Improving Accuracy of Recommender System by Clustering Items Based on Stability of User Similarity

Truong Khanh Quan[(1)], Ishikawa Fuyuki[(1)], Honiden Shinichi[(1, 2)]
(1): *The University of Tokyo*
(2): *National Institute of Informatics*
E-mail: quan@nii.ac.jp

Abstract

Collaborative Filtering, one of the most widely used approach in Recommender System, predicts a user's rating towards an item by aggregating ratings given by users having similar preference to that user. In existing approaches, user similarity is often computed on the whole set of items. However, because the number of items is often very large, and so is the diversity among items, users who have similar preference in one category of items may have totally different judgement on items of another kind. In order to deal with this problem, we propose a method of clustering items, so that inside a cluster, similarity between users does not change significantly. After that, when predicting rating of a user towards an item, we only aggregate ratings of users who have high similarity degree with that user inside the cluster to which that item belongs. Experiments evaluating our approach are carried out on the real dataset taken from movies recommendation system of MovieLens web site. Preliminary results suggest that our approach can improve prediction accuracy compared to existing approaches.

1. Introduction

One of the major problems of the E-commerce is that users are often confused when deciding what to choose among abundance of commodities. In order to help users to choose the most suitable item according to their preference, a lot of web sites have implemented Recommender System (RS) to suggest items that are most likely to be preferred by each specific user.

With the explosive growth of E-commerce sites, RS has attracted a lot of interest from researchers. A great deal of algorithms and corresponding extensions have been proposed in the last ten years. However, there are still a lot of problems with existing approaches, including improving accuracy in predicting user's rating, the cold start problem, scalability problem…

In RS applications, one of the most often used approaches is Collaborative Filtering (CF). In CF method, the most important process is computing similarity between users. Existing approaches usually compute users' similarity on the whole set of items. We argue that this does not necessarily reflect the appropriate similarity between users. In various domains, such as books or movies or electronic devices, the number of items is very large. Since there are a lot of items, the diversity between items also becomes higher, and users who have similar preference in one category of items may turn out to have totally different judgement when it comes to another kind of items.

In order to deal with this problem, we propose an approach of item clustering to improve accuracy in user's rating prediction. We divide items into smaller groups, so that inside a group similarity between users does not change significantly, and then apply existing CF methods to compute prediction of users' ratings in each group.

To be more specific, when predicting rating of a user towards an item, we only aggregate ratings of users who have high similarity degree with that user inside the cluster to which that item belongs. In this case, since the preference similarity of users does not change a lot inside an item cluster, users who have similar evaluation towards other items in that cluster will tend to have similar evaluation towards the predicting item too. Consequently, if we aggregate evaluation of similar users inside that cluster, it is expected that we can predict the rating of item more accurately.

Traditional approaches, which cluster items based on the *"distance"* between items, try to minimize distance between items inside a group and maximize distance between items in different groups. Distance between items are often computed by comparing item attributes or computing Pearson correlation between items' rating vectors. Our approach, on the other hand, clusters items by trying to minimize the variance of

International Conference on Computational Intelligence for Modelling Control and Automation,and International Conference on Intelligent Agents,Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)
0-7695-2731-0/06 $20.00 © 2006 IEEE

user similarity inside a group. Specifically, inside a group user similarity does not change significantly from item to item, on the other hand, user similarity values computed on different groups of items will be remarkably different from each other.

The remainder of our paper is organized as follows: Section 2 gives a brief review about CF. Section 3 describes existing item clustering approaches, especially the approach of clustering items based on genre, and discuss problems of these approaches. In section 4, we present our approach of item clustering based on stability of user similarity. Section 5 discusses about our approach in detail. In section 6 we describe our experiments and results. Related work is presented in section 7, and we finally summarize this paper and describe some future work in section 8.

2. Collaborative Filtering

The most famous and widely used approach in RS is CF. CF approaches predicts a user's (called *active user*) rating towards an item by aggregating ratings given to that item by other users who have similar preference to the active user. CF algorithms are often divided into two classes: Model-based and Memory-based algorithm [4].

