Recommender Systems for Large-scale E-Commerce: Scalable Neighborhood Formation Using Clustering

Badrul M. Sarwar[†,*], George Karypis[‡], Joseph Konstan[†], and John Riedl[†]
{sarwar, karypis, konstan, riedl}@cs.umn.edu
[†]GroupLens Research Group / [‡]Army HPC Research Center
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455, USA

Abstract

Recommender systems apply knowledge discovery techniques to the problem of making personalized product recommendations during a live customer interaction. These systems, especially the k-nearest neighbor collaborative filtering based ones, are achieving widespread success in E-commerce nowadays. The tremendous growth of customers and products in recent years poses some key challenges for recommender systems. These are: producing high quality recommendations and performing many recommendations per second for millions of customers and products. New recommender system technologies are needed that can quickly produce high quality recommendations, even for very large-scale problems. We address the performance issues by scaling up the neighborhood formation process through the use of clustering techniques.

1 Introduction

The largest E-commerce sites offer millions of products for sale. Choosing among so many options is challenging for consumers. Recommender systems have emerged in response to this problem. A recommender system for an E-commerce site recommends products that are likely to fit her needs. Today, recommender systems are deployed on hundreds of different sites, serving millions of consumers. One of the earliest and most successful recommender technologies is *collaborative filtering* [5, 8, 9, 13]. Collaborative filtering (CF) works by building a database of preferences for products by consumers. A new consumer, Neo, is matched against the database to discover *neighbors*, which are other consumers who have historically had similar taste to Neo. Products that the neighbors like are then recommended to Neo, as he will probably also like them. Collaborative filtering has been very successful in both research and practice. However, there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems.

The first challenge is to improve the scalability of the collaborative filtering algorithms. These algorithms are able to search tens of thousands of potential neighbors in real-time, but the demands of modern E-commerce systems are to search tens of millions of potential neighbors. Further, existing algorithms have performance problems with individual consumers for whom the site has large amounts of information. For instance, if a site is using browsing patterns as indications of product preference, it may have thousands of data points for its most valuable customers. These "long customer rows" slow down the number of neighbors that can be searched per second, further reducing scalability. The second challenge is to improve the quality of the recommendations for the consumers. Consumers need recommendations they can trust to help them find products they will like. If a consumer trusts a recommender system, purchases a product, and finds out he does not like the product, the consumer will be unlikely to use the recommender system again. In some ways these two challenges are in conflict, since the less time an algorithm spends searching for neighbors, the more scalable it will be, and the worse its quality. For this reason, it is important to treat the two challenges simultaneously so the solutions discovered are both useful and practical.

The focus of this paper is two-fold. First, we introduce the basic concepts of a collaborative filtering based recommender system and discuss its various limitations. Second, we present a clustering-based algorithm that is suited for a large data set, such as those are common in E-commerce applications of recommender systems. This algorithm has characteristics that make it likely to be faster in online performance than many previously studied algorithms, and we seek to investigate how the quality of its recommendations compares to other algorithms under different practical circumstances.

The rest of the paper is organized as follows. The next section provides a brief overview of collaborative filtering based recommender systems and discusses

*Currently with the Computer Science Department, San Jose State University, San Jose, CA 95112, USA. Email: sarwar@cs.sjsu.edu, Phone: +1 408-245-8202

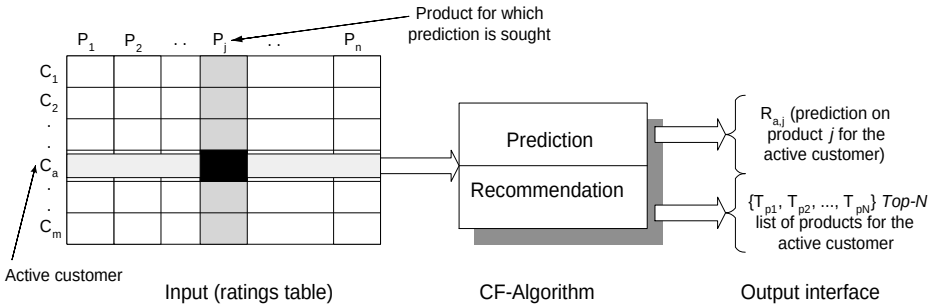

Figure 1: The Collaborative Filtering Process.

some of its limitations. Section 3 describes the details algorithm of applying clustering based approach to address these limitations. Section 4 describes our experimental framework, experimental results, and discussion. The final section provides some concluding remarks and directions for future research.

