



ELSEVIER

Expert Systems with Applications 27 (2004) 203–210

Expert Systems  
with Applications

[www.elsevier.com/locate/eswa](http://www.elsevier.com/locate/eswa)

# A scalable P2P recommender system based on distributed collaborative filtering

Peng Han\*, Bo Xie, Fan Yang, Ruimin Shen

*Department of Computer Science and Engineering, Shanghai Jiaotong University, 6th Floor Haoran High-tech Building, Shanghai 200030, China*

## Abstract

Collaborative Filtering (CF) technique has been proved to be one of the most successful techniques in recommender systems in recent years. However, most existing CF based recommender systems worked in a centralized way and suffered from its shortage in scalability as their calculation complexity increased quickly both in time and space when the record in user database increases. In this article, we first propose a distributed CF algorithm called PipeCF together with two novel approaches: significance refinement and unanimous amplification, to further improve the scalability and prediction accuracy. We then show how to implement this algorithm on a Peer-to-Peer (P2P) structure through distributed hash table method, which is the most popular and efficient P2P routing algorithm, to construct a scalable distributed recommender system. The experimental data show that the distributed CF-based recommender system has much better scalability than traditional centralized ones with comparable prediction efficiency and accuracy.

© 2004 Elsevier Ltd. All rights reserved.

*Keywords:* Recommender system; Collaborative filtering; Peer-to-Peer; Significance refinement; Unanimous amplification

## 1. Introduction

Recommender system is a system that helps users to find their wanted items by making recommendations based on either the content of the recommended items (Content-based Filtering), or ratings of similar users on the recommended items (Collaborative Filtering, CF). Since Goldberg, Nichols, Oki, and Terry (1992) published the first account of using CF for information filtering; CF has proved to be one of the most successful techniques in recommendation systems by its advantage of that no explicit description of items is needed. The key idea of CF is that users will prefer those items that people with similar interests prefer, or even that dissimilar people do not prefer, so most CF algorithms can be separated into three steps as addressed by Herlocker, Konstan, Borchers, and Riedl (1999): (1) Similarity Weight: weight all users with respect to similarity with the active user, which refer to the user whose preferences are to be predicted; (2) Selecting Neighborhoods: select those users used to make prediction; (3) Rating Normalization and Prediction Making: normalize and calculate the weighted sum of selected users' ratings, then make prediction based on that. According to different techniques used in the first

part mentioned above, CF algorithms can be divided into two classes: memory-based algorithms and model-based algorithms. Breese et al. performed an empirical analysis on both of two kinds of CF algorithms in Breese, Heckerman, and Kadie (1998) while Herlock et al. presented an algorithmic framework for performing CF in Herlocker et al. (1999).

GroupLens (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994) was the first CF algorithm to automate prediction and used a memory-based algorithm. Like most memory-based algorithms, GroupLens need to compute across the whole user database to calculate the similarities between active user and other users to make prediction. Ringo (Shardanand & Maes, 1995) only used those neighbors whose correlation were greater than a given threshold to make prediction. This approach not only reduced the calculation complexity but also proved to improve the performance. By choosing top-N users with the highest correlations the same improvement can also been obtained. However, all the other users' similarities still have to be calculated and its complexity increased quickly both in time and space as the record in the database increases.

Basically, there are two ways to reduce this calculation complexity. The first one is used a model-based algorithm which first constructs some certain

\* Corresponding author. Tel./fax: +86-21-62933205.  
E-mail address: phan@mail.sjtu.edu.cn (P. Han).

mathematical models, such as Bayesian Network, Bayesian Classifiers et al., to describe the users and/or their ratings, then learns these models from the database and use them to make prediction. However, these approaches also need complex calculation when compiling models and also require a central database to keep all the user data which is not easy to achieve sometime not only for techniques reasons but also for privacy reasons.