In Model-based algorithms, users' ratings are used to estimate or learn a model, which is then used for prediction of unevaluated ratings. Some of Model-based algorithms include probabilistic model, user cluster model, Bayesian network…

In Memory-based algorithms, users' ratings are stored in the memory and used directly to compute prediction for unevaluated items. The most widely used Memory-based algorithm is K-Nearest-Neighbor (KNN), which can be described as follows:

1. Compute similarity between active user (user for whom we are going to compute prediction) and all other users
2. Select K nearest neighbors (K users who have highest similarity with the active user)
3. Predict evaluation of active user towards an item *i* by aggregating evaluations given to *i* by K nearest neighbors

The process that most affects the efficiency of KNN algorithm is users' similarity computation. In KNN algorithm, usually the correlation between users' rating vectors (vector that takes as element the ratings given to items by a user) is used as the similarity between users. The two methods often used to compute correlation between two rating vectors are Vector Cosine method and Pearson Correlation method [4] .

Vector Cosine method computes user similarity as the scalar product of rating vectors:

$$s(a,u)=\frac{\sum_{i\in R(a,u)} r_{a,i}\times r_{u,i}}{\sqrt{\sum_{i\in R(a,u)} r_{a,i}^2}\times\sqrt{\sum_{i\in R(a,u)} r_{u,i}^2}} \quad (1)$$

in which, *s(a,u)* is the similarity degree between user *a* and user *u*, *R(a,u)* is the set of items rated by both user *a* and user *u*, $r_{x,i}$ is rating that user *x* gives to items *i*

Pearson Correlation method is similar to Vector Cosine method, but before the scalar product between two vectors is computed, ratings are normalized as the difference between real ratings and average rating of user:

$$s(a,u)=\frac{\sum_{i\in R(a,u)} (r_{a,i}-\bar{r}_a)\times(r_{u,i}-\bar{r}_u)}{\sqrt{\sum_{i\in R(a,u)} (r_{a,i}-\bar{r}_a)^2}\times\sqrt{\sum_{i\in R(a,u)} (r_{u,i}-\bar{r}_u)^2}} \quad (2)$$

in which, $\bar{r}_x$ is the average rating of user *x*

3. Item clustering

Most CF algorithms compute user similarity on the whole set of items. We argue that this is too coarse, since when the number of items is large, the diversity among items becomes higher, and people that have the same opinion on items of one kind may not agree anymore when it comes to a different kind of items.

In this chapter we give examples on the movies recommendation domain, and focus discussion on movies. However, it can be easily realized that the same story also occurs in other domains, such as book recommendation, electronic goods recommendation…

Suppose Alice and Bob have very close preference on animations, then when asked about a new animation, it is highly probable that they will agree again. However, if we ask Alice and Bob for evaluation about a War movie, it is less likely that their opinion will be the same again, since animations and War movies are very different in their characteristics, therefore people having the same evaluation on animations may not agree anymore when it comes to a War movie.

Therefore, we believe that higher prediction accuracy can be achieved if we divide items into smaller groups, and apply the CF algorithm to each group separately (i.e. when predicting evaluation for an item *i*, we only use user similarity computed on the group to which item *i* belongs).

The problem is how to divide items into groups. It can be easily realized that random clustering of items

International Conference on Computational Intelligence for Modelling Control and Automation,and International Conference on Intelligent Agents,Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)
0-7695-2731-0/06 $20.00 © 2006 IEEE

will not help increase accuracy. Then what method should be used to divide items?

A simple solution is to divide items based on item's metadata. For example we can divide movies into different groups according to the genre that movies belong to (e.g. Action movies, War movies…), the date of production, director, main actors…The Internet Movie DataBase (imdb) [5] divides movies in their database into 19 genres: *Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, IMAX, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western*.

Because metadata often have some semantics, when divided into groups based on metadata, items belonging to the same group will have some common characteristics. For example when we divide movies according to genre, we can expect that movies in the same group will have something common in general content. Therefore, user similarity computed only on movies in a group will be more suitable to be used in finding nearest neighbors to ask for opinion about a movie in that group. (Note that when dividing items based on metadata, an item may belong to more than one group, but this should cause no problem because we can compute prediction for that item in each group separately, and then aggregate the results) .