2 Collaborative Filtering-based Recommender Systems

Collaborative filtering (CF) [8, 9, 13] is the most successful recommender system technology to date, and is used in many of the most successful recommender systems on the Web. CF systems recommend products to a target customer based on the opinions of other customers. These systems employ statistical techniques to find a set of customers known as *neighbors*, that have a history of agreeing with the target user (*i.e.*, they either rate different products similarly or they tend to buy similar set of products). Once a neighborhood of users is formed, these systems use several algorithms to produce recommendations.

In a typical E-Commerce scenario, there is a list of m customers $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ and a list of n products $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. Each customer c_i expresses his/her opinions about a list of products. This set of opinions is called the "ratings" of customer c_i and is denoted by P_{c_i}. There exists a distinguished customer $c_a \in \mathcal{C}$ called the *active customer* for whom the task of a collaborative filtering algorithm is to find a product suggestion.

Most collaborative filtering based recommender systems build a neighborhood of likeminded customers. The Neighborhood formation scheme usually uses Pearson correlation or cosine similarity as a measure of proximity [13]. The neighborhood formation process is in fact the model-building or learning process for a recommender system algorithm. The main goal of neighborhood formation is to find, for each customer C, an ordered list of k customers $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$ such that $C \notin \mathcal{N}$ and $sim(C, N_1)$ is maximum, $sim(C, N_2)$ is the next maximum and so on. Where $sim(C, N_i)$ indicates similarity between two customers, which is most often computed by finding the *Pearson-r* correlation between the customers C and N_i.

Once these systems determine the proximity neighborhood, they produce recommendations that can be of two types:

- *Prediction* is a numerical value, $R_{a,j}$, expressing the predicted opinion-score of product p_j for the active customer c_a. This predicted value is within the same scale (e.g., from 1 to 5) as the opinion values provided by c_a.

- *Recommendation* is a list of N products, $TP_r = \{T_{p1}, T_{p2}, \ldots, T_{pN}\}$, that the active user will like the most. The recommended list usually consists of the products not already purchased by the active customer. This output interface of CF algorithms is also known as *Top-N recommendation*.

Figure 1 shows the schematic diagram of the collaborative filtering process. CF algorithms represent the entire $m \times n$ customer-product data as a ratings matrix, $\mathcal{A}$. Each entry $a_{i,j}$ in $\mathcal{A}$ represent the preference score (ratings) of the ith customer on the jth product. Each individual rating is within a numerical scale and it can as well be 0, indicating that the customer has not yet rated that product.

These systems have been successful in several domains, but the algorithm is reported to have shown some limitations, such as:

- **Sparsity.** Nearest neighbor algorithms rely upon exact matches that cause the algorithms to sacrifice recommender system coverage and accuracy [8, 11]. In particular, since the correlation coefficient is only defined between customers who have rated at least two products in common, many pairs of customers have no correlation at all [1]. Accordingly, Pearson nearest neighbor algorithms may be unable to make many product recommendations for a particular user. This problem is known as reduced coverage, and is due to sparse ratings of neighbors.

