

# Distributed Collaborative Filtering for Peer-to-Peer File Sharing Systems

Jun Wang, Johan Pouwelse, Reginald L. Lagendijk, Marcel J.T. Reinders  
Information and Communication Theory Group,  
Faculty of Electrical Engineering, Mathematics and Computer Science,  
Delft University of Technology  
Mekelweg 4, 2628 CD Delft, The Netherlands

{j.wang, j.a.pouwelse, r.l.lagendijk, m.j.t.reinders}@ewi.tudelft.nl

## ABSTRACT

Collaborative filtering requires a centralized rating database. However, within a peer-to-peer network such a centralized database is not readily available. In this paper, we propose a fully distributed collaborative filtering method that is *self-organizing* and operates in a *distributed* way. Similarity ranks between multimedia files (items) are calculated by log-based user profiles and are stored locally at these items in so-called *buddy tables*. This intuitively creates a semantic overlay to organize multimedia files. Based on this semantic overlay and the items that a user has downloaded previously (indicating the profile of the user), recommendations can be performed and the recommended items can be easily located. We have tested our distributed collaborative filtering approach and compared it to centralized collaborative filtering, showing that it has similar performance. It is therefore a promising technique to facilitate filtering for relevant multimedia data in P2P networks.

## Keywords

Recommendation, Personalization, Collaborative Filtering, Peer-to-Peer Networks

## 1. INTRODUCTION

Peer-to-peer (P2P) networks have become a new and popular way for people to exchange multimedia data that is stored on their local storage devices. Examples are P2P file sharing systems like: Gnutella<sup>1</sup>, BitTorrent<sup>2</sup> and P2P Music streaming systems like iTunes<sup>3</sup>. P2P networks increase content availability dramatically since the involvement of third parties that manage the centrally stored content can be avoided. This comes, however, at a price: both the users as well as the data are distributed and dynamically chang-

<sup>1</sup><http://www.gnutella.com>

<sup>2</sup><http://bitconjurer.org/BitTorrent/>

<sup>3</sup><http://www.apple.com/itunes/>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'06 April 23-27, 2006, Dijon, France

Copyright 2006 ACM 1-59593-108-2/06/0004 ...\$5.00.

ing which make it difficult to filter (and search) and localize the available content within the P2P network.

Collaborative filtering propose a similarity measure that expresses the relevance between an item (the content) and the preference of a user. Current collaborative filtering analyzes a rating database of user profiles for similarities between users (user-based) or items (item-based). Within the context of P2P networks there is, however, no centralized rating database so that current collaborative filtering approaches cannot be applied.

To this end, we propose a self-organizing distributed user-content relevance model that enables to organize and recommend multimedia files within the context of a P2P network.

## 2. RELATED WORK

Generically, collaborative filtering is any algorithm that filters information for a user based on a collection of user profiles. One common characteristic of these algorithms is that they require a centralized user-item rating matrix as the input source. These approaches can be divided into two categories: 1) memory-based. Examples are: item correlation-based methods ([6]), item clustering ([8]), user clustering ([15]) and locally weighted regression ([1]). 2) model-based methods. Examples are: decision trees ([1]), latent class models ([5]).

Recently, a few early attempts towards decentralized collaborative filtering have been introduced ([2, 10, 11, 12, 14]). In [10], five architectures are proposed to find and store user rating data to make rating based recommendation, namely, a central server, random discovery similar to Gnutella, transitive traversal, Distributed Hash Tables (DHT), and secure Blackboard. These solutions aim to aggregate the rating data in order to make a recommendation and they hold independently of any semantic structure of the networks. This inevitably increases the amount of traffic within the network. Different with them, we implicitly learn user interests from users' interaction data. Most importantly, since item similarity is calculated and stored by making use of the downloading connections, no extra connections are required.

## 3. SYSTEM DESIGN

In this section, we first describe a relevance model between the user and the multimedia content. We then introduce a method to update these relevance ranks in a distributed and dynamic way. Finally, we present how to make recommen-

dations based on this relevance model.

