Personalized Communities in a Distributed Recommender System

Sylvain Castagnos, Anne Boyer

Personalized Communities in a Distributed Recommender System

CASTAGNOS Sylvain and BOYER Anne

LORIA - Université Nancy 2
Campus Scientifique - B.P.239
54506 Vandoeuvre-lès-Nancy Cedex, France
{sylvain.castagnos, anne.boyer}@loria.fr

Abstract. The amount of data exponentially increases in information systems and it becomes more and more difficult to extract the most relevant information within a very short time. Among others, collaborative filtering processes help users to find interesting items by modeling their preferences and by comparing them with users having the same tastes. Nevertheless, there are a lot of aspects to consider when implementing such a recommender system. The number of potential users and the confidential nature of some data are taken into account. This paper introduces a new distributed recommender system based on a user-based filtering algorithm. Our model has been transposed for Peer-to-Peer architectures. It has been especially designed to deal with problems of scalability and privacy. Moreover, it adapts its prediction computations to the density of the user neighborhood.

1 Introduction

With the development of information and communication technologies, the size of information systems all over the world has exponentially increased. Consequently, it becomes harder and harder for users to identify relevant items in a reasonable time, even when using a powerful search engine. Collaborative filtering techniques [1] are a good way to cope with this difficulty. It amounts to identifying the active user to a set of persons having the same tastes, based on his/her preferences and his/her past actions. This system starts from the principle that users who liked the same items have the same topics of interest. Thus, it is possible to predict the relevancy of data for the active user by taking advantage of experiences of a similar population.

There are several fundamental problems when implementing a collaborative filtering algorithm. In this paper, we particularly pay attention to the following significant limitations for industrial use:

- scalability and system reactivity: there are potentially several thousand users and items to manage in real time;
- intrusions into privacy: we have to be careful to be as unintrusive as possible and at least to guarantee the anonymity of users;

– novelty in predictions: according to the context, users want to have more or less new recommendations. Sometimes their main concern is to retrieve the items that they have high-rated, even if it means having less new recommendations. This is why we introduce an adaptive minimum-correlation threshold of neighborhood which evolves in accordance with user expectations.

We propose an algorithm which is based on an analysis of usage. It relies on a distributed user-based collaborative filtering technique. Our model has been integrated in a document sharing system called "SofoS".[1]

Our algorithm is implemented on a Peer-to-Peer architecture because of the document platform context. In a lot of companies, documents are referenced using a common codification that may require a central server[2] but are stored on users' devices. The distribution of computations and contents matches the constraints of scalability and reactivity.

In this paper, we will first present the related work on collaborative filtering approaches. We will then introduce our Peer-to-Peer user-centered model which offers the advantage of being fully distributed. We called this model "Adaptive User-centered Recommender Algorithm" (AURA). It provides a service which builds a virtual community of interests centered on the active user by selecting his/her nearest neighbors. As the model is ego-centered, the active user can define the expected prediction quality by specifying the minimum-correlation threshold. AURA is an anytime algorithm which furthermore requires very few computation time and memory space. As we want to constantly improve our model and the document sharing platform, we are incrementally and modularly developing them on a JXTA platform[3].

2 Related work

In centralized collaborative filtering approaches, finding the closest neighbors among several thousands of candidates in real time without offline computations may be unrealistic [2]. By contrast, decentralization of data is practical to comply with privacy rules, as long as anonymity is fulfilled [3]. This is the reason why more and more researchers investigate various means of distributing collaborative filtering algorithms. This also presents the advantage of giving the property of profiles to users, so that they can be re-used in several applications.[4] We can mention research on P2P architectures, multi-agents systems and decentralized models (client/server, shared databases).

