IEICE TRANS. INF. & SYST., VOL.E88–D, NO.5 MAY 2005

LETTER

An Improved Neighbor Selection Algorithm in Collaborative Filtering

Taek-Hun KIM[†a)], ***Student Member and*** **Sung-Bong YANG**[†b)], ***Nonmember***

SUMMARY Nowadays, customers spend much time and effort in finding the best suitable goods since more and more information is placed on-line. To save their time and effort in searching the goods they want, a customized recommender system is required. In this paper we present an improved neighbor selection algorithm that exploits a graph approach. The graph approach allows us to exploit the transitivity of similarities. The algorithm searches more efficiently for set of influential customers with respect to a given customer. We compare the proposed recommendation algorithm with other neighbor selection methods. The experimental results show that the proposed algorithm outperforms other methods.

key words: *recommender system, neighbor selection algorithm, collaborative filtering*

1. Introduction

Nowadays, customers spend much time and effort in finding the best suitable goods since more and more information is placed on-line. To save their time and effort in searching the goods they want, a customized recommender system is required. A customized recommender system should predict the most suitable goods for customers by retrieving and analyzing their preferences. A recommender system utilizes in general an information filtering technique called *collaborative filtering*. Collaborative filtering is widely used for such recommender systems as Amazon.com and CDNow.com [1]–[4]. A recommender system using collaborative filtering which we call *CF*, is based on the ratings of the customers who have similar preferences with respect to a given (*test*) customer.

CF calculates the similarity between the test customer and every other customer who has rated the items that are already rated by the test customer. CF uses in general the Pearson correlation coefficient for calculating the similarity. A weak point of the "pure" CF is that it uses all other customers including "useless" customers as the neighbors of the test customer. Since CF is based on the ratings of the neighbors who have similar preferences, it is very important to select the neighbors properly to improve prediction quality.

There have been many investigations on how to select proper neighbors based on neighbor selection methods such as the nearest method and the clustering-based method. They are quite popular techniques for recommender systems. These techniques then predict customer's preferences for the items based on the results of the neighbors' evaluation on the same items [1], [4]–[6], [9].

With millions of customers and items, a recommender system running on an existing algorithm will suffer seriously the scalability problem. Therefore, there are demands for a new approach that can quickly produce high quality predictions and can resolve the very large-scale problem. Clustering techniques often lead to worse accuracy than other methods [7], [9], [12]. Once clustering is done, however, performance can be very good, since the size of a cluster that must be analyzed is much smaller. Therefore the clustering-based method can solve the very large-scale problem in recommender systems.

In this paper we present an improved recommendation algorithm using a graph approach. The proposed algorithm applies the k-means clustering method to the input dataset for finding a handful of promising clusters each of which holds a set of customers with influential similarities for the test customer. It then searches the promising clusters, as if they are graphs, in a breadth-first manner to extract certain customers, called *the neighbors*, who have either higher similarities or lower similarities with respect to the test customer. A cluster is considered a graph in which vertices are customers and each edge has a weight representing the similarity between the end points(customers) of the edge. The graph approach allows us to exploit the transitivity of similarities.

To compare the performance of the clustering-based method with that of our method, the EachMovie dataset of the Compaq Computer Corporation has been used [10]. The EachMovie dataset consists of 2,811,983 preferences for 1,628 movies rated by 72,916 customers explicitly. The experimental results show that the proposed recommendation algorithm provides better prediction quality than other methods.

The rest of this paper is organized as follows. Section 2 describes briefly CF and the clustering-based neighbor selection method. Section 3 illustrates the proposed neighbor selection algorithm in detail. In Sect. 4, the experimental results are presented. The conclusions are given in Sect. 5.

2. Clustering-Based Neighbor Selection in CF

2.1 Collaborative Filtering

CF recommends items through building the profiles of the

Manuscript received June 23, 2004.
Manuscript revised November 3, 2004.

[†]The authors are with the Dept. of Computer Science, Yonsei Univ., Korea.
a) E-mail: kimthun@cs.yonsei.ac.kr
b) E-mail: yang@cs.yonsei.ac.kr

DOI: 10.1093/ietisy/e88–d.5.1072

customers from their preferences for each item. In CF, preferences are represented generally as numeric values which are rated by the customers. Predicting the preference for a certain item that is new to the test customer is based on the ratings of other customers for the 'target' item. Therefore, it is very important to find a set of customers, called *neighbors*, with more similar preferences to the test customer for better prediction quality.

