Konstantin Pussep, Sebastian Kaune, Jonas Flick, Ralf Steinmetz:
A Peer-to-Peer Recommender System with Privacy Constraints. In: Leonard Barolli, Fatos Xhafa, Spain Hui-Huang Hsu: Third International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC-2009), p. 409--414, IEEE Computer Society Press, March 2009. ISBN 978-0-7695-3575-3.

A Peer-to-Peer Recommender System with Privacy Constraints

Konstantin Pussep, Sebastian Kaune, Jonas Flick, Ralf Steinmetz
Technische Universität Darmstadt, Germany
E-Mail: Konstantin.Pussep@kom.tu-darmstadt.de

Abstract

A recommender system *can be used to suggest users potentially interesting content based on their previous consumption behavior. Such services already became common in centralized systems, such as Amazon, and approaches exist for decentralized recommender systems. However, common P2P recommender systems expose the user's preferences in the whole system. This is not desirable if* privacy *is required.*

Realization of a recommender system in a private P2P environment is not a trivial task, since we cannot gather the user data at central servers or just spread them in the community. In this work we propose a private file sharing application based on social contacts. Instead of gathering all the information about users at one place the users exchange information only with their social contacts. We show how a personalized recommender system can be built in such an environment.

1. Introduction

Recommender systems enable users to find new interesting yet unknown content based on their estimated preferences. This allows to find relevant content, e.g. files, products or pictures, despite a huge amount of available data. Such services are already common in centralized systems, such as YouTube or Amazon, where users can receive recommendations generated based on their previous consumption behavior. Here, a central entity collects the user behavior information and analyzes it to find similar patterns.

For peer-to-peer (P2P) systems several recommender systems have been proposed, that can operate in different environments, e.g. Distributed Hash Tables (DHTs), unstructured networks, or super-peer based networks. Here instead of analyzing the information about users in a centralized database, the data about user's preferences or item popularity is spread in the network. These approaches work fine as long as user's privacy does not become an issue.

However, considering a scenario where users share content that should not be visible to everybody, but only to their "friends"[1], the unrestricted forwarding of user's preference is not desirable. For example, in social networks, such as Facebook, StudiVZ or Flickr, users can restrict the access to their data. A similar restriction can also apply to a P2P social network where users exchange their private information only with authorized friends [11]. In order to achieve *privacy*, a P2P recommender service that works in such an environment must assure that the information about user's interests and content offered and consumed by him is not exposed to arbitrary users.

In this work we analyze the feasibility of privacy conserving recommender systems in social networks. We aim to recommend files, such as pictures, knowledge documents, movies, without restricting the algorithms to any specific domain. Furthermore, we don't restrict ourselves to online social networks, such as Facebook, but consider more generally any kind of social networks that connect users with social relationships.

The contributions of this paper are: At first, we show how to build a private social file-sharing network where users exchange their data only with their friends Secondly, on top of this network a recommender system is built to filter content that matches user's interests. We show that a suitable recommendation algorithm can take advantage of *user similarity* in social networks and offer good recommendation quality while presuming privacy, even compared with algorithms that utilize global knowledge.

In Section 2 we discuss the related work on P2P recommender systems. Our system model is presented in Section 3, including the proposed private overlay and system's architecture. The recommendation algorithms considered are presented in Section 4 and evaluated in Section 5. Finally, Section 6 concludes the paper.

[1] We consider users being friends if there is a direct link between them in the underlying social network. This link can base upon private or professional relationship, depending on what kind of content the user wants to share.

2. Related Work

Recommender algorithms are a substantial research topic, both in the industry and academic community [4, 2, 3, 13, 12, 9]. Besides the established industrial players, such as Google and Amazon, that rely on centralized databases [2, 3], several decentralized approaches have been proposed, e.g. [7, 12, 9].

In general the recommendation algorithms can be divided in two categories [3]:

- User similarity based approaches try to identify users with similar interests and then to recommend content possessed by these users.
- Object similarity based approaches try to find items similar to those already possessed by the given user.