The second way is to implement CF in a decentralized way. In fact, as Peer-to-Peer (P2P) gains more and more popularity, some researchers have already begun to consider it as an alternative architecture to reduce the calculation complexity (Canny, 2002; Olsson, 2003; Tveit, 2001) of centralized CF algorithms. The main difference between centralized CF-based recommender system and distributed ones is that the originally centralized user database are maintained in a decentralized way which means each peer will only keep a fraction of user database and when making prediction for a particular user, needed record should first be retrieved to the user's own database from other peers and calculated locally. In order to do this, the following two problems have to be addressed: (1) how to store the user database distributed efficient so that needed information can be found efficiently; (2) how to identify those records needed to make prediction for a particular user and fetch them efficiently as retrieving all other users' votes back is not only unreasonable but also unnecessary.

The main contributions of this article are:

- (1) We propose PipeCF: a distributed CF algorithm which can be implemented on a P2P overlay network;
- (2) We propose two novel approaches: significance refinement (SR) and unanimous amplification (UA), to improve the performance of our distributed CF algorithm;
- (3) We give the framework of implementing our distributed CF algorithm on P2P overlay network through distributed hash table (DHT) based technique to obtain efficient user database management and retrieval to construct decentralized CF recommender system.

The rest of this article is organized as follows. In Section 2, several related works are presented and discussed. In Section 3, we introduce the architecture and key features of our DHT-based CF system. Two techniques: SR and UA are also proposed in this section to improve the scalability and prediction accuracy of DHT-based CF algorithm in this section. In Section 4 the experimental results of our system are presented and analyzed. Finally we make a brief concluding remark and give the future work in Section 5.

## 2. Related work

### 2.1. Memory-based CF algorithm

Generally, the task of CF is to predict the votes of active users from the user database which consists of a set of votes  $v_{i,j}$  corresponding to the vote of user  $i$  on item  $j$ . Memory-based CF algorithm calculates this prediction of as a weighted average of other users votes on that item through the following formula:

$$P_{a,j} = \bar{v}_a + \kappa \sum_{i=1}^n \varpi(a,j)(v_{i,j} - \bar{v}_i) \quad (1)$$

where  $P_{a,j}$  denotes the prediction of the vote for active user  $a$  on item  $j$  and  $n$  is the number of users in user database.  $\bar{v}_i$  is the mean vote for user  $i$  as:

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j} \quad (2)$$

where  $I_i$  is the set of items on which user  $i$  has voted. The weights  $\varpi(a,j)$  reflect the similarity between active user and users in the user database.  $\kappa$  is a normalizing factor to make the absolute values of the weights sum to unity.

Most memory-based algorithms use Eq. (1) to make prediction and only distinguish between the ways they calculate the weights:

#### 2.1.1. Pearson correlation coefficient

Pearson correlation coefficient was first introduced into collaborative filtering as a weighting method in the GroupLens project The correlation between user  $a$  and  $i$  is:

$$\varpi(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} \quad (3)$$

where the summations is calculated over those items for which both users  $a$  and  $i$  have voted.

#### 2.1.2. Vector similarity

The vector similarity was first used to measure the similarity between two documents. Each document was viewed as a vector of word frequency and their similarity was computed as the cosine of the angle between these two vectors. In Collaborative Filtering, we treat each user record as a document and their votes as frequency of items. So the weights can now be calculated as:

$$\varpi(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}} \quad (4)$$

## 2.2. P2P system and DHT routing algorithm

The term ‘Peer-to-Peer’ refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner. With the pervasive deployment of computers, P2P is increasingly receiving attention in research and more and more P2P systems have been deployed on the Internet. Some of the benefits of a P2P approach include: improving scalability by avoiding dependency on centralized points; eliminating the need for costly infrastructure by enabling direct communication among clients; and enabling resource aggregation. Among all these applications, three main classes of peer-to-peer applications have emerged: parallelizable, content and file management, and collaborative.

As the main purpose of P2P systems are to share resources among a group of computers called peers in a distributed way, efficient and robust routing algorithms for locating wanted resource is critical to the performance of P2P systems. Among these algorithms, distributed hash table (DHT) algorithm is one of the most popular and effective and supported by many P2P systems such as CAN (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001), Chord (Stocal et al., 2001), Pastry (Rowstron & Druschel, 2001), and Tapestry (Zhao et al., 2001).