In CF, Eq. (1) is used to predict the preference of a customer. Note that in the following equation $w_{a,k}$ is the Pearson correlation coefficient as in Eq. (2) [2]–[5], [7].

$$P_{a,i} = \overline{r_a} + \frac{\sum_k \{w_{a,k} \times (r_{k,i} - \overline{r_k})\}}{\sum_k \mid w_{a,k} \mid}. \quad (1)$$

$$w_{a,k} = \frac{\sum_j (r_{a,j} - \overline{r_a})(r_{k,j} - \overline{r_k})}{\sqrt{\sum_j (r_{a,j} - \overline{r_a})^2 \sum_j (r_{k,j} - \overline{r_k})^2}}. \quad (2)$$

In the above equations $P_{a,i}$ is the preference of customer a with respect to item i. $\overline{r_a}$ and $\overline{r_k}$ are the averages of customer a's ratings and customer k's ratings, respectively. $r_{k,i}$ and $r_{k,j}$ are customer k's ratings for items i and j, respectively, and $r_{a,j}$ is customer a's rating for item j.

If customers a and k have similar ratings for an item, $w_{a,k} > 0$. We denote $\mid w_{a,k} \mid$ to indicate how much customer a tends to agree with customer k on the items that both customers have already rated. In this case, customer a is a "positive" neighbor with respect to customer k, and vice versa. If they have opposite ratings for an item, then $w_{a,k} < 0$. Similarly, customer a is a "negative" neighbor with respect to customer k, and vice versa. In this case $\mid w_{a,k} \mid$ indicates how much they tend to disagree on the item that both again have already rated. Hence, if they don't correlate each other, then $w_{a,k} = 0$. Note that $w_{a,k}$ can be in between -1 and 1 inclusive.

2.2 The Clustering-Based Neighbor Selection

The k-means clustering method creates k clusters each of which consists of the customers who have similar preferences among themselves. In this method we first select k customers arbitrarily as the initial center points of the k clusters, respectively. Then each customer is assigned to a cluster in such a way that the distance between the customer and the center of a cluster is minimized. The distance is calculated using the Euclidean distance, that is, a square root of the element-wise square of the difference between the customer and each center point.

We then calculate the mean of each cluster based on the customers who currently belong to the cluster. The mean is now considered as the new center of the cluster. After finding the new center of each cluster, we compute the distance between the new center and each customer as before in order to find the cluster to which the customer should belong. Recalculating the means and computing the distances are repeated until a terminating condition is met. The condition is in general how far each new center has moved from the previous center; that is, if all the new centers moved within a certain distance, we terminate the loop.

If the clustering process is terminated, we choose the cluster with the shortest Euclidean distance from its center to the test customer. Finally, prediction for the test customer is calculated with all the customers in the chosen cluster. The clustering-based neighbor selection method can give a recommendation quickly to the customer in case of the very large-scale dataset, because it selects customers in the best cluster only as neighbors [9].

3. The Proposed Neighbor Selection Algorithm

We first describe a neighbor selection algorithm(NSA) in [8]. It considers both high and low similarities with respect to the test customer and exploits the transitivity of similarity using a graph approach. We then explain the computational complexity of the NSA.

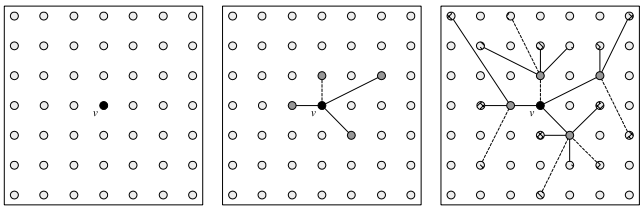
3.1 NSA

The key ideas of NSA are to exploit the transitivity of similarities and to consider both higher and lower similarities in selecting neighbors. We regard a portion of the input dataset a complete undirected graph in which a vertex represents each customer and a weighted edge corresponds to the similarity between two end points(customers) of the edge.

NSA creates k clusters from the input dataset with the k-means clustering method. Then it finds the best cluster C with respect to the test customer u among the k clusters. NSA searches the customers in C to find the 'best' vertex v with the highest similarity with respect to u. NSA then searches the unmarked vertices adjacent to v who have the similarities either larger than H or smaller than L, where H and L are some threshold values for the Pearson correlation coefficients. Note that as the threshold values change, so does the size of the neighbors. The search is performed in a breath-first manner. That is, we search the adjacent vertices of v according to H and L to find the neighbors of u, and then search the adjacent vertices of each neighbor of v in turn. The search stops when we have enough neighbors for prediction.

