j) ← usefulness(c_j,u,U-Net_u[1..i − 1])
   |  best ← arg max_{c∈candidates}(score(c));
6  until i=N or score(best) > score(U-Net[i]);
7  if score(best) > score(U-Net[i]) then
8  |  after← U-Net_u[i..N]; U-Net_u[i] ← best; i++;
9  |  candidates ← candidates − best;
10 |  candidates ← after ∪ candidates; U-Net_u ← U-Net_u − after;
11 |  while i < N and candidates≠ ∅ do
12 |  |  for each c_j ∈ candidates do
13 |  |  |  score(c_j) ← usefulness(c_j, u, U-Net_u[1..i − 1]);
14 |  |  best ← arg max_{c∈candidates}(score(c)); U-Net_u[i] ← best;
15 |  |  candidates ← candidates − best; i++;
```

4 Experimental Evaluation

In this section, we provide an experimental evaluation to validate our approach and compare it to other state-of-the-art solutions. We conducted a set of experiments using three datasets which correspond to *MovieLens*, *Flickr* and *LastFM*. In Section 4.1, we introduce the experimental setup of our evaluation. Then, in Section 4.2, we present and discuss the experimental results.

4.1 Experimental Setup

We ran our experiments on the *PeerSim* simulator[5]. We used three different datasets: *MovieLens*, *Flickr* and *LastFM*. *MovieLens* dataset is composed of users that rated movies. *Flickr* dataset is composed of users that submitted or added a picture to their favorites. Each user also associates tags to the pictures he/she submits. Finally, LastFM dataset is composed of users who listen and associate tags to artists. Each dataset has different features, in particular users are more or less redundant if the number of items per user is more or less respectively. The characteristics of the datasets are summarized in the following table.

dataset	**items**	**# items**	**# users**	**avg items/user**
MovieLens	Movies	3,900	6,040	166
Flickr	Pictures	2,029	2,000	3.7
LastFM	Artists	23,346	2,000	98

The queries used in the experiments consist of: In *MovieLens*, for each user, a random subset of movies are shared and the rest are used as the queries to submit. In particular, the words in the title are used as separate keywords. In *Flickr* and *LastFM* queries are computed as the random association of several tags submitted by a given user on a given item. An experiment is composed of two parts. First, all users gossip during 400 rounds until convergence. Then, every 20 gossip rounds all users submit one of their queries. The experiment stops at 500 gossip rounds. We measure the average recall results. The recall enables to compute the fraction of items that has been successfully recommended as presented in Section 2. On the *MovieLens* dataset, the recall value is 1 if the movie has been found and 0 otherwise. On *Flickr* and *LastFM*, the recall is the proportion of pictures in the whole dataset that contains all query's keywords that have been returned to the user. On the *Flickr* and *LastFM* experiments, we have computed the *variance* which enables to compute the variability of the recall and is computed as follows: $V(X) = 1/N \times \sum_{i=1}^{n}(x_i - m)^2$ where m is the average recall.

In our experiments, we use the following relevance scores:

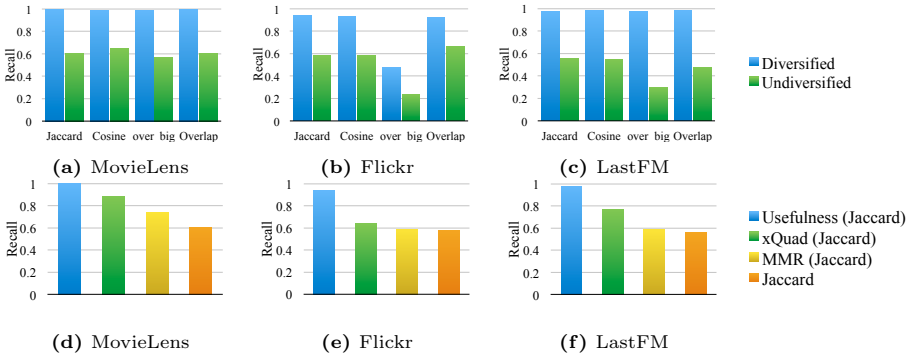
$\mathbf{overlap}(\mathbf{u},\mathbf{v}) = \lvert I_u \cap I_v \rvert$	$\mathbf{over\_big}(\mathbf{u},\mathbf{v}) = \lvert I_u \cap I_v \rvert + \lvert I_v \rvert$
$\mathbf{Jaccard}(\mathbf{u},\mathbf{v}) = \lvert I_u \cap I_v \rvert \ / \ \lvert I_u \cup I_v \rvert$	$\mathbf{cosine}(\mathbf{u},\mathbf{v}) = Ir_u \times Ir_v \ / \ \lvert\lvert Ir_u \rvert\rvert \times \lvert\lvert Ir_v \rvert\rvert$

where I_u and I_v are the items shared by u and v, respectively, and where Ir_u and Ir_v are the set of ratings u and v gave to the items they share. We have fixed the *U-Net*'s size to 16 and TTL to 3. Other values have been tested and showed similar results. The size of the random view (5 in our case) is not important as it only modifies the convergence speed.