A DHT overlay network is composed of several DHT nodes and each node keeps a set of resources (e.g. files, rating of items). Each resource is associated with a key (produced, for instance, by hashing the file name) and each node in the system is responsible for storing a certain range of keys. Peers in the DHT overlay network locate their wanted resource by issue a *lookup(key)* request which returns the identity (e.g. the IP address) of the node that stores the resource with the certain key. The primary goals of DHT are to provide an efficient, scalable, and robust routing algorithm which aims at reducing the number of P2P hops, which are involved when we locate a certain resource, and to reduce the amount of routing state that should be preserved at each peer. In Chord (Stocal et al., 2001), each peer keeps track information of  $\log N$  other peers ( $N$  is the total number of peers in the community). When a peer joins and leaves the overlay network, this highly optimized version of DHT algorithm will only require notifying  $\log N$  peers about that change.

## 3. Our distributed CF algorithm

### 3.1. Basic PipeCF algorithm

The first step to implement CF algorithm in a distributed way is to divide the original centralized user database into fractions which can be stored in Peers distributed. For concision, we use the term *bucket* to denote one fraction of the whole user database in the following of this article.

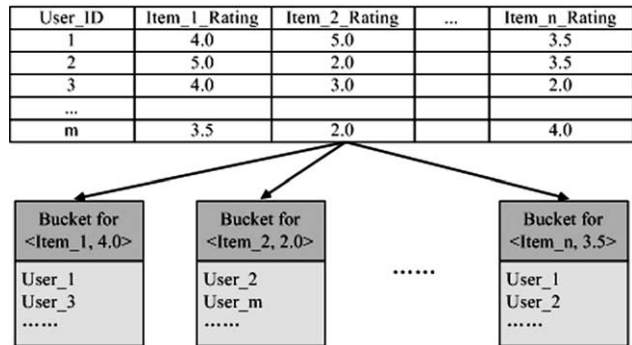


Fig. 1. User database division strategy.

Each bucket should also be assigned an identifier through which they can be located later when needed. The way we do the division is to make each bucket hold a group of users' record who has at least rated one item with the same vote. By that we construct one bucket for every different  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple and use that tuple as the identifier for the bucket. Fig. 1 shows our division strategy:

In order to reduce the calculation complexity and achieve scalability, we wish to find a strategy which can choose neighbors from the most suitable buckets when making prediction. The PipeCF algorithm chooses neighbors based on the heuristic that people with similar interests at least rate one item with similar votes. As we can see in Fig. 2, this strategy have very high hitting ratio. So when making prediction, the PipeCF only uses those users' records who are in at least one same bucket with the active users. Through which we reduce about 50% calculation than traditional CF algorithm and obtain comparable prediction as shown in Figure 8.

### 3.2. Improved PipeCF algorithm

#### 3.2.1. Significance refinement

In the basic PipeCF algorithm, we use all users which are in the same bucket with the active user and find that the algorithm has an  $O(N)$  fetched user number ( $N$  is the total user number) as Fig. 8 shows. In fact, as Breese et al. (1998) presented by the term inverse user frequency, universally liked items are not as useful as less common items in capturing similarity. So we introduce a new concept SR, which reduces the returned user number of the basic PipeCF algorithm by limiting the number of returned users for each bucket. We term the algorithm improved by SR as Return K which means 'for every item, the PipeCF algorithm returns no more than K users for each bucket'. The experimental result in Section 5.3.3 shows that this method reduces the returned user number dramatically and also improves the prediction accuracy.