### 3.1 A User-Content Relevance Model

We consider the following formal setting: Multimedia files (e.g. image, movie, or audio files) are represented as a set of *items*. Without using meta data, each of them can be identified by assigning a unique ID obtained from raw data by some hash functions ([4]). There are  $N$  items, denoted as  $I_a$ ,  $a = 1, \dots, N$ , distributed throughout the network, and  $M$  users<sup>4</sup>, denoted as  $P_i$ ,  $i = 1, \dots, M$ .

We adopt a probabilistic relevance model proposed earlier for text retrieval ([7, 13]). Since a user profile (set of items) represents the current interest of that user, we can treat a user profile as a query and introduce random variables  $r$  and  $\bar{r}$  to denote whether an item is “relevant” or “irrelevant” for the user. To avoid estimating  $P(I_a, P_i)$ , the relevance rank (denoted as  $R_{I_a, P_i}$ ) of the item  $I_a$  for a peer  $P_i$  can be formulated as the odds of relevance:

$$R_{I_a, P_i} = \log \frac{P(r|I_a, P_i)}{P(\bar{r}|I_a, P_i)} \quad (1)$$

By factorizing  $P(\bullet|I_a, P_i)$  with  $\frac{P(P_i|I_a, \bullet)P(\bullet|I_a)}{P(P_i|I_a)}$ , the following log-odds ratio can be obtained:

$$R_{I_a, P_i} = \log \frac{P(r|I_a, P_i)}{P(\bar{r}|I_a, P_i)} = \log \frac{P(P_i|I_a, r)}{P(P_i|I_a, \bar{r})} + \log \frac{P(r|I_a)}{P(\bar{r}|I_a)} \quad (2)$$

In our user profiling method, for the sake of simplicity but without loss of generality, we only observe the positive evidence. By following the language model ([7]), we now assume that 1)  $P_i$  and  $I_a$  are assumed independent in the irrelevant case ( $\bar{r}$ ), i.e.  $P(P_i, |I_a, \bar{r}) = P(P_i|\bar{r})$ ; and, 2) equal priors for both  $P_i$  and  $I_a$ , given that the item is irrelevant. Then the two irrelevance terms can be removed and the relevance rank becomes:

$$R_{I_a, P_i} \propto \log P(P_i|I_a, r) + \log P(r|I_a) \quad (3)$$

Note that these two negative terms in Eq. (2) can always be added to the model when the negative evidences are captured.

#### 3.1.1 Incorporation User Profiles

The items that a user previously interacted with (e.g. watched, played, or downloaded) represent positive evidence for the interest of the user. Let  $L_i$  denote the download list of user  $P_i$ .  $L_i(I_b) = 1$  (or  $I_b \in L_i$ ) indicates that item  $I_b$ ,  $b = 1, \dots, N$ , is in the list while  $L_i(I_b) = 0$  (or  $I_b \notin L_i$ ) otherwise. Thus, a set of items that are downloaded are stored into a *download list* which could represents the user profile and could be treated as query term. We further assume that items are conditionally independent from each other given that they are in the same download list. Although this naive Bayes assumption does not hold in many real situations, it has been empirically shown to be a competitive approach (e.g. in text classification domain ([3]) as well as in our experiments (see Sec. 4)). By using each item in the download list as a query term, then Eq. (3) becomes:

$$R_{I_a, P_i} \propto \sum_{\forall I_b: I_b \in L_i} \log P(I_b|I_a, r) + \log P(r|I_a) \quad (4)$$

<sup>4</sup>Throughout the paper, we use the terms peer and user interchangeably.

When applying the Bayes rule once more:

$$R_{I_a, P_i} \propto \sum_{\forall I_b: I_b \in L_i} \log \frac{P(r|I_a, I_b)}{P(r|I_a)} + \log P(r|I_a) \quad (5)$$

Eq. (5) shows that, in order to make a recommendation (ranking the relevance of a target item towards the user profile), we need to estimate the prior probability of the relevance of an item,  $P(r|I_a)$ , as well as the probability of the relevance between two items  $P(r|I_a, I_b)$ ,  $a, b \in \{1, \dots, N\}$ .