There are several ways to classify collaborative filtering algorithms. In [4], authors have identified, among existing techniques, two major classes of algorithms: memory-based and model-based algorithms. Memory-based techniques

[1] SofoS is the acronym for "Sharing Our Files On the System".
[2] This allows to have document IDs and to identify them easily.
[3] http://www.jxta.org/
[4] As the owner of the profile, the user can apply it to different pieces of software. In centralized approaches, there must be as many profiles as services for one user.

offer the advantage of being very reactive, by immediately integrating modifications of users profiles into the system. They also guarantee the quality of recommendations. However, Breese *et al.* [4] are unanimous in thinking that their scalability is problematic: even if these methods work well with small-sized examples, it is difficult to change to situations characterized by a great number of documents or users. Indeed, time and space complexities of algorithms are serious considerations for big databases. According to Pennock *et al.* [5], model-based algorithms constitute an alternative to the problem of combinatorial complexity. Furthermore, they perceive in these models an added value beyond the function of prediction: they highlight some correlations in data, thus proposing an intuitive reason for recommendations or simply making the hypotheses more explicit. However, these methods are not dynamic enough and they react badly to insertion of new contents into the database. Moreover, they require a penalizing learning phase for the user.

Another way to classify collaborative filtering techniques is to consider user-based methods in opposition to item-based algorithms. For example, we have explored a distributed user-based approach within a client/server context in [6]. In this model, implicit criteria are used to generate explicit ratings. These votes are anonymously sent to the server. An offline clustering algorithm is then applied and group profiles are sent to clients. The identification phase is done on the client side in order to cope with privacy. This model also deals with sparsity and scalability. We highlight the added value of a user-based approach in the situation where users are relatively stable, whereas the set of items may often vary considerably. On the contrary, Miller *et al.*[7] show the great potential of distributed item-based algorithms. They propose a P2P version of the item-item algorithm. In this way, they address the problems of portability (even on mobile devices), privacy and security with a high quality of recommendations. Their model can adapt to different P2P configurations.

Beyond the different possible implementations, we can see there are a lot of open questions raised by industrial use of collaborative filtering. Canny [3] concentrates on ways to provide powerful privacy protection by computing a "public" aggregate for each community without disclosing individual users' data. Furthermore, his approach is based on homomorphic encryption to protect personal data and on a probabilistic factor analysis model which handles missing data without requiring default values for them. Privacy protection is provided by a P2P protocol. Berkovsky *et al.* [8] also deal with privacy concern in P2P recommender systems. They address the problem by electing super-peers whose role is to compute an average profile of a sub-population. Standard peers have to contact all these super-peers and to exploit these average profiles to compute predictions. In this way, they never access the public profile of a particular user. We can also cite the work of Han *et al.*[9], which addresses the problem of privacy protection and scalability in a distributed collaborative filtering algorithm called PipeCF. Both user database management and prediction computation are split between several devices. This approach has been implemented on Peer-to-Peer overlay networks through a distributed hash table method.

In this paper, we introduce a new hybrid method called AURA. It combines the reactivity of memory-based techniques with the data correlation of model-based approaches by using an iterative clustering algorithm. Moreover, AURA is a user-based model which is completely distributed on the user scale. It has been integrated in the SofoS document platform and relies on a P2P architecture in order to distribute either prediction computations, content or profiles. We design our model to tackle, among others, the problems of scalability, and privacy.

3 SofoS

SofoS is a document platform, using a recommender system to provide users with content. Once it is installed, users can share and/or search documents, as they do on P2P applications like Napster. We conceive it in such a way that it is as open as possible to different existing kinds of data: hypertext files, documents, music, videos, etc. The goal of SofoS is also to assist users to find the most relevant sources of information efficiently. This is why we add the AURA recommender module to the system. We assume that users can get pieces of information either by using our system or by going surfing on the web. SofoS consequently enables to take visited websites into account in the prediction computations.

We are implementing SofoS in a generic environment for Peer-to-Peer services, called JXTA. This choice is motivated by the fact it is greatly used in our research community.

In [7], the authors highlight the fact that there are several types of possible architectures for P2P systems. We can cite those with a central server (such as Napster), random discovery ones[5] (such as Gnutella or KaZaA), transitive traversal architectures, content addressable structures and secure blackboards.

We conceived our model with the idea that it could be adapted to different types of architectures. However, in this paper, we will illustrate our claims by basing our examples on the random approach even if others may have an added value. The following subsection aims at presenting the AURA Algorithm.