4.2 Experiments

Figure 2 presents the results of our experiments. More precisely, Figures 2a, 2b and 2c compare the recall results of the used relevance scores with and without

[5] `www.peersim.sourceforge.net`



(a) MovieLens **(b)** Flickr **(c)** LastFM

(d) MovieLens **(e)** Flickr **(f)** LastFM

Fig. 2: Effect on recall of diversification

including our usefulness score, while Figures 2d, 2e and 2f compare the recall results of several diversification methods.

Not surprisingly, diversifying the *U-Net* enables for all relevance score to significantly increase recall. On the *MovieLens* dataset, the recall results without diversification range between 0.58 and 0.62 while they range between 0.978 and 0.999 with diversification. On the *Flickr* dataset, the gains are slightly smaller. Since all users share their own pictures, their profiles are very different and already diversified. Therefore, diversification has less impact on the recall. Finally, the *LastFM* dataset recall results are up to 3.26 times higher.

In addition to improve the recall, diversified solutions enable to reduce the variance compared to undiversified solutions. For instance, on *Flickr*, the variance decreases from 0.116 to 0.013 when using `Jaccard`. This can be explained by the fact that in the undiversified solution, users in *U-Net* are very similar among them. As a consequence, either all are relevant to the query, and hence they provide a high recall; or none of them is, thus producing a low recall. Diversification enables to increase coverage and therefore, it increases the probability to answer any kind of query.

In addition, we ran these experiments with different sizes of *U-Net* and values of *TTL*. For instance, on the *MovieLens* dataset, with a *U-Net* of size 5 and a *TTL* of 2, the recall is in average 2.37 times higher compared to undiversified solutions. Indeed, without diversification, recall values are in average of 0.26 while they reach $0,61$ using usefulness.

We have also compared three different diversification methods. The first is the *usefulness* score presented in Equation 7. The second method we use is the *Maximal Marginal Relevance*, known as *MMR* [13]. *MMR* chooses users that minimize the maximum similarity between any two users in u's *U-Net*. Finally, the last method is *Explicit Query Aspect Diversification* known as *xQuad* [14]. *xQuad* chooses users such that each user v_i in u's *U-Net* is similar to u in a different way. For instance, suppose that u shares items i_1 and i_2. If v_1 is in u's *U-Net* and is similar to u because it also shares i_1, then, *xQuad* chooses a user v_2 such that v_2 is similar to u because it shares the item i_2. In this experiment, we use *Jaccard* as the similarity measure.

Figures 2d, 2e and 2f show that all diversification methods enable to increase the recall values compared to undiversified methods. Among them, *usefulness* obtains the best gain in terms of recall closely followed by *xQuad*. Finally, *MMR* shows the worst gain in terms of recall. Indeed, *MMR* chooses users that minimize the maximum similarity between any two users in u's *U-Net*. Therefore, it prefers users that are a little bit similar with every user in u's *U-Net*, and that do not necessarily increase recall results.

5 Related Work

Distributed recommendation for web data based on collaborative filtering has been recently proposed with promising results. In this section, we compare our recommendation approach with state of the art solutions.

In [15], Loupasakis and Ntarmos propose a decentralized approach for social networking with three goals in mind: privacy, scalability with profitability and availability. They propose an architecture based on a DHT for keywords query search. Since, DHTs are better suited for exact-match queries, the author propose to decompose each query into several single word exact-match queries. The main drawback is that responses that have medium scores with respect to each keyword but high scores with respect to all the keywords are likely to be missed.