- **Scalability.** Nearest neighbor algorithms require computation that grows with both the number of

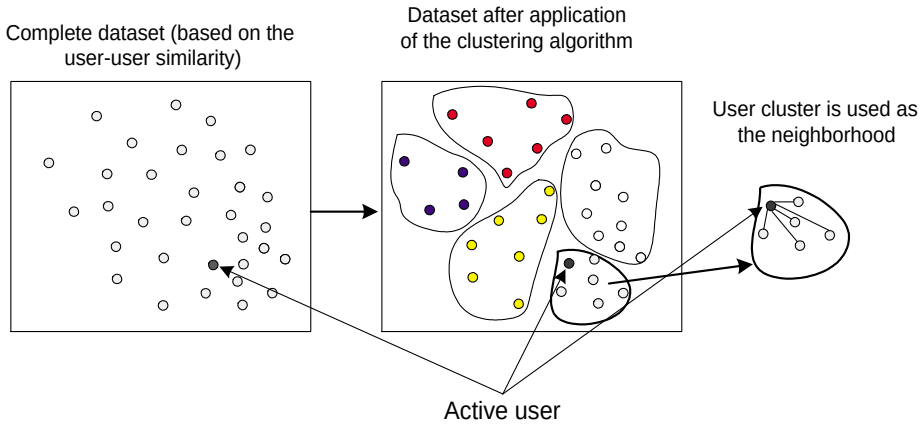

Figure 2: Neighborhood formation from clustered partitions

customers and the number of products. With millions of customers and products, a typical web-based recommender system running existing algorithms will suffer serious scalability problems.

The weakness of *Pearson* nearest neighbor approach for large, sparse databases led us to explore alternative recommender system algorithms. Our first approach attempted to bridge the sparsity by incorporating semi-intelligent filtering agents into the system [11]. We addressed the scalability challenge in an earlier work [12], where we showed that forming neighborhoods in the low dimensional eigen-space provided better quality and performance. Here we present a different dimensionality reduction approach by first clustering the data set and then forming neighborhoods form the partitions. The application of clustering techniques reduces the sparsity and improves scalability of recommender systems. Clustering of users can effectively partition the ratings database and thereby improve the scalability and sparsity. Earlier studies [2, 8, 15] indicate the benefits of applying clustering in recommender systems. We outline our research approach in the next section.

3 Scalable Neighborhood Using Clustering

Clustering techniques work by identifying groups of users who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster. Some clustering techniques represent each user with partial participation in several clusters. The prediction is then an average across the clusters, weighted by degree of participation. Clustering techniques usually produce less-personal recommendations than other methods and most often lead to worse accuracy than nearest neighbor algorithms [3]. Once the clustering is complete, however, performance can be very good, since the size of the group that must be analyzed is much smaller.

3.1 Scalable Neighborhood Algorithm

The idea is to partition the users of a collaborative filtering system using a clustering algorithm and use the partitions as neighborhoods. Figure 2 explains this idea. A collaborative filtering algorithm using this idea first applies a clustering algorithm on the user-item ratings database to divide the database A into p partitions. The clustering algorithm may generate fixed sized partitions, or based on some similarity threshold it may generate a requested number of partitions of varying size. In the next step, the neighborhood for the active customer c_a is selected by looking into the partition where he/she belongs. The entire partition, A_i is then used as the neighborhood for that active customer c_a. Prediction is generated using basic collaborative filtering technique. We now present the algorithm formally.

Algorithm: Clustered neighborhood formation

1. Apply the clustering algorithm to produce p partitions of users using the training data set. Formally, the data set A is partitioned in $A_1, A_2, \ldots, A_p$, where $A_i \cap A_j = \phi$, for $1 \leq i, j \leq p$; and $A_1 \cup A_2 \cup \ldots \cup A_p = A$.

2. Determine the neighborhood for a given user u. If $u \in A_i$ then the entire partition A_i is used as the neighborhood.

3. Once the neighborhood is obtained, classical collaborative filtering algorithm is used to generate prediction from that. In particular, the prediction score $R_{a,j}$ for a customer c_a on product p_j is computed by using the following formula [9].

$$R_{a,j} = \bar{P_{C_a}} + \frac{\sum_i \text{rates } j (P_{C_{i,j}} - \bar{P_{C_i}}) r_{a,i}}{\sum_i |r_{a,i}|} \quad (1)$$