3.1 AURA Algorithm

We presume that each peer in SofoS corresponds to a single user on a given device.[6] For this reason, we have conceived the platform in such a way that users have to open a session with a login and a password before using the application. In this way, several persons can use the same computer (for example, the different members of a family) without disrupting their respective profiles. That is why each user on a given peer of the system has his/her own profile and a single ID. The session data remain on the local machine in order to enhance privacy. There

[5] Some of these architectures are totally distributed. Others mixed centralized and distributed approaches but elect super-peers whose role is to partially manage sub-groups of peers in the system.

[6] We can easily distinguish devices since SofoS has to be installed on users' computers.

is no central server required since sessions are only used to distinguish users on a given peer.

For each user, we use a hash function requiring the IP address and the login in order to generate his/her ID on his/her computer. This use of a hash function H is suitable, since it has the following features:

- non-reversible: knowing "y", it is hard to find "x" such as $H(x) = y$;
- no collision: it is hard to find "x" and "y" such as $H(x) = H(y)$;
- knowing "x" and "H", it is easy to compute $H(x)$;
- H(x) has a fixed size.

In this way, an ID does not allow identification of the name or IP address of the corresponding user. The communication module uses a IP multicast address to broadcast the packets containing addressees' IDs. In order to reduce the information flow, we can optionally elect a super-peer which keeps a list of IDs whose session is active: before sending a message, a peer can ask if the addressee is connected. If the super-peer has no signal from a peer for a while, it removes the corresponding ID from the list.

Users can both share items on the platform and integrate a feedback about websites they consult. Each item has a profile on the platform. In addition to the available documents, each peer owns 7 pieces of information: a personal profile, a public profile, a group profile and 4 lists of IDs (list "A" for IDs of peers belonging to its group, list "B" for those which exceed the minimum-correlation threshold as explained below, list "C" for the black-listed IDs and list "O" for IDs of peers which have added the active user profile to their group profile). An example of the system run is shown on figure 1.

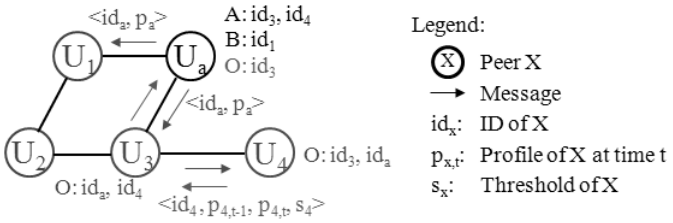

Fig. 1. Run of AURA.

In order to build the personal profile of the active user u_a, we use both explicit and implicit criteria. The active user can always check the list of items that he/she shares or has consulted. He/She can explicitly rate each of these items on a scale of values from 1 to 5. The active user can also initialize his/her personal profile with a set of criteria[7] proposed in the interface in order to partially face the cold start problem. This offers the advantage of completing the profile with more consistency and of finding similarities with other users more quickly, since everyone can fill the same criteria rating form.

[7] Ideally, the set of items in the criteria set should cover all the implicit categories that users can find on the platform.

We assume that, despite the explicit voluntary completion of profiles, there are a lot of missing data. We consequently add to AURA a user modeling function, as we did in [6]. The explicit ratings and the estimated numerical votes constitute the active user's personal profile. The public profile is the part of the personal profile that the active user accepts to share with others.

The algorithm also has to build a group profile. It represents the preferences of a virtual community of interests, and has been especially designed to be as close as possible to the active user's expectations. In order to do that, the peer of the active user asks for the public profiles of all the peers it can reach through the platform. Then, for each of these profiles, it computes a similarity measure with the personal profile of the active user. The active user can indirectly define a minimum-correlation threshold which corresponds to the radius of his/her trust circle (cf. infra, figure 2).

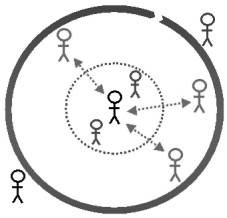

Fig. 2. Virtual community centered on u_a.

If the result is lower than this fixed threshold which is specific to each user, the ID of the peer is added to the list "A" and the corresponding profile is included in the group profile of the active user, using the procedure of table 1.