P2PRec [3] has been proposed as a gossip-based search and recommendation solution. The profile of each user u is represented as a set of topics computed based on the items u shares. Then, using gossip protocols, similar users in term of topics, are clustered together and used to guide recommendation as we do. However, since diversity is not taken into account, users within each cluster can be redundant, thus limiting recall results. In [6], Kermarrec et al. focus on recommendation and propose to combine gossip algorithms and random walks. First, the users are clustered based on relevance through gossip protocols. A user has knowledge of the items shared by its neighbors. To compute the recommendation, each user runs locally a random walk using a transition similarity matrix. However, computing this matrix and its inverse seems not scalable due to the computational complexity of the algorithm with respect to the size of the neighborhood and the number of items. Also in the context of recommendation, in [7] Kermarrec et al. claim that, because of the heterogeneity of the users in the network, a single similarity measure to cluster users is not sufficient to achieve good recall results. Instead they propose that each user employ its own similarity measure to build its view (clustering layer) of the network. Nevertheless, the concept of diversity is different from ours as it represents the usage of various relevance scores depending on each user's profile. As a consequence, each user's cluster may still carry redundant user profiles, because there is no explicit diversification. In [1], Bai et al. propose a solution for personalized *P2P top-k search* in the context of collaborative tagging systems, called *P4Q*. In this solution, the users are clustered based on relevance through gossip protocols. The users in each cluster are split into two groups: 1) the c closest users from which u replicates all items metadata (*i.e.* tagging actions) and 2) the n less similar

users from which u knows only the profile (*i.e. bloom filter*). Still, diversity is not taken into account and users within the clusters are likely to be redundant.

6 Conclusion and Future Work

In this paper, we proposed a new gossip-based search and recommendation approach with new measures and techniques. We first showed that usefulness, by combining relevance and diversity, is very effective in increasing recall results and can be used as a clustering score. Then, we designed a new clustering algorithm based on usefulness that combines relevance and diversity. We validated our proposal with an experimental evaluation using the *MovieLens* dataset. Compared with state of the art solutions, we obtain major gains with recall results more than two times better.

In future work we intend to exploit other recommendation scenarios such as multisite recommendation.

References

1. Bai, X., et al.: Collaborative personalized top-k processing. Transactions on Database Systems **36**(26) (December 2011)
2. Carretero, J., et al.: Geology: Modular georecommendation in gossip-based social networks. In: ICDCS. (2012) 637–646
3. Draidi, F., Pacitti, E., Parigot, D., Verger, G.: P2Prec: a social-based P2P recommendation system. In: CIKM. (2011) 2593–2596
4. Voulgaris, S., Steen, M.: Epidemic-style management of semantic overlays for content-based searching. In: Euro-Par. (2005) 1143–1152
5. Jelasity, M., Babaoglu, O.: T-man: Gossip-based overlay topology management. In: ESOA. Volume 3910 of Lecture Notes in Computer Science., Berlin, Heidelberg (2005) 1–15
6. Kermarrec, A., Leroy, V., Moin, A., Thraves, C.: Application of random walks to decentralized recommender systems. In: OPODIS. Volume 6490 of Lecture Notes in Computer Science. (2010) 48–63
7. Kermarrec, A., Taïani, F.: Diverging towards the common good: Heterogeneous self-organisation in decentralised recommenders. In: SNS. (2012) 3–8
8. Angel, A., Koudas, N.: Efficient diversity-aware search. In: SIGMOD. (2011) 781–792
9. Chen, H., Karger, D.: Less is more: Probabilistic models for retrieving fewer relevant documents. In: SIGIR. (2006) 429 – 436
10. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
11. Kowalczyk, W., Jelasity, M., Eiben, A.: Towards data mining in large and fully distributed peer-to-peer overlay networks. In: BNAIC. (2003) 203–210
12. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc. (1999)
13. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: SIGIR. (1998) 335–336
14. Santos, R., Peng, J., Macdonald, C., Ounis, I.: Explicit search result diversification through sub-queries. In: ECIR. (2010) 87–99
15. Loupasakis, A., Ntarmos, N.: eXO: Decentralized autonomous scalable social networking. In: CIDR. (2011) 85–95