To make things clearer, let's consider the following situation: suppose we want to predict Alice's rating towards movie "Schindler's List". We compute the prediction for Alice's rating by aggregating ratings given to "Schindler's List" by neighbors with highest similarity to Alice in the War movies group.

We believe that when predicting evaluation of "Schindler's List", user similarity computed on War movies only will be more suitable to use than user similarity computed on the whole set of movies, since user preference may change significantly from genre to genre. For example, Alice and Bob may have very similar judgement on War movies, but when it comes to animation or comedy movies, their evaluation may become significantly different from each other. If we compute similarity on the whole set of movies, Bob may not have high similarity with Alice, and his opinion is not considered when predicting Alice's evaluation towards "Schindler's List". This is certainly not a preferable situation, since Bob actually has similar taste with Alice about War movies, and hopefully "Schindler's List" as well, therefore his opinion should be highly regarded.

However, dividing movies according to genre (and other metadata as well) suffers from some major drawbacks. Simply dividing movies according to genre may still appear to be too coarse in some cases. For example inside the War movies groups, Alice and Bob may have the same opinion on movies about WW2, but when it comes to movies about ancient Chinese war, their taste may turn out to be quite different. Besides, there may be hidden characteristics that can not be described by metadata, but can help increase accuracy of prediction if we group movies based on these characteristics.

Another critical disadvantage of using metadata is that in some domains, there are no available metadata at all. A contradictory problem also occurs when there are so many kinds of metadata that it is not easy to choose which one should be used. For example, should we divide movies according to genre, or their date of production, or their director, main actors, or the length of movies?

Furthermore, even if metadata are available, we may never know whether the metadata are added to items in a rational way or not. Metadata are often created by human, and thus suffer from inevitable error and the subjectivity of the metadata creator. For instance, deciding whether a movie belongs to "Action movies" group or not is absolutely a subjective problem, and the decision may differ from person to person.

In order to solve the above problems, we propose an approach of item clustering based on stability of user similarity (definition of stability of user similarity will be given in the next section). Our approach uses only users' ratings to cluster items, therefore the problems with metadata naturally disappear. More importantly, our approach can have the same, if not better, effect in improving prediction accuracy compared to clustering items based on genre. We present our approach in detail in the next chapter.

4. Approach: Item clustering based on stability of user similarity

4.1. Starting ideas

In previous section we argue that when dividing items into groups according to genre, and compute user-to-user similarity in each genre separately, we can improve the accuracy of prediction. We also mentioned the problems and disadvantages of dividing items based on genre. Now a natural question arises: *"Can we think of a method that divides items into groups in a way that help increase prediction accuracy, without using metadata about genre?"*

In order to answer that question, we must know the reason why dividing items based on genre can make prediction more accurate.

We suggest that the answer is when evaluating movies in the same genre, user similarity does not change remarkably. On the other hand, we argue that

user similarity does change significantly across movies of different genres.

To illustrate the above assumption, let's consider the following example. Suppose we have a group containing some War movies, and Alice and Bob have the same opinion on movies belonging to that group. Then if we ask them to evaluate "Schindler's List", we can expect the probability that they will agree again is reasonably high. Consequently, if we add "Schindler's List" to the above group, the similarity between Alice and Bob computed on that group will not change a lot. Now we take "Snow White and the Seven Dwarfs" and let Alice and Bob evaluate it. This time we can expect the probability that their opinion will coincide again is lower than the previous case. As a result, if we add "Snow White and the Seven Dwarfs" to the above group, the similarity between Alice and Bob computed on that group will be likely to change significantly.

At this point we have realized one thing: user similarity does not change significantly from one movie to another inside a group of movies of the same genre. On the other hand, user similarity will change more significantly when computed on movies of different genres. The question is why this could help improve prediction accuracy? The answer is illustrated in the following example.