Procedure AddToGroupProfile(public profile of u_n)
  $W = W + |w(u_a, u_n)|$
  for each item i **do**
    $(u_{l,i}) = (u_{l,i}) * (W - |w(u_a, u_n)|)$
    $(u_{l,i}) = ((u_{l,i}) + w(u_a, u_n) * (u_{n,i}))/W$
  end for

With: $(u_{l,i})$ the rating for item i in the group profile;
$(u_{n,i})$ the rating of user n for item i;
W the sum of $|w(u_a, u_i)|$, which is stored;
$w(u_a, u_n)$ the correlation coefficient between the active user u_a and u_n.

Table 1. Add a public profile to the group profile.

We used the Pearson correlation coefficient to compute similarity, since the literature shows it works well [10]. Of course, if this similarity measure is higher than the threshold, we add the ID of the peer to the list "B". The list "C" is used to systematically ignore some peers. It enables to improve trust – that is to say the confidence that users have in the recommendations – by identifying malicious users. The trust increasing process will not be considered in this paper.

When his/her personal profile changes, the active user has the possibility to update his/her public profile p_a. In this case, the active peer has to contact every

peer[8] whose ID is in the list "O". Each of these peers re-computes the similarity measure. If it exceeds the threshold, the profile p_a has to be removed from the group profile, using the procedure of table 2. Otherwise, p_a has to be updated in the group profile, that is to say the peer must remove the old profile and add the new one.

Procedure RemoveToGroupProfile(old profile of u_n)
$W = W - \|w(u_a, u_n)\|$
for each item i **do**
$(u_{l,i}) = (u_{l,i}) * (W + \|w(u_a, u_n)\|)$
$(u_{l,i}) = ((u_{l,i}) - w(u_a, u_n) * (u_{n,i}))/W$
end for

Table 2. Remove a public profile from the group profile.

By convention, we use the notation $< id, p >$ for the peer-addition packet, that is to say new arrivals. $< id, p, s >$ corresponds to the packet of a peer which is already connected and sends data to a new arrival. "s" is the threshold value. There is no need to specify the threshold value in the peer-addition packet, since there is a default value ($|correlation| >= 0$). At last, $< id, p_{t-1}, p_t, s >$ is the notation for the update packet. In each of these packets, the first parameter corresponds to the ID of the source of the message. In order to simplify the notation, we do not include the addressees' ID in figure 3.

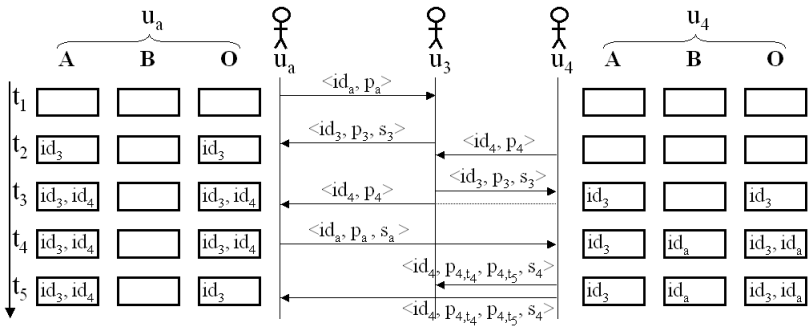

Fig. 3. Example of user interactions.

Figure 3 illustrates how the system works. In this example, we consider 3 of the 5 users from figure 1. We show the registers of the active user u_a and the user u_4. At time t_1, the active user u_a tries to contact, for the first time, other peers by sending his/her public profile and his/her ID to neighbors. This is the packet $< id_a, p_a >$. u_3 receives the packet and answers at t_2. u_a computes the distance between the public profiles p_3 and p_a. As the Pearson coefficient is inevitably within the default threshold limit, u_a adds id_3 to his/her list "A". If the computed correlation coefficient is higher than "s_3" which is the threshold of u_3, u_a adds id_3 to his/her list "O". Meanwhile, some of the reached peers will

[8] A packet is broadcasted with an heading containing peers' IDs, the old profile and the new public profile.