Figure 1 depicts an example of how NSA selects the neighbors of v who has the higher similarities with respect to u when the search depth is 2. A solid line indicates that the weight on the edge is larger than H. A dotted line means that the weight of the edge is smaller than L.

We now describe NSA in detail. In the algorithm, before we apply the graph approach to the input dataset, we want to remove insignificant customers with the k-means clustering method. After we obtain k clusters with the k-means clustering method, we choose a cluster (the 'best'



(a) Assume that v has the highest similarity for the test customer.

(b) After finding the neighbors of v (the search depth=1).

(c) After selecting the neighbors of each neighbor of v (the search depth=2).

Fig. 1 An example of searching the neighbors in NSA.

cluster) whose mean has the highest similarity with respect to the test customer u. We then apply the graph approach only to the best cluster. The following describes the overall algorithm in detail. When the algorithm is terminated, the set *Neighbors* is returned as output.

The Neighbor Selection Algorithm
Input: the test customer $\boldsymbol{u}$, the input dataset $\boldsymbol{S}$
Output: ***Neighbors***

1. Create k clusters from $\boldsymbol{S}$ with the k-means clustering method;
2. Find the best cluster $\boldsymbol{C}$ for the test customer $\boldsymbol{u}$;
3. Find the best customer $\boldsymbol{v}$ in cluster $\boldsymbol{C}$ who has the highest similarity with respect to $\boldsymbol{u}$;
4. Add $\boldsymbol{v}$ to ***Neighbors***;
5. Traverse $\boldsymbol{C}$ from $\boldsymbol{v}$ in a breadth-first manner when visiting vertices(customers). The similarity of the customer is checked to see if it is either higher than $\boldsymbol{H}$ or lower than $\boldsymbol{L}$. If so, add the customer to ***Neighbors***;

Note that Step5 is executed until we get enough neighbors and such terminating condition can be imposed with how many levels(depths) we search from v in a breadth-first manner.

3.2 The Computational Complexity of the NSA

The k-means clustering algorithm is arguably the most well-known clustering algorithm in use today. Its computational complexity is $O(krn)$, where k is the number of desired clusters, r is the number of iterations, and n is the size of the dataset [11]. In collaborative filtering the complexity for the prediction depends on the size of the neighbors. Therefore, if we consider all the points (customers) in the chosen cluster (the best cluster) as the neighbors, the complexity is only $O(c)$, where c is the size of the chosen cluster.

The complexity of the breadth-first search (BFS) in a complete undirected graph is $O(n^2)$, where n is the size of the vertex set. NSA searches the neighbors with transitivity in a graph using BFS and stops the search if the search depth is greater than a threshold value predetermined. BFS for the neighbors using the transitivity mechanism in a complete undirected graph takes $O(dn)$, where d is the search depth. If we consider only the chosen cluster, the search complexity is $O(dc)$.

Although NSA searches the neighbors using BFS, it searches only 'not selected' vertices, that is, all vertices selected as the neighbors in the previous search steps are considered as the selected vertices. Therefore, the complexity $T(c)$ of NSA can be computed with (3). $T(c)$ is derived from $\sum_d T_d = T_1 + T_2 + \ldots + T_d$, where T_d is the complexity when the search depth is d. $T(c)$ is computed by each step of the following appendix. In (3) and the appendix S_d is the number of the selected vertices when the search depth is d and S_d^i is the number of the selected vertices for each vertex i of S_{d-1}. For each depth if there is only one selected vertex, then $T(c)$ is $O(dc)$, which is the worst case. If all vertices are selected when the search depth is one, then $T(c)$ is $O(c)$, which is the best case.

$$T(c) = \sum_d (S_{d-1}(c - \sum_{i=1}^{d-1} S_i)) - \sum_d (\sum_{i=1}^{S_{d-1}-1} (S_{d-1} - i) S_d^i), \quad (3)$$

$$where \;\; S_0 = 1, \sum_d S_d = c, and \sum_i S_d^i = S_d.$$

4. Experimental Results

4.1 Experiment Environment

In order to evaluate the prediction accuracy of the proposed recommendation algorithm, we used the EachMovie dataset of the Digital Equipment Corporation for experiments [10]. The EachMovie dataset consists of 2,811,983 preferences for 1,628 movies rated by 72,916 customers explicitly. The customer preferences are represented as numeric values from 0 to 1 at an interval of 0.2, i.e., 0.0, 0.2, 0.4, 0.6, 0.8, and 1. In the EachMovie dataset, one of the valuable attributes of an item is the genre of a movie. There are 10 different genres such as action, animation, art-foreign, classic, comedy, drama, family, horror, romance, and thriller.