These algorithms cannot be easily applied to our scenario because we don't have the knowledge about the complete user base (unlike centralized solutions) or even a significant part of the network (unlike classical P2P file sharing networks where the content owned by users is exposed to the whole community). Instead, we have to restrict the data exchange to related users.

Therefore, centralized recommender systems [2, 3] require a global knowledge about users behavior in the system (e.g. who buys what on Amazon, or who watches which movies on YouTube) are not applicable.

Considering the algorithms proposed for P2P systems we discover that they still rely on a semi-global knowledge, e.g. on super-peers analyzing search requests from normal peers [4]. These super-peers then are able to collect user data and, therefore, contradict privacy requirements. Further systems, such as [1], rely on a Distributed Hash Table to manage the recommendation information. However, peers have only few control over the data that is stored in a DHT, and typical DHTs allow arbitrary peers to fetch and store information. Our approach is different as we don't have super-peers or any other entities gathering knowledge about other peers. Instead, we aim to restrict the information exchange to friends only.

Ruffo et al. [9] proposed an unstructured "preference network" that offers a recommendation mechanism. Differently to our approach it does not consider how to distribute the user's preference and to preserve user's privacy.

Distributed Collaborative Filtering [12] generates file recommendations based on the global popularity of items/files in the system. This approach was designed for classical file sharing applications where a user can access content shared by any other users and, therefore, distributes the information about shared content in the network. We consider this algorithm as a reference system while proposing a recommender algorithm that presumes user's privacy.

3. System Model

In order to realize a private P2P recommender system we propose to use an existing social network to leverage existing user relationships. We use the network's API to discover user's friends and exchange information with them. This *social overlay* is then used to build a *private file sharing overlay* by deciding which friends are allowed to access content shared by a user.

We argue that such an overlay offers a high degree of user's similarity due to the following observation: In social networks users can organize themselves in *groups* that express their interest in some specific topic. The members of such a group do not have to maintain a friendship relationship. Nevertheless, as measured by representative studies, e.g. by Mislove et al. in [5], the groups are highly clustered, i.e. on average many of the users in a group are also friends in the friendship overlay.

Our assumption is that if friends mostly share common interests regarding the interest groups, they would also share common interest in content, e.g. files. Therefore, we propose to build a private file sharing service on top of such a social network. The basic functionality is then to offer content for sharing and to browse the content shared by friends. Furthermore, in order to deal with a big amount of data (i.e. music collections, pictures, movies, or literature databases) a recommender system allows users to discover new relevant content.

In order to assure privacy only authorized friends of a user are allowed to see and access the shared content. This results in a private social overlay for file sharing.

3.1. Private Overlay

Community networks such as Facebook or Orkut already offer open APIs and more networks announced to support the Opensocial API developed by Google [6]. In fact, any network that connects users with their friends can be used, not only an online social network, but even an instant messaging service where users are connected to other users and can obtain their online status and exchange messages. All we need is an API with the following primitives:

- getListOfContacts(): List
- getOnlineContacts(): List
- sendMsgToContact(Contact C, Message M): void
- receiveMsg(): Message

3.2. Architecture

Figure 1 shows the architecture of our system. Social networks with open APIs are used to find friends in available social networks and to connect to these peers. The private overlay works as a substrate on top of social networks

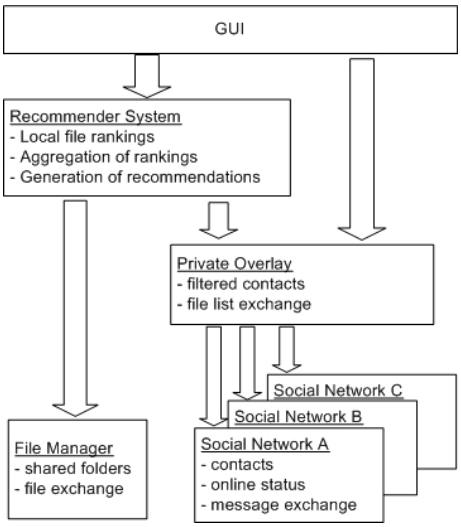


Figure 1. Application Architecture

to communicate among users that mutually granted access to their shared content. Our recommender module uses the private overlay to collect information about friends and their preferences in order to recommend content. Finally, a file exchange component manages the exchange of files among users.