add p_a to their list "A" if the correlation is higher than their threshold (this is the case for u_3). At time t_2, u_4 arrives on the platform and sends a packet to u_3. At time t_3, u_3 replies to u_4 and sends the packet of u_4 to peers that he/she already knows. u_a receives it and adds id_4 to his/her list "A". He/She also adds id_4 to the list "O", since u_4 is a new arrival and has a default threshold. At time t_4, u_a consequently gives his/her public profile to u_4. At the same time, u_4 has changed his/her threshold and considers that u_a is too far in the user/item representation space, that is to say the correlation coefficient between u_a and u_4 exceeds the limit. Thus, u_4 adds id_a in the list "B". In the packet $< id_a, p_a, s_a >$, "s_a" allows u_4 to know that he/she must complete the list "O" with id_a. At last, u_4 updates his/her public profile. Afterwards, he/she notifies the change to the IDs in the list "O". This is the packet $< id_a, p_{4,t_4}, p_{4,t_5}, s_4 >$. p_{4,t_4} and p_{4,t_5} are respectively the old and new public profiles of u_4. When u_a receives this packet, he/she updates the list "O" by removing id_4 since s_4 is too high for him/her.

3.2 Adaptive minimum-correlation threshold

As shown in the previous subsection, the active user can indirectly define the minimum-correlation threshold that other people must reach in order to be a member of his/her community (radius of the circle on figure 2). Concretely, a high correlation threshold means that users taken into account in prediction computations are very close to the active user. Recommendations will be consequently extremely similar to his/her own preferences. On the contrary, a low correlation threshold sets forth the will of the active user to stay aware of generalist information by integrating distant users' preferences. In this way, the user avoids freezed suggestions by accepting novelty. In the SofoS interface, a slide bar allows the active user to ask for personalized or generalist recommendations. This allows AURA to know the degree to which it can modify the threshold[9]. The default threshold value is 0, which means that we take all the peers into account. The default step of threshold is 0.1, but it can be adapted to the density of population.

As shown in figure 4, we split the interval of the Pearson coefficient's possible values $[-1; +1]$ into subsets. For each subset, we keep the count of peers which have got in touch with the active user and whose correlation coefficient is contained in the interval corresponding to the subset. Thus, when a user sends a packet to u_a, the Pearson coefficient is computed in order to know if the active user's group profile has to be updated according to the threshold value. At the same time, we update the corresponding values in the population distribution histogram. For example, if u_a receives an update packet and the Pearson coefficient changes from 0.71 to 0.89, we decrement the register of the interval $[0.7; 0.8)$ and we increment the register of the interval $[0.8; 0.9)$. In this way, we constantly have the population density for each interval.

[9] By "threshold", we mean the minimum absolute value of Pearson coefficients to consider in the group profile computation. For example, if the system sets the threshold to 0.1, it means that only peers u_i whose correlation coefficient $|w(u_a, u_i)|$ is higher than 0.1 will be included in the group profile of the active user.

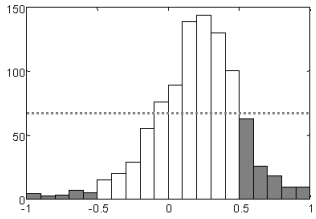

Fig. 4. Adaptive threshold based on density.

When the total number of users whose Pearson coefficient is higher than $(threshold + 0.1)$ exceeds a given limit (dashed line on figure 4), we increase the threshold. If there are too many users in the next subset, the threshold increase is lower. For the moment, the maximum threshold value is 0.2 for users who want a high degree of novelty and 0.9 for those who expect recommendations close to their preferences.[10] These values have been arbitrarily chosen. We plan to do statistical tests to automatically determine the ideal thresholds according to the context.

4 Discussion

In order to define the degree of privacy of our recommender system, we refer to 4 axes of personalization [11]. Cranor assumes that an ideal system should be based on an explicit data collection method, transient profiles, user initiated involvment and non-invasive predictions. In our system, the users have complete access to their preferences. They have an effect on what and when to share with others. Only numerical votes are exchanged and the logs of user actions are transient. Even when the active user did not want to share his/her preferences, it is possible to do predictions since public profiles of other peers are temporarily available on the active user device. Each user has a single ID, but the anonymity is ensured by the fact that there is no table linking IDs and identities. This privacy-enhanced process requires more network traffic than in [8], but it allows the system to perform user-centered rather than community-centered predictions.