Different from the previous item-based collaborative filtering techniques ([6, 8]), we use a probabilistic framework to convert the initial relevance rank between user and item into a relevance rank between items. By doing so, the prior probability of the item (i.e. the popularity of item) is systematically incorporated into the model. Thus, our model can overcome the disadvantage that it tends to recommend the most popular items in the traditional item based collaborative filtering [6, 8, 10].

### 3.2 Self-organizing Distributed Buddy Tables

In this section, first, we propose a dynamic approach to update the relevance probabilities. Then, we introduce the *buddy table* that stores these relevance ranks and relevance links in a distributed way.

#### 3.2.1 Dynamically Updating Relevance Ranks

If a user likes two items  $I_a$  and  $I_b$ , then this increases the relevance (similarity) of item  $I_a$  with respect to item  $I_b$  (and vice versa). Consequently, in a centralized situation (i.e. when all user profiles are available), the relevance probability of item  $I_a$  and  $I_b$  and the prior relevance probability of an item can be calculated from the profiles of all users (see also, [6]):

$$P(r|I_a, I_b) = \left( \sum_{i=1}^M L_i(I_a) \cap L_i(I_b) \right) / M, \quad (6)$$

$$P(r|I_a) = \left( \sum_{i=1}^M L_i(I_a) \right) / M$$

where  $\sum_{i=1}^M L_i(I_a) \cap L_i(I_b)$  is the number of times that items  $I_a$  and  $I_b$  appear in the same download list (i.e. the co-occur).

In a P2P network the user profiles are, however, distributed throughout the entire network. A naive way to collect user profiles is to broadcast the user profiles throughout the P2P network as in [14, 10] or using DHTs to map keys to profiles [12]. Obviously, this is not efficient. We have found the solution in *dynamically* updating these relevance probabilities.

At a given moment in time, the relevance between two items is updated according to:

$$P_t(r|I_a, I_b) = P_{t-\Delta t}(r|I_a, I_b) + \Delta P_t(r|I_a, I_b) \quad (7)$$

where  $\Delta P_t(r|I_a, I_b)$  is the update of the relevance between two items from time  $t - \Delta t$  to  $t$ . The update is only non-zero when there is a user that downloads one of the items  $I_a$  or  $I_b$  while that user in the past already expressed interest in the opposite item (listed in the download list). Hence, relevance updates only occur when multimedia files are downloaded. The relevance update can thus be expressed in terms of the

transactions that take place between  $t - \Delta t$  and  $t$ , i.e.:

$$\Delta P_t(r|I_a, I_b) = \sum_{\forall T_k: t-\Delta t < k < t} \Delta P_k(r, T_k|I_a, I_b) \quad (8)$$

where  $T_k$  denotes the download transaction at time  $k$ , and the notation  $\Delta P_k(r, T_k|I_a, I_b)$  represents the relevance update between items  $I_a$  and  $I_b$  when considering transaction  $T_k$ .

Since the relevance between  $I_a$  and  $I_b$  is not changed when considering a transaction that does not involve the downloading of either two items, the relevance update can be simplified to:

$$\begin{aligned} \Delta P_t(r|I_a, I_b) = & \sum_{\forall T_k: t-\Delta t < k < t} \Delta P_k(r, I_a = \text{item}(T_k), I_b \in L_{\text{peer}(T_k)}|I_a, I_b) + \\ & \sum_{\forall T_k: t-\Delta t < k < t} \Delta P_k(r, I_b = \text{item}(T_k), I_a \in L_{\text{peer}(T_k)}|I_a, I_b) \end{aligned} \quad (9)$$

and

$$\begin{aligned} \Delta P_k(r, I_b = \text{item}(T_k), I_a \in L_{\text{peer}(T_k)}|I_a, I_b) = \\ \Delta P_k(r, I_a = \text{item}(T_k), I_b \in L_{\text{peer}(T_k)}|I_a, I_b) = 1/M \end{aligned} \quad (10)$$

where  $\text{item}(T_k)$  indicates the item being downloaded in transaction  $T_k$  and  $\text{peer}(T_k)$  indicates the peer that performs the download.  $I_b \in L_{\text{peer}(T_k)}$  means item  $I_b$  is in the downloading list of the peer that performs the download. Eq. (10) indicates that – when  $I_a$  is downloaded by a peer ( $\text{peer}(T_k)$ ) with  $I_b$  in the download list ( $L_{\text{peer}(T_k)}$ ) (or vice versa) – the relevance between item  $I_a$  and  $I_b$  increases  $1/M$ . This is because there is one more download list in which the item  $I_a$  and  $I_b$  both exist together over all the download lists.