We built a first prototype as a proof-of-concept. It uses Skype[2] to obtain the network of friends. This is done via Skype's open API[3] that can be used by third-party tools such as our application. The recommender system runs on top of the private overlay composing Skype users running our application and exchange the file lists together with the ranking information using the Skype API.

3.3. System Operation

Our system works as follows: once the user starts the application for the first time he defines a directory with the content he wants to share. Then the system connects to the supported social networks, and loads the lists of contacts. The user is asked which of his friends should be allowed to access the shared content.

Once the content exchange permit is granted by users the normal operation mode takes place. Here the system exchanges in the background the lists of shared files and recommendation metadata, such as popularity of single files. This is done in two ways:

- The *push mechanism* issues each time a user connects to the social network or adds new content to the shared folder a list update and sends it to connected sharing friends.
- The *pull mechanism* requests the list of shared content from friends each time the application is started.

Additionally to the list of shared content the file rankings are exchanged among friends. Based on this ranking information and the content lists received from sharing friends the recommender system tries to identify the content mostly interesting to the user and suggests this content for download. The user receives the list of recommended files and can decide whether to download them or not. This decision is then used as feedback to the recommender service.

4. P2P Recommendation Algorithms

Before we introduce the privacy-conserving recommender algorithm used in our system, we discuss two comparison algorithms, being representative for the related work. The first approach, File Popularity Recommendations (FPR) [4], works completely locally using the list of own files and the lists of neighbors' files as data basis. This algorithm is expected to work well if the interest similarity among neighbors is high. Contrary, the second approach, Distributed Collaborative Filtering (DCF) [12], relies on the global knowledge of file popularities in the network. Therefore, it can operate in any environment, regardless whether the neighbors in the network share similar interests or not. As such a system contradicts the privacy requirements, we consider DCF only for comparison purposes, to estimate the relative performance of privacy-conserving algorithms.

4.1. Files' Popularity based Recommendation (FPR)

This approach can be classified as object similarity based. For all items within the network (or files within a cluster of friends in our case) a *popularity ranking* is computed. This ranking uses a reference file to determine the relative popularity of other files among users, that already have this item. The user supplies a reference file f_x to the recommender service to obtain a list of files, popular with users that already own f_x. More precisely the popularity of a file f_k is computed as follows:

$$Pop(f_k) = \frac{|peers\ owning\ f_k\ and\ f_x|}{|peers\ owning\ f_x|} \qquad (1)$$

A recommendation will then consist of all files, whose popularity is bigger than a certain threshold value. We adopted this algorithm with a threshold of 10 %, meaning that only file's with a popularity of at least 0.1 are recommended.

This algorithm requires the recommendation information being exchanged among friends only but users have to explicitly choose a reference file.

[2] www.skype.com

[3] https://developer.skype.com/wiki/Java_API

4.2. Distributed Collaborative Filtering (DCF)

Distributed Collaborative Filtering [12] is an attempt to utilize the recommendation quality of traditional collaborative filtering techniques [10] in a decentralized environment. It uses an object similarity based approach, which is based on the so-called item relevance ranks.