Suppose there is a group of movies in which user similarity does not change a lot when computed on different movies inside that group. Assume that Alice and Bob have high similarity on a sub set of movies in that group. Now we want to predict Alice's rating on a movie in that group. Because similarity of users inside that group does not change a lot from item to item, it is expected that Alice and Bob also have similar opinion on the item we want to predict, and therefore if we predict Alice's rating based on Bob's rating, it is highly probable that the prediction value is close to the actual rating Alice would give to that movie.

Now we have the answer for the question why dividing movies based on genre can make prediction more accurate. It is because user similarity computed on different items of the same group will not change a lot. Users having high similarity on a sub set of movies in that group will still have similar opinion on other movies in the group, and therefore if we use opinions of users who have high similarity to the active user, we can expect that the prediction value will be close to the actual rating of active user.

However, clustering movies based on genre has only limited and uncertain effect on reducing the change of user similarity inside a group. For example, as we discussed in the previous section, inside the War movies groups, Alice and Bob may have the same opinion on movies about WW2, but when it comes to movies about ancient Chinese war, their taste may turn out to be quite different.

In order to deal with this limitation, we propose a method of clustering items by directly minimizing the variance of user similarity inside a cluster. Right below we will describe our method.

4.2. Item clustering algorithm

In the previous sub section, we talk about the phenomenon of "user similarity does not change significantly when computed on different items inside a group". We call this phenomenon *"stability of user similarity inside a group"*. Right below we give more specific definition of "stability of user similarity".

Definition: *inside a group of items, if user similarity values computed on arbitrary sub item groups are not very different (the difference is smaller than a predefined threshold) from each other, and from the user similarity computed on the whole group, then we say that variance of user similarity inside that group is small, or user similarity is stable inside that group.*

In case there are several users, we just take the average of variance of each user-user similarity values. If the average variance is small, we say that user similarity is stable inside that group.

In the previous sub section we mention that if variance of user similarity inside a group is small, rating prediction will be more accurate. Therefore we come to the conclusion that if we cluster items in a way that maximizes the stability (i.e. minimizes the variance) of user similarity inside each group, then we will have item groups that enable higher prediction accuracy when applying CF algorithms to each group separately. Hereafter we will present our proposed item clustering algorithm that satisfies the above condition:

Input: user-item rating table, number of groups *K*.

Output: Groups of items. Variance of user similarity inside each group is small.

Definition of optimal grouping: in each group, try removing items, one after another according to a predefined order. After each removal of item, recompute user similarity. Finally compute the variance of user similarity during this process of item removal. The optimal grouping is the grouping with smallest average user similarity variance among all groups.

Algorithm:

1. *Item sorting*: Arranges items into list based on the decreasing order of number of rating users
2. *Group initializing*: Assigns top 10% items that are rated by most users to each group consecutively. Initially each group will have approximately the same number of items

3. Compute user-user similarity in each group
4. For each unassigned item *i*:
   4.1. Try assigning *i* to all groups, in each group compute average user similarity change
   4.2. Actually assign *i* to the group with smallest average user similarity change
   4.3 Update user similarity in that group
5. Start again from the beginning of item list, for each item *i*:
   5.1. Try removing *i* from the current group, compute average user similarity change in that group
   5.2. Try assigning *i* to all other groups, in each group compute average user similarity change
   5.3. Move *i* to the group with smallest average user similarity change. If the current group has smallest user similarity change, item will remain in that group. Go to step 6.
   5.4. Update user similarity in previous and new group of item *i*
6. Repeat step 5 until no item moves from one group to another in a loop

5. Discussion

In this section we discuss the time complexity, possible extensions and remaining problems of the proposed algorithm. First of all we would like to introduce some notations:

L: number of loops ; *K*: number of groups

M: number of users ; *N*: number of items

m: average number of users who rated an item (*m*=*R*/*N*, which *R* is total number of ratings)

The step with highest computation cost in our algorithm is step 5: in each loop, we scan through the list of items, and for each item, we try assigning that item to all groups. Consequently the time complexity is $O(LNK \cdot C)$, in which *C* is the cost of computing user similarity change in a group.