A further investigation of Equation (9) shows that the relevance between two items can be updated using only the information about the item ( $\text{item}(T_k)$ ) that is being downloaded and the user profile (e.g.  $L_{\text{peer}(T_k)}$ ) of the peer that is downloading the item. Now we show how to store these between-item relevances and the prior relevances in a *distributed* way.

### 3.2.2 Distributively Updating Relevance Ranks

Equations (8) and (9) show that the between-item relevance probabilities can be calculated incrementally. To store the relevance ranks in a fully distributed way, we propose to store the between-item relevance ranks locally at the location of both items. This is implemented by attaching to each item a so called *buddy table*, which is denoted as  $B_{I_a}$  for item  $I_a$ .

The buddy table stores the information (including an index to their location) about the *top-N* relevant items. The location information can be used to locate items when these items are being recommended to a peer. Most importantly, the buddy tables automatically create a self-organizing semantic overlay that implicitly cluster similar multimedia files (see Sec. 4)).

The relevance ranks stored in the buddy table can be updated according to the following strategy. For each transaction  $T_k$ , the buddy table of the item that is being downloaded ( $I_a = \text{item}(T_k)$ ), is updated, based on the user profile of the peer that performs the download ( $L_{\text{peer}(T_k)}$ ). For all items in user profile,  $\forall I_b : I_b \in L_{\text{peer}(T_k)}$ , the between-item relevance ranks recorded in the buddy table are updated:

$R_{I_a}(I_b) = R_{I_a}(I_b) + 1/M$ . The prior relevance rank is also updated:  $R_{I_a} = R_{I_a} + 1/M$ .

The overall protocol for updating buddy tables can be simply run as a *demon* program in each peer that is only activated in two events: 1) UPLOAD request from other peers (leading to an update of the buddy table of the requested item); and 2) DOWNLOAD request from the local user (leading to sending the user profile to the peer that owns the downloaded file). The protocol is scalable, and simple to implement even on small network-enabled computing devices.

Eq. (9) shows that the relevance probability between item  $I_a$  and  $I_b$  only needs to be updated in two situations: either item  $I_a$  is downloaded while  $I_b$  is in the list, or, vice versa. Careful investigation of the buddy table update protocol shows that the update of this relevance probability is stored in the buddy table of item  $I_a$  while  $I_a$  is being downloaded and stored in the buddy table of item  $I_b$  when  $I_b$  is downloaded. That is, the accumulated relevance probability between the two items in a given time is equal to the sum of the two relevance ranks from the buddy tables of *both* items at that time:

$$P_t(r|I_a, I_b) = R_{I_a}(I_b) + R_{I_b}(I_a) \quad (11)$$

To increase the efficiency and to minimize the communication between the peers, we would like to use *only* one of the relevance ranks stored in either of the two buddy tables to approximate this accumulated between-item relevance probability. It is easily to see that if items in one download list are conditionally independent to each other, the following holds:

$$P_t(r|I_a, I_b) = 2 * R_{I_a}(I_b) = 2 * R_{I_b}(I_a)$$

With this proposition the accumulated relevance probability between the two items can be calculated using only the information stored in one of the buddy tables.

## 3.3 Distributed Recommendation

By using the relevance ranks stored in the buddy tables a recommendation can be generated as follows:

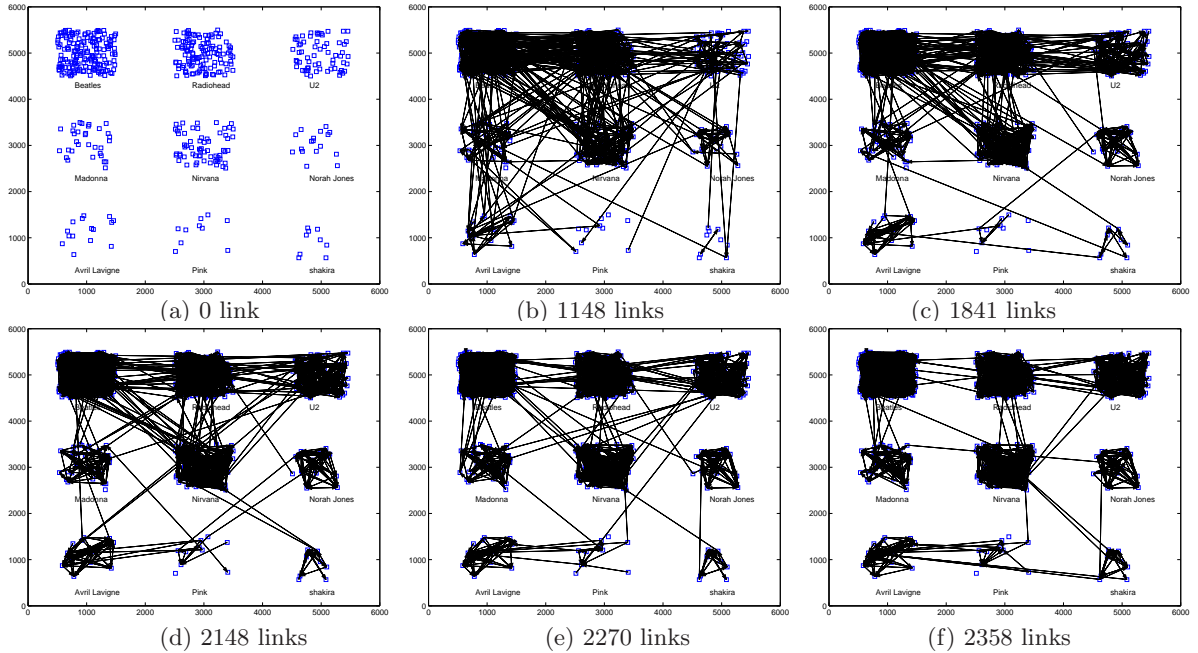
$$R_{I_a, P_i} \propto \sum_{\forall I_b: I_b \in L_i} \log \frac{2 \cdot R_{I_b}(I_a)}{R_{I_a}} + \log R_{I_a} \quad (12)$$

Items in the cache can be safely ignored because their relevance ranks are small (lower than the relevance rank of the  $N$ th item in the buddy table). The final ranking for recommendation then becomes:

$$R_{I_a, P_i} = \sum_{\forall I_b: I_b \in L_i \cap I_a \in B_{I_b}} \log R_{I_b}(I_a) - (q-1) \log R_{I_a} \quad (13)$$

where  $q$  is number of the elements in the first term.  $I_a \in B_{I_b}$  means  $I_a$  is in the buddy table of item  $I_b$  and its relevance rank  $R_{I_b}(I_a)$  is available.

Finally, the procedure to perform a recommendation goes as follows: Firstly, for a given user, the buddy tables of all the items within the profile of the user ( $L_i$ ) are downloaded. Then, for all the items in the collected buddy tables, the relevance ranks towards the user are calculated based on our user-content relevance model (applying Eq. (13)). As a result, the *top-N* ranked items (with their location indicated in the buddy tables) are recommended to the user. This is illustrated in Fig. 1.



**Figure 2: Illustration of dynamically created relevance links between the songs of nine artists. Each song is represented by a blue rectangle. Songs from the same artist are clustered within a grid, resulting in the nine rectangle regions. For clarity, the links to songs of other artists (than the nine showed) have been removed. The panels show the relevance links after (a) 0, (b) 74.906 (iteration 1), (c) 149.812 (iteration 2), (d) 224.718 (iteration 3), (e) 299.624 (iteration 4), and, (f) 374.530 (iteration 5) transactions.**

```

1 /*Input:  $L_i$  download list*/
2 /*Output:  $\{I_a\}$ ; Top-N Recommendation list*/
3 recommend( $L_i$ ) {
4   for  $I_b \in L_i$  { /*For each item in the list*/
5      $\{B_{I_b}\} = \{B_{I_b}\} + B_{I_b}$ ; /*Get buddy tables*/
6   }
7   for  $I_a \in \{B_{I_b}\}$  {
8     get  $R_{I_b}(I_a), R_{I_a}$  from  $\{B_{I_b}\}$ ; /*Get relevance rank*/
9      $R_{I_a}, P_i = \text{eq. (13)}$ ; /*Calculate rec.*/
10  }
11  return topN( $\{I_a, R_{I_a}, P_i\}$ ); /*Return rec. items*/
12 }

```

**Figure 1: Recommendation Procedure.**

## 4. EXPERIMENTS

To validate our proposed self-organizing distributed collaborative filtering approach, we simulated a situation in which users exchange music files on a P2P network.