The main *relevance rank* describes the relative popularity of an item in the system:

$$R_{I_a} = \frac{|Peers\ owning\ I_a|}{|Peers\ in\ the\ network|} \quad (2)$$

This global value, however, does not reflect the degree of similarity among items. For this purpose *between-item relevance rank* metric defines the relative popularity of items I_a and I_b:

$$R_{I_a}(I_b) = \frac{|Peers\ owning\ both\ Items\ I_a\ and\ I_b|}{|Peers\ in\ the\ network|} \quad (3)$$

In a decentralized environment these relevance ranks can not be computed for all items, if not all the information about the network topology is known. Therefore, every peer stores so-called *buddy tables* for each item the peer owns. A buddy table contains the approximated relevance rank of the item and all approximated between-item relevance ranks to all the other items the peer owns or knows about. These ranks are distributed in the network, whenever another peer exchanges information with this peer. The relevance rank $R(I_a)$ is updated dynamically by adding $1/|Peers\ in\ the\ network|$ every time a new user that owns I_a is found. The between-items relevance ranks to all own files I_b are also updated in the same manner. This way the more information is exchanged, the better the approximation of the ranks becomes.

For a list of items received from another peer the final recommendation ranking for each item I_a for a user P_i can be computed based on the relevance ranks as follows:

$$R_{I_a,P_i} = \sum_{\forall I_b \in B_i(I_a)} \log R_{I_b}(I_a) \ -(|B_i(I_a)|-1)\log R_{I_a} \quad (4)$$

Whereas $B_i(I_a)$ is the list of peer P_i's own items whose buddy tables contain I_a.

4.3. Limitations of DCF

In our scenario there are two issues about this algorithm:

1. The ranking computation of R_{I_a} and $R_{I_a}(I_b)$ requires the knowledge of the network size that is unknown in a P2P system. This value has to be estimated, e.g. by counting the unique users found in the buddy tables of files. Therefore, user ids are stored in the buddy tables.
2. The buddy tables of peer's items must be propagated in the network to achieve good results. Due to the user ids stored in the buddy tables this contradicts the privacy constraint. While some techniques, such as hashing of ids, might reduce the visible information, it still cannot avoid the undesirable exposure of user's data. An eavesdropper would just compare the hashes of known users with the hashes found in the buddy tables.

4.4. Privacy-Conserving Distributed Collaborative Filtering (PCDCF)

In the original implementation of the DCF algorithm every peer distributes the buddy tables that contain user ids. Therefore, this information is shared in the entire network and contradicts the privacy requirements. We refined the algorithm in such a way that peers store only the estimated relevance ranks (R_{I_a} and $R_{I_a}(I_b)$) of the files he knows about together with the *number of peers* that contributed to the calculation of this metric (we denote this number of peers as P_{I_a}).

The computation of the final recommendation ranks after a peer receives a list with shared files from a friend is done as follows:

For every file Ia in the list, that is not already owned by the receiver, the new rank R_{I_a} is obtained by calculating the average based on the previous relevance ranks of the user himself and his friends. For a set F of friends of the given user f' that means:

$$R^{f'}_{I_a} = \frac{\sum_{f \in F \cup \{f'\}} P^f_{I_a} \cdot R^f_{I_a}}{\sum_{f \in F} P^f_{I_a}} \quad (5)$$

Where $R^f_{I_a}$ is the rank computed by user f and $P^f_{I_a}$ is the number of contributors to this rank.

The rank $R_{I_a}(I_b)$ is calculated accordingly. Finally, the overall recommendation rank is computed as given by expression 4.

In contrast to traditional DCF, to generate recommendations our algorithm requires only to exchange the values P_{I_a}, R_{I_a}, $P_{I_a}(I_b)$, and $R_{I_a}(I_b)$ that do not contain any private information about other peers. The peers are only able to see the list of files shared by their direct friends and anonymized relevance ranks.

Table 1 summarizes the most relevant properties of considered algorithms. FPR distributes information only with direct friends, while DCF spreads the information in the whole network. In turn, PCDCF exchanges anonymized file statistics with direct neighbors to compute recommendations. Furthermore, FPR requires a manually selected reference file.

Table 1. Recommender algorithms.

	Data Dissemination	Reference File
FPR	neighbors only	required
DCF	global	no
PCDCF	gossiping	no

5. Evaluation of Recommendation Algorithms

5.1. Evaluation Goal and Metrics

In order to show the feasibility of our approach we evaluate the performance of the selected recommender algorithms in our scenario. We compare the performance of the all three considered algorithms: localized version of FPR, normal DCF and our privacy conserving DCF (PCDCF).