In the worst case $C = O(NM^2)$, so the total complexity is: $O(LNK \cdot NM^2) = O(LKN^2M^2)$.

However in reality, the average computation cost for computing user similarity change in a group can be reduced to $O(mM \cdot N/K)$ if we compute user similarity by Pearson Correlation method, or only $O(m^2 \cdot N/K)$ if we use Vector Cosine method. This is because when we add an item to a group, we only have to recompute user similarity between users who rated that item (which is *m* in average), and in addition, the average number of items in a group is only *N*/*K*. Consequently, the average complexity of our algorithm can be reduced to as low as $O(LKN \cdot m^2N/K) = O(LN^2m^2)$ (if we compute user similarity by Vector Cosine method).

Furthermore, if we store old values of user similarity and update user similarity incrementally whenever an item is added, the complexity for recomputing user similarity in a group can be reduced to $O(m^2)$ [2]. Consequently, the total complexity is reduced to $O(LKNm^2)$. (Note that in the case of incremental computation of user similarity, more memory must be used to store old values of similarity).

Generally, $K \ll N$ and $m \ll M$, and in addition, the process of item clustering can be done offline, therefore the above complexity is absolutely feasible.

In step1, "*Item sorting*", and step 2 ,"*Group initializing*", we just give heuristic methods that are not necessarily optimal. We suppose that the optimal methods for item sorting and group initializing is domain-dependent, and to find out which method is best for which domain is one of our research directions in the future.

Another feature needs considering is the algorithm's finish condition. In our algorithm, the finish condition is : "there is no replacement of item in a loop", which is too strong. With this condition, the transition of group states may fall into an infinite loop. In our experiments, whenever the state of groups fall into an infinite loop, we stop the algorithm and output the clustering at that point. It can be realized that this is an adhoc solution, and may affect the quality of the output clustering. However, the ultimate purpose of our research is to improve prediction accuracy in RS, therefore, we are working on considering other finish conditions that assure execution termination of our algorithm, while still having the effect of improving prediction accuracy.

Our approach also suffers from some inherent limitations of clustering algorithms, namely:

1. The final clustering is dependent on the order of items to be processed in a loop
2. The final clustering is dependent on the initial grouping
3. The final clustering may be locally optimal
4. The number of groups *K* must be given

Methods used to overcome these problems in general clustering algorithms, such as running the algorithm several times with several parameter settings and choose the best clustering achieved, can also be applied to our algorithm.

To evaluate the effect of our approach, we are conducting experiments on real data set taken from MovieLens [3], a movies recommendation web site. The preliminary results of our experiments are described in the next section.

International Conference on Computational Intelligence for Modelling
Control and Automation,and International Conference on
Intelligent Agents,Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)
0-7695-2731-0/06 $20.00 © 2006 IEEE

6. Evaluation

We carry out experiments with real data set taken from MovieLens web site. The original data set contains 100,000 real ratings given by 943 users to 1682 movies. Ratings are discrete and in the range of [1, 5].

In order to have a closer look on each user, we decide to limit the number of users to 100 in the preliminary phase of experiment. Therefore, we randomly select 100 users from the original list of 943 users, and carry out experiments on ratings given by these users only. The selection of users is carried out in a uniformly random manner.

The selected 100 users give a total of 11,019 ratings to 1238 movies. Consequently, we have a smaller data set consisting of 11,019 ratings given by 100 users to 1238 movies.

Firstly we want to confirm the effect of dividing items into groups according to genre. We divide 1238 movies into 19 groups: Unknown, Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, and Western. The information of which movie belongs to which group is based on the taxonomy of imdb. After grouping items, we apply Vector Cosine K-Nearest Neighbor (VC-KNN) algorithm to each group to predict ratings of items in that group.

Secondly, we want to evaluate our approach of dividing items into groups based on stability of user similarity. We first use our item-clustering algorithm (described in section 4.2) to divide movies into 19 groups (to be correlated with the number of groups in the case of dividing movies according to genre). After that, we also apply VC-KNN algorithm to each group to predict ratings in that group.

Finally, the comparison baseline is the traditional approach, which applies VC-KNN algorithm to the whole set of items.