Since the standard data sets (i.e. movelens) are rating-based, which is not suitable for testing our algorithm. We need to log based user profiles to test our method. The user logs we use are collected from the *Audioscrobber*<sup>5</sup> community. The audioscrobber data set collects the play-lists of the users in the community by using a plug-in in the users' media players (for instance, Winamp, iTunes, XMMS etc). We use 475531 transactions from 3854 userIDs and 10869 itemIDs. The sparsity is 98.86%.

For cross-validation, we randomly divided the data set into a training set (80% of the users) and a test set (20% of the users). For each test user, 50% of the items of a test user were put into the download list of that user (the user profile). The other 50% of the items were used to test the recommendations. We used the training set to calculate the buddy tables, the relevance links and to build the

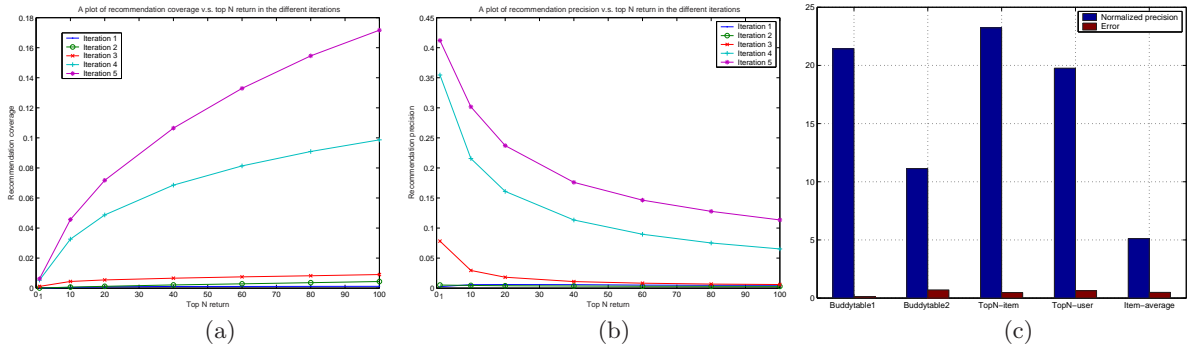
recommendations. The test set was used for evaluating the accuracy of the recommendations.

To simulate media files sharing in the P2P network, we uniformly distributed each item across the different peers. Each peer runs a *demon* program described in Section 3.2.2 to update the buddy tables during each transaction: Each time a transaction takes place (i.e. when a multimedia file is downloaded), the relevance links in the buddy table of the item that is being downloaded are updated. The dynamic behavior of the relevance between items can thus be studied. Figure 2(a) to (f) illustrate the links that were created after: (a) 0, (b) 74.906, (c) 149.812, (d) 224.718, (e) 299.624, and, (f) 374.530 transactions.

The figure shows the songs (items) of nine artists selected such that they reflect different genre of music and different amounts of songs within the database. Songs from the same artist are grouped together. This results in nine clusters shown in the figure. For each item (song), links to the *top-5* relevant items (according to their buddy table) are displayed as directed arrows (pointing outwards the buddy table item). For reasons of clarity, only the links between the displayed items are shown.