As regards scalability, our model no longer suffers from limitations since the algorithms used to compute group profiles and predictions are in $o(b)$, where b is the number of commonly valuated items between two users, since computations are made incrementally in a stochastic context. In return, AURA requires quite a lot of network traffic. This is particularly true if we use a random discovery architecture. Other P2P structures can improve communications [7].

Furthermore, we assume that quality of predictions in real situation should be better – providing that we found enough neighbors – since the virtual community of interests on each peer is centered on the active user (cf. infra, figure 2). We can influence the degree of personalization by adjusting the threshold according to

[10] That is to say they want to retrieve items that they have high-rated

the density of the active user's neighborhood. The system just has to increase the threshold in order to ensure users to retrieve the items that they have high-rated among their recommendations. To highlight this phenomenon, we generated a rating matrix of 1,000 users and 1,000 items. The votes follow a gaussian law and we can see the average number of neighbors as regards Pearson coefficient scaling on figure 5. We randomly removed 20% of these votes and applied the AURA algorithm. Then, we compute the Recall which measures how often a list of recommendations contains an item that the user have already rated in his/her top 10. When increasing the threshold in the system, this measure becomes higher.

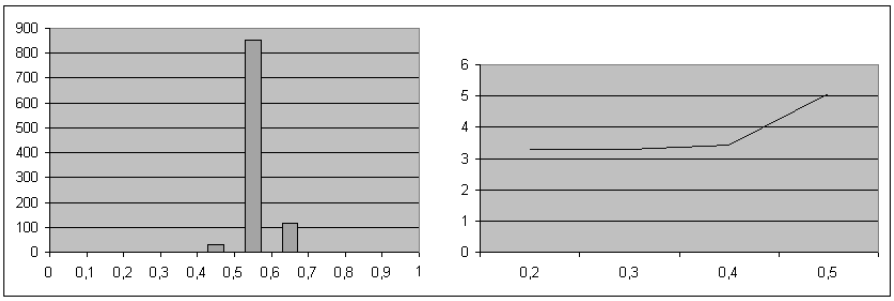

Fig. 5. On the left, average distribution of users as regards Pearson coefficient. On the right, recall as threshold grows.

We have also evaluated our model in terms of prediction relevancy. We used the *Mean Absolute Error* (`MAE`). `MAE` is a widely used metric which shows the deviation between predictions and real user-specified values. Consequently, we computed the average error between the predictions and 100,000 ratings of the GroupLens test set[11].

We simulate arrivals of peers by progressively adding new profiles. As shown on figure 6, we get predictions as good as using the PocketLens algorithm [7]. PocketLens relies on a distributed item-based approach. This comparison consequently demonstrates that AURA provides as relevant results as a performant item-based collaborative filtering.

At last, we compared our recommender system with two centralized algorithms (Item-Item [2] and the Correlation-based Collaborative Filter CorrCF [12]) to illustrate the added value of the distributed approach. In order to determine the computation times of these algorithms, we have generated random public profiles with different numbers of items. In this simulation, the votes of each user follow a Gaussian distribution centered on the middle of the representation space. Moreover, only 1% of data in the generated profiles is missing.[12] Since the Item-Item and CorrCF are centralized, we first aggregate the profiles in a vote matrix.

[11] http://www.grouplens.org/

[12] Only 1% of missing data is not realistic but can potentially increase the computation time what is interesting in this case.

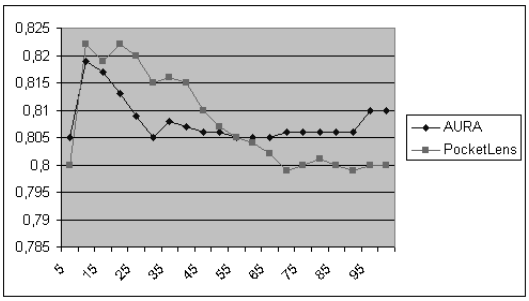

Fig. 6. MAE as neighborhood size grows.