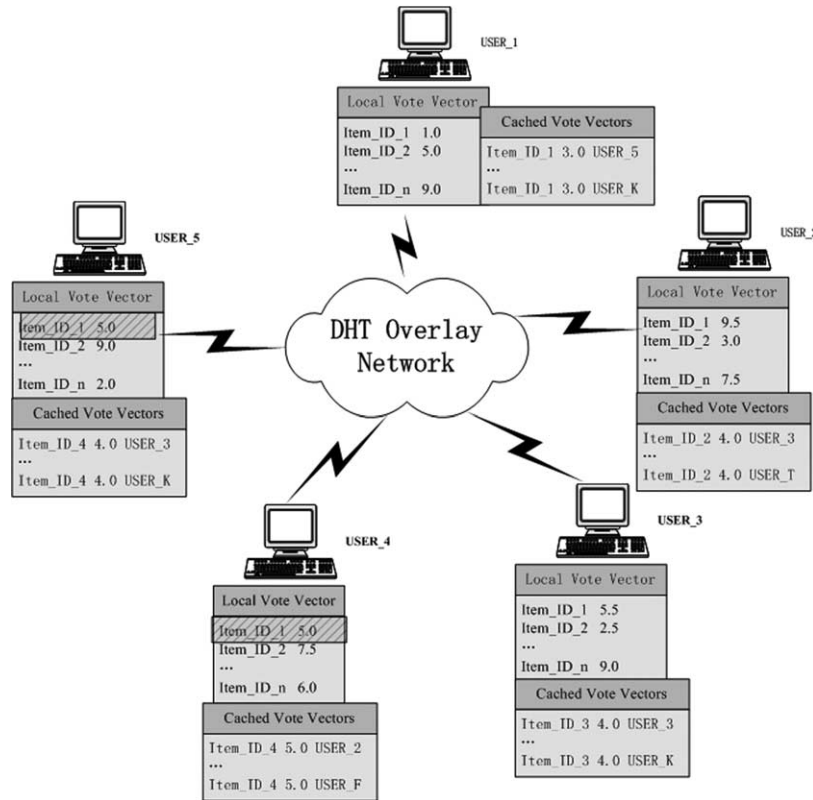


Fig. 2. Architecture of DHT-based CF recommender system.

### 3.2.2. Unanimous amplification

Enlightened by the method of case amplification (Breese et al., 1998) which emphasizes the contribution of the most similar users to the prediction by amplifying the weights close to 1, we argue that we should give special award to the users who rated some items with the same vote by amplify their weights, which we term UA. We transform the estimated weights as follows:

$$w'_{a,i} = \begin{cases} w_{a,i} & N_{a,i} = 0 \\ w_{a,i}\alpha & 0 < N_{a,i} \leq \gamma \\ w_{a,i}\beta & N_{a,i} > \gamma \end{cases} \quad (5)$$

where  $N_{a,i}$  denotes the number of items which user  $a$  and user  $i$  have the same votes. A typical value for  $\alpha$  for our experiments is 2.0,  $\beta$  is 4.0, and  $\gamma$  is 4. Experimental result in Section 4.3.4 shows that UA approach improves the prediction accuracy of the PipeCF algorithm.

## 4. DHT-BASED CF recommender system

### 4.1. Architecture of DHT-based CF recommender system

The main advantage of our DHT-based CF recommender system vs. traditional centralized CF recommender system is that both the maintenance of user database and the complex computation task of making prediction

are done in a decentralized way so as to obtain better scalability. Here, we treat each bucket as resource and the unique key for the resource is generated by the particular  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple associated with the bucket. Each peer in the DHT overlay network will keep one or several buckets locally. Fig. 2 gives the architecture of our DHT-based CF recommender system.

So the implementation of PipeCF in DHT overlay network is straightforward except that the bucket is stored distributed so that when a user wants to look up other similar users which have the same particular  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple, it need fetch them from DHT overlay network. Still we need special scheme to manage the distributed storage of buckets. DHT has provided two function *lookup(key)* and *put(key)* which we will describe later to do these jobs and its efficiency has been guaranteed by the algorithm itself. So with the DHT overlay network, all the users in the CF system are connected together and can find their wanted similar neighbors efficiently through a DHT routing algorithm.

### 4.2. Implementation of PipeCF on DHT

On the basis of the decentralized storage of user votes, we introduce our implementation of PipeCF algorithm on the DHT overlay network, called DHT-based CF algorithm, as shown in Fig. 3.