From the figure, we observe the following: 1) The number of relevance links increases with an increase in the number of transactions. 2) The relevance links converge and cluster songs (items) of the same artists. This can be seen from the large number of links between songs of the same artist that arises with an increase in the number of transactions. 3) Figure 2 also shows that, besides relevance links between songs by the same artist, relevance links have been created between songs of the same genre (reflecting the interest of a group of users). For instance, relevance links have been created between *U2*, *Radiohead* and *Nirvana*, groups that belong to the rock genre. Links between *Avril Lavigne*,

<sup>5</sup>It can be obtained at [www.audioscrobber.com](http://www.audioscrobber.com)



**Figure 3: Recommendation results.** (a) Coverage as a function of the  $N$  ( $top-N$ ) most relevant items after training using the five different iterations. (b) Similarly for the precision. (c) The normalized precision for: (1) our proposed distributed collaborative filtering approach with the prior term of equation 5; (2) without the prior term; (3) the centralized item-based  $top-N$  suggest method; (4) the centralized user-based  $top-N$  suggest method; and (5) a reference recommendation method based on a average ranking.

*Pink* and *Shakira* may indicate a group of young female pop music artists. *Norah Jones* is somehow isolated since she belongs to the jazz genre, a different style compared to those of the other eight artists.

We treat each user in the test set as a target user in the system. For each target user, a recommendation was calculated by applying the algorithm described in Fig. 1, based on the calculated buddy tables from the training users. The resulting recommended items were then compared to the ground truth items.

First, we investigated the impact of user interaction (transaction) on the recommendation performance. The coverage and precision of the recommendation in the five iterations are shown in Fig. 3 (a) and (b). The results indicate that as the number of transactions increases, the recommendation results become better.

Next, we compared our distributed collaborative filtering approach with the  $Top-N$  suggest recommendation engine, a well-known centralized collaborative filtering approach ([6])<sup>6</sup>. Both the item-based version and the a user-based version were compared. The parameters had been set according to the user manual. Additionally, we compared the three recommenders to a non-personalized recommendation approach. For each item, its prior relevance  $P(r|I_T)$  was used. The items were then ranked and recommended accordingly.

For our distributed approach, we show the results of two different settings: one with the prior relevance (second term of Eq. (5)) and one without it.

In order to make a fair comparison, we adopted a normalized precision ([9]) to compare the methods. We normalized the precision according to the precision of a random recommendation of the same number of elements. The normalized precision of the five methods is shown in Fig. 3 (c). It shows that the performance of our distributed recommendation is comparable to that of the centralized methods. Our approach using prior relevance outperforms even the centralized  $top-N$  user-based recommendation method and approximates the best  $top-N$  item-based method.

## 5. CONCLUSIONS

In this paper, we have proposed a self-organizing distributed user-item relevance model. This has been realized by introducing the concept of buddy tables, which are

attached to items that are distributed throughout a P2P network. Simulation experiments with music play-list data showed that our approach is a promising technique for recommendation in a P2P network.

## 6. REFERENCES

- [1] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, 1998.
- [2] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. of ACM ICIR*, 1999.
- [3] S. Eyheramendy, D. Lewis, and D. Madigan. On the naive bayes model for text categorization. In *Proc. of Artificial Intelligence and Statistics*, 2003.
- [4] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *Proc. of Intl Conf on Music Information Retrieval*, 2002.
- [5] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of IJCAI*, 1999.
- [6] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proc. of the tenth international conference on Information and knowledge management*, 2001.
- [7] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. *Language Modeling and Information Retrieval, Kluwer International Series on Information Retrieval*, V.13, 2003.
- [8] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, Jan/Feb.:76–80, 2003.
- [9] B. Marlin. Collaborative filtering: a machine learning perspective. Master's thesis, Department of Computer Science, University of Toronto, 2004.
- [10] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004.
- [11] T. Oka, H. Morikawa, and T. Aoayama. Vineyard : A collaborative filtering service platform in distributed environment. In *Proc. of the IEEE/IPSJ Symposium on Applications and the Internet Workshops*, 2004.
- [12] H. Peng, X. Bo, Y. Fan, and S. Ruimin. A scalable p2p recommender system based on distributed collaborative filtering. *Expert systems with applications*, 2004.
- [13] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. of SIG-IR*, 1998.
- [14] A. Tveit. peer-to-paper based recommendation for mobile commerce. In *Proc. of the First International Mobile Commerce Workshop*, pages 26–29, 2001.
- [15] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR '05*, 2005.

<sup>6</sup><http://www-users.cs.umn.edu/~karypis/suggest/>