We compare the above three approaches in term of *Mean Absolute Error* (*MAE*). Hereafter is the definition of *MAE*:

First we compute *MAE* for each user as follows:

$$MAE(u) = \frac{\sum_{i \in T(u)} |p_{u,i} - r_{u,i}|}{|T(u)|}$$

which *T(u)* is set of items rated by user *u* in the test set, $p_{u,i}$ is the prediction for user *u* about item *i* , $r_{u,i}$ is the real rating that user *u* actually gave to item *i*.

The final *MAE* is computed as the average of all users' *MAEs*.

We divide all ratings in the data set into 80%-20% training and test set, and use ratings in the training set to predict ratings in the test set. The whole data set is divided five times separately to create five divisions: *u1* - *u5*, each division has non-overlapping test set (i.e. the union of the five test sets will make the original data set), so that we can carry out cross-validation experiment.

We carry out experiment with each of five divisions and take the average of all five experiments as the final result. We compare *MAE* in three cases:

1. Without movie grouping (NoGrouping)
2. Grouping movies into groups according to genre (GroupByGenre)
3. Our approach: grouping movies based on stability of user similarity (GroupByUSimStability)

We change the average number of neighbors to ask for opinion when predicting ratings, and compute the *MAE* of each training-test set division. Finally, we compute the average *MAE* of five experiments and plot them again average number of neighbors. The result is showed in Figure 1. Smaller *MAE* indicates better result.

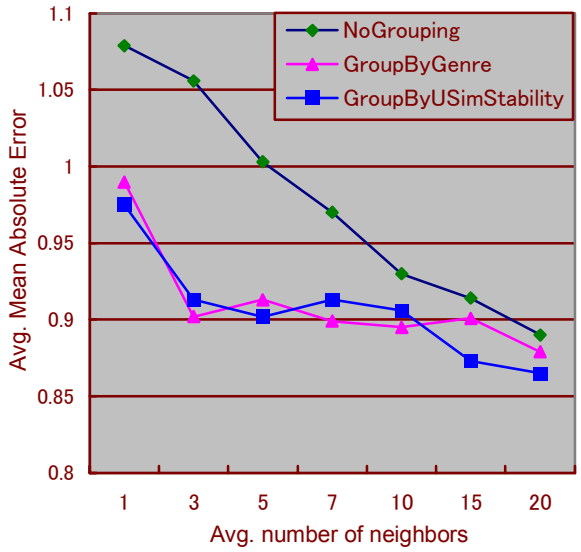

Figure 1: Average *MAE* in three cases: without item grouping, grouping based on genre, and grouping based on stability of user similarity

In Figure 1, as we expected, both two grouping approaches outperform the traditional approach of not grouping items in term of prediction accuracy (i.e. *MAE* is smaller). The difference is especially significant when average number of neighbor is small, with relative improvement in *MAE* between our

approach and NoGrouping approach climes as high as 14% when average number of neighbors is 3.

On the other hand, the two grouping approaches give relatively close results to each other. To be more specific, grouping based on stability of user similarity is a little better, but the superiority is not clear. However, grouping based on stability of user similarity is superior in term of feasibility, since it only takes as input ratings of users, and does not use metadata of genre. As we discussed in section 3, metadata are not always available, and even if available, often suffer from the problem of subjectivity of the metadata creator.

When the number of neighbors increases, the difference between grouping and non-grouping approaches decreases gradually, and finally stay at around 3% when average number of neighbor is 20. However, in real applications, we can only consider opinions of a few nearest neighbors due to the scalability problem, and since our approach yield significantly more accurate prediction than traditional approach when number of neighbor is small, our approach can efficiently contribute to improving the accuracy of rating prediction in real applications.

We are carrying out more experiments to evaluate our algorithms. We plan to run our algorithms with several initial conditions to see how initial settings affect the quality of the final clustering. In particular, we try changing the number of groups K, the order of items to be processed in a loop, the initialization of groups, and try to figure out how these parameters can affect the shape of the final clustering. After clustering items, we apply CF methods to each item group and try to find out to what extent our approach can help improving accuracy of prediction in various conditions. We hope to present more results of our experiment in the nearest future.