**Algorithm:** DHT-based collaborative filtering  
**Input:** training set, test set, a target item and a given top  $K$  selection  
**Output:** mean absolute error of prediction  
**Method:**

- 1) Construct a DHT overlay network (we simulate its two main functions: *put(key)* and *lookup(key)*).
- 2) Every user of training set hashes  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  as a key and put all the keys to the DHT overlay network by calling *put(key)* function.
- 3) Each instance of the test set fetches similar neighbors from DHT overlay network by calling *lookup(key)* function. All fetched users' votes construct a local training set LOCAL\_TRAINING\_SET.
- 4) Each instance of the test set searches the top  $K$  neighbors from LOCAL\_TRAINING\_SET and computes the corresponding prediction for the target item.
- 5) Compute the mean absolute error of prediction and output.

Fig. 3. DHT-based CF algorithm.

There are two key pieces to the DHT-based CF system algorithm: the lookup mechanism used to locate similar users and fetch their actual rating. The decentralized storage (and hence decentralized retrieval) in decentralized CF system makes the CF calculation inherently scalable (every user do recommendation locally instead of depending on a centralized server); the hard part is finding the similar peers from which to retrieve the actual rating.

We devise a scalable solution to the problem of locating similar users in decentralized CF system, i.e. give a user vote vector; we can find the IP address of the node(s) which is similar to the user. Our DHT-based solution can reach following goal:

- *Scalability:* it must be designed to scale to several million nodes.
- *Efficiency:* similar users should be located reasonably quick and with low overhead in terms of the message traffic generated.
- *Dynamicity:* the system should be robust to frequent node arrivals and departures in order to cope with highly transient user populations' characteristic to decentralized environments.
- *Balanced load:* in keeping with the decentralized nature, the total resource load (traffic, storage, etc) should be roughly balanced across all the nodes in the system.

So we only select similar users in the subset in which users have same  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple. The key idea of our algorithm is hashing every user for every rated item. Our DHT-based CF algorithm includes two main DHT

**Algorithm:** DHT-based CF puts a peer  $P$ 's vote vector to DHT overlay network  
**Input:** test set ( $P$ 's vote vector)  
**Output:** NULL  
**Method:**

- 1)  $P$  generates a unique 128-bit DHT Key  $K_{local}$  (i.e. hash the system unique username).
- 2)  $P$  hashes one  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple to key  $K$ , and routes it with test set ( $P$ 's vote vector) to the neighbor  $P_i$  whose local key  $K_{i\_local}$  is the most similar with  $K$ .
- 3) When  $P_i$  receives the PUT message with  $K$ , it caches it. And if the most similar neighbor is not itself, it just routes the message to its neighbor whose local key is most similar with  $K$ .
- 4) For each rated item,  $P$  repeats step 2 and 3.

Fig. 4. DHT-based CF put function.

functions: *put(key)* and *lookup(key)*, and Figs. 4 and 5 show them separately.

DHT-based CF Put algorithm is used to construct DHT overlay network and fill data in it. DHT-based CF Lookup algorithm is used to lookup and fetch similar uses with same  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple in order to construct a local training set to make recommendation. The main purpose of steps 2 and 3 in Fig. 3 is to make every peer in the DHT overlay network keep several buckets which contain a group of users with same  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple, from which the Lookup algorithm can fetch similar users later in its steps 2 and 3.

**Algorithm:** DHT-based CF lookups similar users for a peer  $P$   
**Input:** test set ( $P$ 's vote vector)  
**Output:** train set (retrieved similar users vote vectors)  
**Method:**

- 1)  $P$  generates a unique 128-bit DHT Key  $K_{local}$  (i.e. hash the system unique username).
- 2)  $P$  hashes one  $\langle \text{ITEM\_ID}, \text{VOTE} \rangle$  tuple to key  $K$ , and routes it with test set ( $P$ 's vote vector) to the neighbor  $P_i$  whose local key  $K_{i\_local}$  is the most similar with  $K$ .
- 3) When  $P_i$  receives the LOOKUP message with  $K$ , if  $P_i$  has enough cached vote vectors with the same key  $K$ , it returns the vectors back to  $P$ , otherwise it just routes the message to its neighbor whose local key is most similar with  $K$ . Anyway,  $P$  will finally get similar users and the corresponding vote vectors for key  $K$ .
- 4) For each rated item,  $P$  repeats step 2 and 3. Then  $P$  outputs all the received similar users' vote vectors.