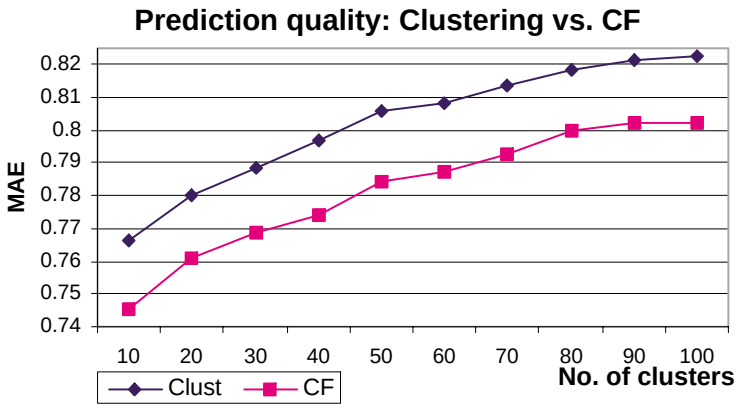

Figure 3: Quality of prediction using clustered-neighborhood vs. classical CF-neighborhood approach

where $r_{a,i}$ denotes the correlation between the active user C_a and its neighbors C_i who have rated the product P_j. $\bar{P_{C_a}}$ denotes the average ratings of customer C_a, and $P_{C_{i,j}}$ denotes the rating given by customer C_i on product P_j.

This method has two benefits—first, it reduces the sparsity of the data set and second, due to the dimensionality reduction and use of a static pre-computed neighborhood, the prediction generation is much faster.

4 Experimental Evaluation

In this section we present a brief discussion of our experimental data set, evaluation metric, and experimental platform followed by the experimental results and discussion.

4.1 Data Sets

We used data from our recommender system MovieLens (`www.movielens.umn.edu`), which is a web-based research recommender system that debuted in Fall 1997. Each week hundreds of users visit MovieLens to rate and receive recommendations for movies. The site now has over $50,000$ users who have expressed opinions on $3,000+$ different movies. We randomly selected enough users to obtain $100,000$ ratings from the database (we only considered users that had rated 20 or more movies). We divided the database into 80% training set and 20% test set. The data set was converted into a user-movie matrix R that had 943 rows(users) and 1682 columns(movies that were rated by at least one of the users).

4.2 Evaluation Metrics

Recommender systems researchers use a number of different measures for evaluating the success of the recommendation or prediction algorithms [13, 11]. For our experiments, we use a widely popular statistical accuracy metric named *Mean Absolute Error (MAE)*, which is a measure of the deviation of recommendations from their true user-specified values. For each ratings-prediction pair $< p_i, q_i >$, this metric treats the absolute error between them i.e., $|p_i - q_i|$ equally. The MAE is computed by first summing these absolute errors of the N corresponding ratings-prediction pairs and then computing the average. Formally, $MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$, The lower the MAE, the more accurately the recommendation engine predicts user ratings.

Our choice of MAE as the primary accuracy metric is due to the fact that it matches the goals of our experiments most closely. MAE is most commonly used and easiest to interpret directly. There is a vast research literature on performing statistical significance testing and computing confidence intervals for MAE. Furthermore, researchers [5] in the related field have also suggested the use of MAE for prediction evaluation metric.

4.3 Experimental Procedure

Benchmark CF system. To compare the performance of item-based prediction we also entered the training ratings set into a collaborative filtering recommendation engine that employs the Pearson nearest neighbor algorithm. We tuned the algorithm to use the best published Pearson nearest neighbor algorithm and configured it to deliver the highest quality prediction without concern for performance (i.e., it considered every possible neighbor to form optimal neighborhoods).

Experimental platform. All our experiments were implemented using C and compiled using optimization flag -06. We ran all our experiments on a Linux based workstation with dual Intel Pentium III processors having a speed of 600 MHz and 2GB of RAM.