For the experiment, we retrieved 3,763 customers who rated at least 100 movies among all the customers in the dataset. We have chosen randomly 50 customers as the test customers and the rest customers are the training customers. For each test customer, we chose 5 movies randomly that are actually rated by the test customer as the test movies. The final experimental results are averaged over the results of the test sets.

4.2 Experimental Metrics

One of the statistical prediction accuracy metrics for evaluating recommender systems is the mean absolute error(MAE) which is the mean of the errors of the actual customer ratings against the predicted ratings in individual prediction [1], [5]–[7]. MAE is computed by (4). In the equation, N is the total number of predictions and ε_i is the error

Table 1 Experimental results.

Methods	MAE	Parameters
PCF	0.190479	
kMCF	0.172146	$k = 21$
GCF	0.178906	$H = 0.8, d = 2$
NSA	0.165700	$k = 21, H = 0.7, L = -0.8, d = 2$

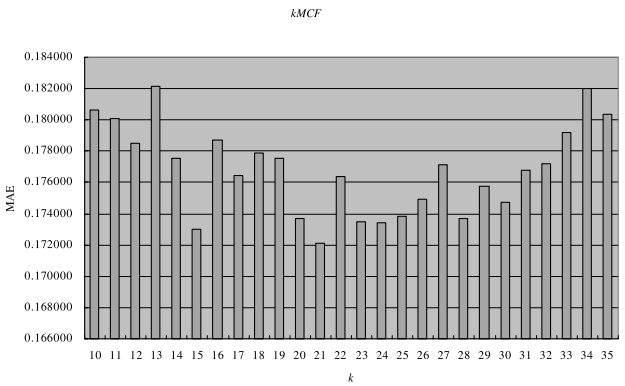


Fig. 2 The sensitivity of the number of clusters k.

between the predicted rating and the actual rating for item i. The lower MAE is, the more accurate prediction with respect to the numerical ratings of customers we get.

$$| E |= \frac{\sum_{i=0}^{N} | \varepsilon_i |}{N}. \quad (4)$$

4.3 Experimental Results

We compare the proposed recommendation algorithm with two other methods. The first method is the pure collaborative filtering method (*PCF*) used by the GroupLens [2], [3]. *PCF* considers all the customers in the input dataset as neighbors without clustering. The second method is a modified *PCF* that utilizes the k-means clustering which we call *kMCF*. For the proposed algorithm we have two different settings. The first one is *NSA*. The other is *NSA* without clustering which we call *GCF*.

The experimental results are shown in Table 1. We determined that k=21 for both *kMCF* and *NSA* through various experiments. That is, this value for k gave us the smallest MAE value among others as shown in Fig. 2. Among the parameters in the table, L and H denote that the threshold values for the Pearson correlation coefficients. For example, if $L = -0.8$ and $H = 0.7$, then we select a neighbor whose similarity is either smaller than -0.8 or larger than 0.7. In the table d is a search depth.

The results in the table show that *NSA* outperforms others when the search depth is 2. We found that when the search depth is grater than 2, *NSA* did not provide better results, because as the size of the neighbors gets larger the chances that unnecessary customers are included in the neighbors get higher. Table 1 also shows that the methods with the graph approach using transitivity of similarities which are *GCF* and *NSA*, find valuable neighbors in both *PCF* and *kMCF*.

5. Conclusions

It is crucial for a recommender system to have the capability of making accurate prediction by retrieving and analyzing customers' preferences. Collaborative filtering is widely used for recommender systems. Hence various efforts to overcome its drawbacks have been made to improve prediction quality.

It is important to select neighbors properly in order to make up the weak points in collaborative filtering and to improve prediction quality. In this paper we proposed an improved neighbor selection algorithm that finds meaningful neighbors using a graph approach along with clustering. The experimental results show the proposed algorithm provides the better prediction quality than other methods.

The proposed method utilizes a clustering technique and transitivity mechanism. Therefore, it can be a choice to solve the very large-scale problem because it reduces the search space using clustering for a large dataset and also produces high prediction quality using the transitivity of similarities. The complexity of the proposed method is not high. However, it may be increased a little more than that of a clustering method in the worst case.

Acknowledgments

We thank the Compaq Equipment Corporation for permitting us to use the EachMovie dataset. This work was supported by the Brain Korea 21 Project in 2004.