Fig. 5. DHT-based CF lookup function.

### 5. Experimental evaluation

In this section, we describe the dataset, metrics and methodology for the comparison between traditional and DHT-based CF algorithm, and present the results of our experiments.

#### 5.1. Data set

We use [Eachmovie collaborative filtering data set \(1997\)](#) to evaluate the performance of improved algorithm. The EachMovie data set is provided by the Compaq System Research Center, which ran the EachMovie recommendation service for 18 months to experiment with a collaborative filtering algorithm. The information they gathered during that period consists of 72,916 users, 1628 movies, and 2,811,983 numeric ratings ranging from 0 to 5. To speed up our experiments, we only use a subset of the EachMovie data set.

#### 5.2. Metrics and methodology

The metrics for evaluating the accuracy of a prediction algorithm can be divided into two main categories: statistical accuracy metrics and decision-support metrics. Statistical accuracy metrics evaluate the accuracy of a predictor by comparing predicted values with user-provided values. Decision-support accuracy measures how well predictions help user select high-quality items. We use Mean Absolute Error (MAE), a statistical accuracy metrics, to report prediction experiments for it is most commonly used and easy to understand:

$$MAE = \frac{\sum_{a \in T} |v_{a,j} - p_{a,j}|}{|T|} \quad (6)$$

where  $v_{a,j}$  is the rating given to item  $j$  by user  $a$ ,  $p_{a,j}$  is the predicted value of user  $a$  on item  $j$ ,  $T$  is the test set,  $|T|$  is the size of the test set.

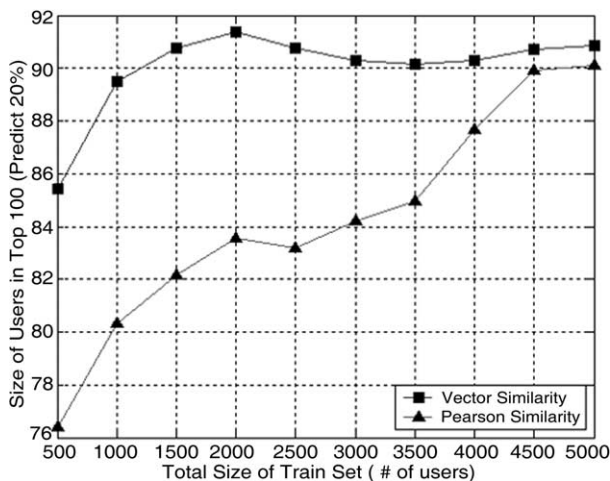


Fig. 6. How many users of PipeCF in traditional CF's top 100.

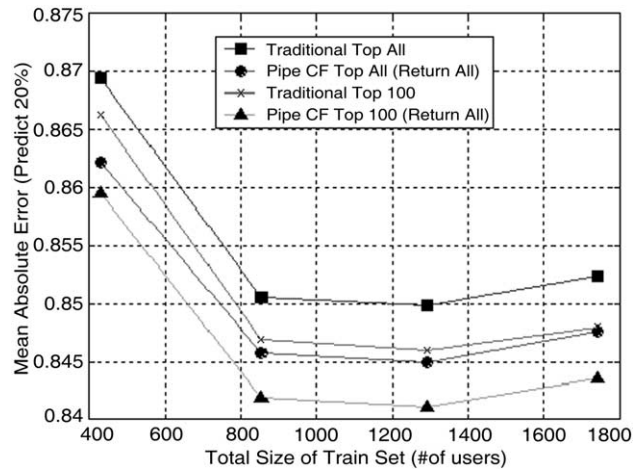


Fig. 7. PipeCF vs. traditional CF.