The metrics applied to evaluate the algorithms performance are:

- **Coverage** determines how many files the user is interested in were recommended by the algorithm.
- **Accuracy** reflects the fraction of interesting files compared to all recommended files.

We applied the *k-fold validation method* to evaluate the recommendation algorithms [8]: Here the files assigned to a user are divided into **k** subsets. For each simulation run, one of the subsets is removed and used for validation purposes. This method is repeated **k** times with different subsets of missing items and the average coverage and accuracy are calculated.

5.2. Setup

We evaluate the selected algorithms in a round-based simulator. The system is modeled as a network of friends, where each user initially stores a set of files. Both the number of friends and files per user are set to 300. Each round peers exchange their lists of the shared content and the recommendation rankings with their neighbors. Based on this information, the local recommendations are generated and the files matching users' interests are downloaded.

In order to create a network of friends we applied a power-law distribution because it is the behavior observed in many social networks [5]. Most users have few contacts only while few users have many contacts. Similar distributions were applied to generate popularity of single files, i.e. the number of copies per file available in the system and the number of files per user.

The system is initialized in three steps: In the first step a number of unique files is divided into 10 ranks. The ranks have power-law properties where few files have many replicas and many files only few replicas.

Then the files are assigned to users, again following a power-law distribution. Finally, in the last step the friendship overlay is created where some users have many friends while many users have only few. In order to model the interest similarity among friends, some friends are selected using a cosine similarity measure while the rest is selected randomly to express potential interest diversity.

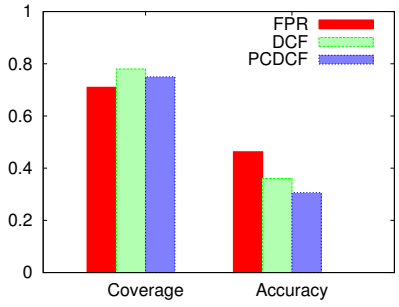


Figure 3. Comparative results of recommendation algorithms.

5.3. Results

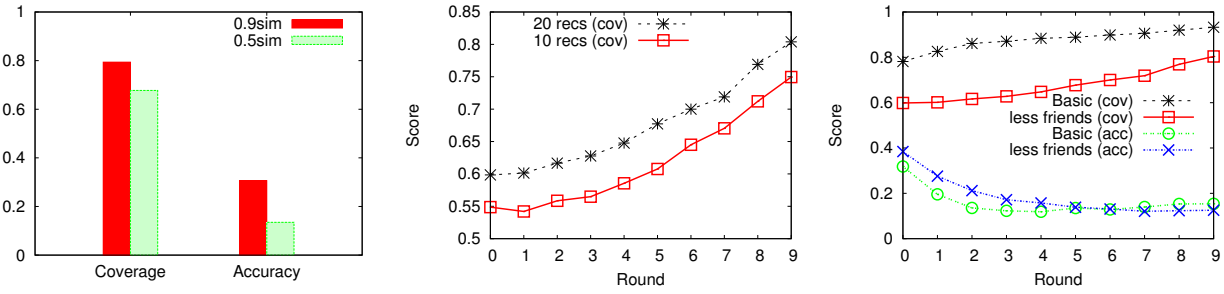
The comparative evaluation of all three considered recommender algorithms is shown in Figure 3. Here the peers are friends of peers with similar interests with the probability of 90%. It turns out that DCF achieves the highest coverage of almost 80%, while PCDCF and FPR achieve slightly lower values of 75 and 71% respectively. This shows that in an environment with a high user similarity, even simple algorithms such as FPR allow to find relevant content.

Additionally, Figure 3 shows the accuracy of the algorithms. Here, the DCF algorithm achieves the accuracy of 36%, while the accuracy of FPR is 10% higher (46%). On the other hand our PCDCF offers a slightly lower accuracy of 31%. This is a good result for PCDCF if we take into account that it shares the private information only with direct friends (unlike DCF) and does not require a reference file (like FPR does).