The results of the tests in term of computation time are shown in the table 3. The announced times for the AURA algorithm do not include the duration required to scan the network in search of public profiles. Of course, the difference between AURA and the two others is mainly due to the fact that we use as many peers as users for computations. However, these results illustrate the considerable gain in comparison with centralized techniques. AURA allows to do real-time predictions. There is no need to do offline computations since we can take into account 10,000 profiles and 150 items in less than an half-second. Moreover, the system does not have to wait until all similarity measures end. As the algorithm is incremental, we can stop considering other peers at any moment.

Items	100			150			1000		
Users	AURA	CorrCF	It-It	AURA	CorrCF	It-It	AURA	CorrCF	It-It
200	0”01	2”60	2”14	0”01	3”17	2”71	0”07	11”09	52”74
1,000	0”03	30”22	8”56	0”05	40”68	12”84	0”30	3’06”	3’25”
10,000	0”31	7:30’	1’22”	0”48	-	2’05”	1”90	-	49’28”
100,000	3”04	-	-	-	-	-	-	-	-

Table 3. Computation times of three collaborative filtering algorithms.

5 Conclusion

SofoS is a document sharing platform including a recommender system. To cope with numerous problems specific to information retrieval, we proposed a Peer-to-Peer collaborative filtering model which is totally distributed. It allows real-time personalization and manages the degree of personalization that users want. We implement it on a JXTA platform which has been used by researchers all over the world. We show in this paper that we can deal with important problems such as scalability, privacy and quality. We highlight the benefits of our system by doing offline performance analysis. We plan on validating these points by testing our model with real users in real conditions.

Our algorithm is anytime and incremental. Contrary to PocketLens, our model is user-based because we consider that the set of items can change. Even

if an item is deleted, we can continue to exploit its ratings in the prediction computations. Moreover, the stochastic context of our model allows the system to update the modified profiles instead of resetting all the knowledge about neighbors. At last, our model is very few memory-consuming because it does not need to store any neighbors' ratings, similarity matrix, dot product matrix and so on. It only requires the sum of pearson coefficients and four lists of user IDs.

Currently, we are developing our protocols further to cope with other limitations, such as trust and security aspects by using specific communication protocols as in [13].

References

1. Goldberg, D., Nichols, D., Oki, B., Terry, D.: Using collaborative filtering to weave an information tapestry. In: Communications of the ACM, Special Issue on Information Filtering. Volume 35(12)., ACM Press (1992) 61–70
2. Sarwar, B.M., Karypis, G., Konstan, J.A., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: World Wide Web. (2001) 285–295
3. Canny, J.: Collaborative filtering with privacy. In: IEEE Symposium on Security and Privacy, Oakland, CA (May 2002) 45–57
4. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), San Francisco, CA (July 1998)
5. Pennock, D.M., Horvitz, E., Lawrence, S., Giles, C.L.: Collaborative filtering by personality diagnosis: a hybrid memory- and model-based approach. In: Proceedings of the sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000), San Francisco, USA, Morgan Kaufmann Publishers (2000)
6. Castagnos, S., Boyer, A.: A client/server user-based collaborative filtering algorithm: Model and implementation. In: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI2006), Riva del Garda, Italy (August 2006)
7. Miller, B.N., Konstan, J.A., Riedl, J.: Pocketlens: Toward a personal recommender system. In: ACM Transactions on Information Systems. Volume 22. (July 2004)
8. Berkovsky, S., Eytani, Y., Kuflik, T., Ricci, F.: Hierarchical neighborhood topology for privacy enhanced collaborative filtering. In: in CHI 2006 Workshop on Privacy-Enhanced Personalization (PEP2006), Montreal, Canada (April 2006)
9. Han, P., Xie, B., Yang, F., Wang, J., Shen, R.: A novel distributed collaborative filtering algorithm and its implementation on p2p overlay network. In: Proc. of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD04), Sydney, Australia (May 2004)
10. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating "word of mouth". In: Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. Volume 1. (1995) 210–217
11. Cranor, L.F.: Hey, that's personal! In: the International User Modeling Conference (UM05). (2005)
12. Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, ACM (1994) 175–186
13. Polat, H., Du, W.: Svd-based collaborative filtering with privacy. In: Proc. of ACM Symposium on Applied Computing, Cyprus (2004)