We select 2000 users and choose one user as active user per time and the remainder users as his candidate neighbors, because every user only make self's recommendation locally. We use the mean prediction accuracy of all the 2000 users as the system's prediction accuracy. For every user's recommendation calculation, our tests are performed using 80% of the user's ratings for training, with the remainder for testing.

#### 5.3. Experimental result

We design several experiments for evaluating our algorithm and analyze the effect of various factors (e.g. SR and UA, etc.) by comparison. All our experiments are run on a Windows 2000-based PC with Intel Pentium 4 processor having a speed of 1.8 GHz and 512 MB of RAM.

##### 5.3.1. The efficiency of neighbor choosing

We used a data set of 5000 users and show among the users chosen by PipeCF algorithm, how many are in the top-100 users in Fig. 6. We can see from the data that when

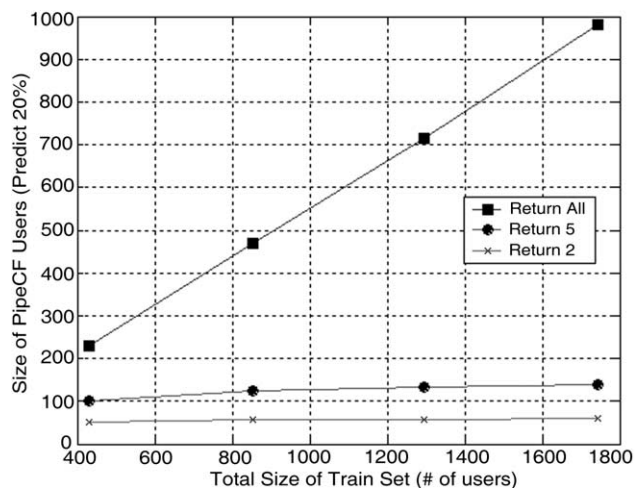


Fig. 8. The effect on scalability of SR on PipeCF.

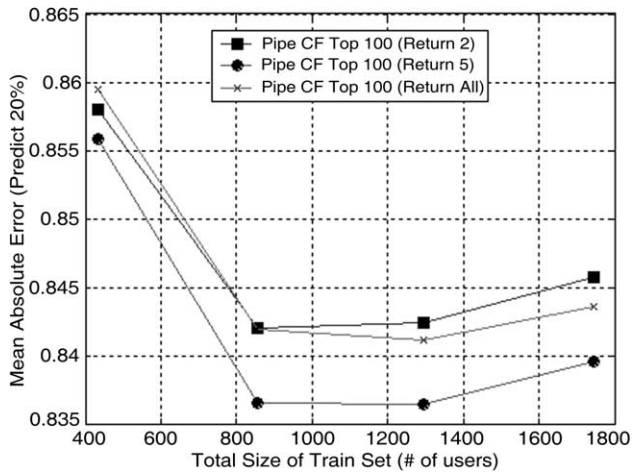


Fig. 9. The effect on prediction accuracy of SR on PipeCF algorithm.

the user number rises above 1000, more than 80 users who have the most similarities with the active users are chosen by PipeCF algorithm.

### 5.3.2. Performance comparison

We compare the prediction accuracy of traditional CF algorithm and PipeCF algorithm while we apply both top-all and top-100 user selection on them. The results are shown as Fig. 7. We can see that the DHT-based algorithm has better prediction accuracy than the traditional CF algorithm.

### 5.3.3. The effect of significance refinement

We limit the number of returned user for each bucket by 2 and 5 and do the experiment in Section 5.3.2 again. The user for each bucket is chosen randomly. The result of the number of user chosen and the prediction accuracy is shown in Figs. 8 and 9, respectively. The result shows:

- (1) 'Return All' has an  $O(N)$  returned user number and its prediction accuracy is also not satisfying;
- (2) 'Return 2' has the least returned user number but the worst prediction accuracy;
- (1) 'Return 5' has the best prediction accuracy and the scalability is still reasonably well (the returned user number is still limited to a constant as the total user number increases).