References

[1] B.M. Sarwar, G. Karypis, J.A. Konstan, and J.T. Riedle, "Application of dimensionality reduction in recommender system—A case study," Proc. ACM WebKDD 2000 Web Mining for E-Commerce Workshop, pp.82–90, 2000.
[2] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl, "GroupLens: Applying collaborative filtering to usenet news," Commun. ACM, vol.40, pp.77–87, 1997.
[3] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," Proc. ACM CSCW94 Conference on Computer Supported Cooperative Work, pp.175–186, 1994.
[4] B.M. Sarwar, G. Karypis, J.A. Konstan, and J.T. Riedl, "Analysis of recommendation algorithms for E-commerce," Proc. ACM E-commerce 2000 Conference, pp.158–167, 2000.
[5] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," Proc. 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.230–237, 1999.
[6] M. O'Connor and J. Herlocker, "Clustering items for collaborative filtering," Proc. ACM SIGIR Workshop on Recommender Systems, 1999.
[7] J.S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," Proc. Conference on Uncertainty in Artificial Intelligence, pp.43–52, 1998.
[8] T.-H. Kim, Y.-S. Ryu, S.-I. Park, and S.-B. Yang, "An improved recommendation algorithm in collaborative filtering," Lecture Notes

in Computer Science, vol.2455, pp.254–261, 2002.

[9] B.M. Sarwar, G. Karypis, J. A. Konstan, and J.T. Riedle, "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering," Proc. Fifth International Conference on Computer and Information Technology, 2002.

[10] S. Glassman, "EachMovie collaborative filtering data set," Compaq Computer Corporation, url:http://research.compaq.com/SRC/eachmovie/ 1997.

[11] S. Kantabutra, "Cluster computing as a tool in theoretical computer science," Proc. National Workshop on Cluster Coputing 2003, pp.1–16, 2003.

[12] T.-H. Kim and S.-B. Yang, "Using attributes to improve prediction quality in collaborative filtering," Lecture Notes in Computer Science, vol.3182, pp.1–10, 2004.

Appendix: The computational complexity $T(c)$

$$T(c) = \sum_d T_d = T_1 + T_2 + \ldots + T_d.$$

$$S_0 = 1, \sum_d S_d = c, and \sum_i S_d^i = S_d.$$

$$T_1 = c.$$

$$\begin{aligned} T_2 &= \sum_{i=1}^{S_1} T_2^i = T_2^1 + T_2^2 + \ldots + T_2^{S_1} \\ &= S_1(c - S_1) \\ &\quad - (S_1 - 1)S_2^1 - (S_1 - 2)S_2^2 \\ &\quad - \ldots \\ &\quad - (S_1 - (S_1 - 1))S_2^{S_1 - 1}. \end{aligned}$$

$$\begin{aligned} &T_2^1 = c - S_1, \\ &T_2^2 = c - S_1 - S_2^1, \\ &\ldots, \\ &T_2^{S_1} = c - S_1 - S_2^1 - S_2^2 - \ldots - S_2^{S_1 - 1}. \\ &\ldots \end{aligned}$$

$$\begin{aligned} T_d &= \sum_{i=1}^{S_{d-1}} T_d^i = T_1 + T_2 + \ldots + T_d^{S_{d-1}} \\ &= S_{d-1}(c - S_1 - S_2 - \ldots - S_{d-1}) \\ &\quad - (S_{d-1} - 1)S_d^1 - (S_{d-1} - 2)S_d^2 \\ &\quad - \ldots \\ &\quad - (S_{d-1} - (S_{d-1} - 1))S_d^{S_{d-1} - 1}. \end{aligned}$$

$$\begin{aligned} &T_d^1 = c - S_1 - S_2 - \ldots - S_{d-1}, \\ &T_d^2 = c - S_1 - S_2 - \ldots - S_{d-1} - S_d^1, \\ &\ldots \\ &T_d^{S_{d-1}} = c - S_1 - S_2 - \ldots - S_{d-1} \\ &-S_d^1 - S_d^2 - \ldots - S_d^{S_{d-1} - 1}. \end{aligned}$$

$$\begin{aligned} \therefore T(c) &= \sum_d T_d = T_1 + T_2 + \ldots + T_d \\ &= \sum_d (S_{d-1}(c - \sum_{i=1}^{d-1} S_i)) \\ &\quad - \sum_d (\sum_{i=1}^{S_{d-1} - 1} (S_{d-1} - i)S_d^i). \end{aligned}$$