In order to assess the impact of interest similarity among friends we modified the probability that friends share similar interests from 90% to 50%. The results in Figure 2(a) show that for PCDCF the coverage drops from almost 80 to 67% and the accuracy from 30 to 14%. This confirms our expectation that the quality of our recommender algorithm decrease depends on the similarity among friends.

We further investigated the impact of different parameters on the performance of PCDCF. Therefore, we simulated our system over ten rounds. We reduced the number of recommendations for each peer from 20 to 10. As we can see in Figure 2(b) the coverage drops by 5% only.

In another experiment we reduced the number of friends



(a) Impact of friends similarity (PCDCF) (b) Impact of recommendations' limit (PCDCF) (c) Impact of number of friends (PCDCF)

Figure 2. Performance results for PCDCF (network of 300 peers and 300 files).

per peer by 50%. The results in Figure 2(c) show that the accuracy stayed the same (around 16%) while the coverage drooped by almost 20%. That means that peers with many friends can find more relevant content.

We conclude that PCDCF is a good candidate to build a recommender system in a private file-sharing network where users are connected according to the similarity of interests. Unlike pure DCF it preserves user's privacy and does not require a reference file as FPR does. PCDCF offers good recommendation accuracy and allows peers to find relevant content.

6. Conclusions and Future Work

In this paper, we have presented a recommender system for private P2P file sharing. Our approach uses existing social relationships to build a private social network for file sharing and a recommender system on top of it. We presented a privacy-conserving recommender algorithm and compared its performance with two extremes: an algorithm that exchanges file lists only among neighbors and an algorithm that distributes file popularity rankings including user data in the entire system. Our algorithm performs well compared to the privacy-violating algorithm and outperforms the simple approach in terms of coverage. The results have shown that it is possible to offer good recommendation quality (both regarding coverage and accuracy) under the assumption of overlapping interests of friends in the social network.

There are two directions for the future work: we aim to build and deploy an application based on our current prototype. This would allow us to gather data about recommendation's quality in the real life. Additionally, we will develop a service to discover new friends with similar interests by exploring the P2P network. This way the system would not just utilize existing social networks but also actively shape them.

References

[1] P. Han, B. Xie, F. Yang, and R. Shen. A scalable p2p recommender system based on distributed collaborative filtering. *Expert Systems With Applications*, 27(2):203–210, 2004.

[2] G. Karypis. *Evaluation of Item-Based Top-N Recommendation Algorithms*. ACM New York, NY, USA, 2001.

[3] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[4] L. Mekouar, Y. Iraqi, and R. Boutaba. Personalized Recommendations in Peer-to-Peer Systems. In *WETICE Collaborative Peer-to-Peer Systems Workshop (COPS'08)*, 2008.

[5] A. Mislove, M. Marcon, P. K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Internet Measurement Conference*, 2007.

[6] J. Mitchell-Wong, R. Kowalczyk, A. Roshelova, B. Joy, and H. Tsai. Opensocial: From social networks to social ecosystem. pages 361–366, 2007.

[7] J. Pouwelse et al. Tribler: A social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 2007.

[8] G. Ruffo and R. Schifanella. Evaluating peer-to-peer recommender systems that exploit spontaneous affinities. In *ACM symposium on Applied computing*, 2007.

[9] G. Ruffo, R. Schifanella, and E. Ghiringhello. A decentralized recommendation system based on self-organizing partnerships. In *NETWORKING 2006*.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *10th international conference on World Wide Web*, 2001.

[11] A. Tootoonchian, K. Gollu, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: Social access control for web 2.0. In *WOSN 2008*, 2008.

[12] J. Wang, J. Pouwelse, R. Lagendijk, and M. J. T. Reinders. Distributed collaborative filtering for p2p file sharing systems. In *ACM Symposium on Applied Computing*, 2006.

[13] G. Xue et al. Scalable collaborative filtering using cluster-based smoothing. In *28th ACM SIGIR conference on Research and development in information retrieval*, 2005.