### 5.3.4. The effect of unanimous amplification

We adjust the weights for each user by using Eq. (5) while setting value for  $\alpha$  as 2.0,  $\beta$  as 4.0,  $\gamma$  as 4 and do the experiment in Section 4.3.2 again. We use the top-100 and 'Return All' selection method. The result shows that the UA approach improves the prediction accuracy of both the traditional and the PipeCF algorithm. From Fig. 10 we can see that when UA approach is applied, the two kinds of algorithms have almost the same performance.

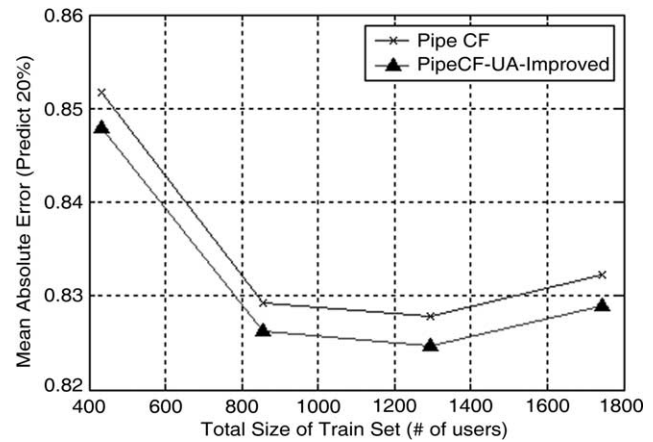


Fig. 10. The effect on prediction accuracy of unanimous amplification.

## 6. Conclusion and future work

In this article, we propose a novel distributed hash table (DHT) based technique to implement efficient user database management and retrieval in decentralized CF system. Then we propose a heuristic algorithm to fetch similar users from DHT overlay network and do recommendation locally. Finally, we propose two novel approaches: SR and UA to improve the performance of our DHT-based CF algorithm. The experimental data show that our DHT-based CF system has better prediction accuracy, efficiency and scalability than traditional CF systems.

Our future work includes investigation on a more efficient decentralized user database management and K-Nearest Neighbor (KNN) methods which can dynamically self-organize users with semantic similar interests combining content-based filtering techniques. We would also like to investigate on the influence of parameters choosing in UA.

## Acknowledgements

The work described in this article is supported partially by National Natural Science Foundation of China under Grant No. 60372078.

## References

- Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 43–52.
- Canny, J. (2002). *Collaborative filtering with privacy*. In: *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press, pp. 45–57.
- Eachmovie collaborative filtering data set, (1997). <http://research.compaq.com/SRC/eachmovie>.

- Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12), 61–70.
- Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 230–237.
- Olsson, T. (2003). *Bootstrapping and Decentralizing Recommender Systems*. Licentiate Thesis 2003-006. Department of Information Technology, Uppsala University and SICS.
- Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content-addressable network. In: *SIGCOMM*.
- Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). *GroupLens: an open architecture for collaborative filtering of netnews*. In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work, October 22–26, 1994, Chapel Hill, North Carolina, United States*, pp. 175–186.
- Rowstron, A., & Druschel, P. (2001). *Pastry: Scalable, distributed object location and routing for large scale peer-to-peer systems*. In: *IFIP/ACM Middleware*, Germany: Hedelberg.
- Shardanand, U., & Maes, P. (1995). *Social information filtering: algorithms for automating 'word of mouth'*. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, May 07–11, 1995, Denver, Colorado, United States*, pp. 210–217.
- Stocal, I., et al. (2001). *Chord: a scalable peer-to-peer lookup service for Internet applications*. In: *ACM SIGCOMM, San Diego, CA, USA*, pp. 149–160.
- Tveit, A. (2001). *Peer-to-peer based recommendations for mobile commerce*. In: *Proceedings of the First International Mobile Commerce Workshop, Rome, Italy*: ACM Press, pp. 26–29.
- Zhao, B. Y. et al (2001). *Tapestry: an infrastructure for fault-tolerant wide-area location and routing*. Tech. Rep. UCB/CSB-0-114, UC Berkeley, EECS.