7. Related work

Researches on applying data clustering methods to RS can be divided by two dimensions:

- Clustered object: cluster user or item
- Cluster structure: create flat cluster or hierarchical cluster

In user clustering approaches, users are divided into groups based on some common characteristics, such as similarity of rating patterns or on similar demographic data. After that, prediction of rating of a user to an item is computed based on opinions of other users in the same group.

Most researches on user clustering aim at reducing computation cost of RS, usually at the cost of sacrificing prediction accuracy. Some researches on user clustering can be found at [7, 11].

Among item clustering approaches, the work that is nearest to our approach is [1], where authors try to cluster items based on the similarity between items. In their work, similarity between two items are computed as the Pearson correlation between items' rating vectors (vector that takes as element ratings given by users to an item).

Some researches [7, 8] also cluster users and items at the same time, utilizing the symmetry of user and item in the database. However, as far as we know none of them can claim a clear-cut superiority over traditional approach of not clustering.

Our approach can be considered as flat item clustering. In data clustering community, generally the process of grouping data points is carried out based on the "distance" between data points [9, 10, 12]. Similarly, existing item clustering approaches usually define some metrics to measure distance between items, and then divide items into groups based on these metrics.

Our approach is totally different from existing approaches, in the sense that we do not use item-to-item distance. Instead, we use item-to-item-group distance. The distance between an item i and an item group G is defined as follows:

"The distance between item i and group G is defined to be proportional to the average change of user similarity in group G when added with item i."

(i.e. if user similarity does not change much, we say the distance between item i and group G is short, and vice versa).

To put it another way, we do not cluster item based directly on distance between items, but indirectly through the stability of user similarity.

8. Summary

In this paper we present a novel approach of item clustering to improve prediction accuracy in RS. We divide items into smaller groups, and then apply existing CF algorithms to each group separately. Our algorithm takes as input ratings of users, and cluster items into groups by trying to minimize the variance of user similarity inside a cluster. We carry out experiments to evaluate the effect of our approach on the real data set taken from MovieLens web site. Experiment results suggest that our approach outperforms the traditional approach of not clustering in term of accuracy in prediction of ratings.

Our future work includes replacing the current heuristic methods of items sorting and group initializing with more effective methods. This will help

creating item clusters with higher quality (i.e. higher stability of user similarity), and consequently higher accuracy in prediction of unevaluated ratings.

9. References

[1] Mark O'Connor, and Jon Herlocker, "Clustering Items for Collaborative Filtering", *in ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation,* 1999.

[2] Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, and Elas Theoharopoulos, "Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms", *ISMIS '05*, 2005, pp. 553-561.

[3] http://movielens.umn.edu

[4] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference,* Morgan Kaufman, 1998, pp. 43-52.

[5] http://www.imdb.com

[6] J. Herlocker, J. Konstan, L. Terveen, and J. Reidl, "Evaluating collaborative filtering recommender systems.", *ACM Transactions on Information Systems vol. 22*, 2004, pp. 5-53.

[7] L. Ungar, and D. Foster, "Clustering methods for collaborative filtering", *in Proceedings of the workshop on recommendation systems*. Menlo Park, CA. AAAI Press, 1988.

[8] Arnd Kohrs, and Bernard Merialdo, "Clustering for collaborative filtering applications", *in Proceedings of CIMCA'99*. IOS Press, 1999.

[9] A. Jain, M. Murty, and P. Flynn, "Data clustering: A review", *ACM Computing Surveys 31 (3)*, 1999, pp. 264-323.

[10] CC Aggarwal, C. Procopiuc, JL Wolf, PS Yu, and JS Park, "Fast algorithms for projected clustering" , *in Proceedings of ACM SIGMOD Conference on Management of Data*, Philadephia, USA, 1999, pp. 61-72.

[11] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen, "Scalable Collaborative Filtering Using Cluster-based Smoothing", *in Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, Brazil, 2005, pp. 114-121.

[12] AK Jain, and RC Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs NJ, USA, 1988.