Experimental steps. To experimentally evaluate the effectiveness of clustering, we use a variant of K-

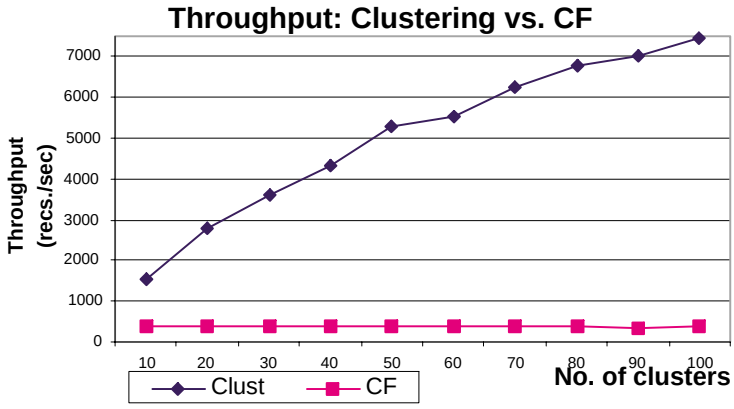

Figure 4: Throughput of clustered-neighborhood vs. classical CF-neighborhood approach.

means [7] clustering algorithm called the *bisecting K-means* clustering algorithm. This algorithm is fast and tends to produce clusters of relatively uniform size, which results in good cluster quality [14]. We divide the movie data set into a 80% training and 20% test portion. For the purpose of comparison, we perform the same experiments using our benchmark CF-based recommender system. We use the same train/test ratio x, and number of neighbors. In case of the clustering approach, the number of neighbors is not always fixed; one cluster may have 30 users, another one may have 55 user and so on. To make our comparisons fair, we recorded the number of neighbors used for prediction computation for each user and forced our basic CF algorithm to use same number of neighbors for prediction generation. We evaluated the results using the MAE metric and also noted the run time elapsed in seconds. We conducted a 10-fold cross validation of our experiments by randomly choosing different training and test sets each time and taking the average of the MAE and run time values.

4.4 Results and Discussion

Figure 3 presents the prediction quality results of our experiments for the clustering as well as basic CF techniques. In this chart, prediction quality is plotted as a function of the number of clusters. We make two important observations from this chart. First, the prediction quality is worse in case of the clustering algorithm but the difference is small. For instance, using 10 clusters, the clustered approach yields an MAE of 0.7665 and the corresponding CF algorithm yields an MAE of 0.7455. It can also be observed from the chart that as we increase the number of clusters the quality tends to be inferior (increased MAE). In case of clustering it is expected as with a fixed number of users, increasing the number of clusters would mean small cluster sizes and hence small number of neighbors to have opinions about items. The same trend is observed in the case of basic CF as we force them to use the same number of neighbors as the clustered approach.

Figure 4 presents the performance results of our experiments for both of the techniques. We plot the throughput as a function of the cluster size. We define throughput of a recommender system as the number of recommendations generated per second. From the plot we see that using the clustered approach, the throughput is substantially higher than the basic CF approach. This is due to the reason that with the clustered approach the prediction algorithm has to use a fraction of neighbors. The throughput increases rapidly with the increase in the number of clusters (small cluster sizes). Since the basic CF method has to scan through all the neighbors, the number of clusters does not impact the throughput.

5 Conclusion and Future Work

Recommender systems are a powerful new technology for extracting additional value for a business from its customer databases. These systems help customers find products they want to buy from a business. Recommender systems benefit customers by enabling them to find products they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Recommender systems are being stressed by the huge volume of customer data in existing corporate databases, and will be stressed even more by the increasing volume of customer data available on the Web. New technologies are needed that can dramatically improve the scalability of recommender systems.

In this paper, we presented and experimentally evaluated a new approach in improving the scalability of recommender systems by using clustering techniques. Our experiments suggest that clustering based neighborhood provides comparable prediction quality as the basic CF approach and at the same time significantly improves the online performance.

We demonstrated the effectiveness of one particu-

lar clustering algorithm (bisecting k-means algorithm). In future, better clustering algorithms as well as better prediction generation schemes can be used to improve the prediction quality. Clustering techniques can also be applied as a "first step" for shrinking the candidate set in a nearest neighbor algorithm or for distributing nearest-neighbor computation across several recommender engines. While dividing the population into clusters may hurt the accuracy or recommendations to users near the fringes of their assigned cluster, pre-clustering may be a worthwhile trade-off between accuracy and throughput.

6 Acknowledgments

Funding for this research was provided in part by the National Science Foundation under grants IIS 9613960, IIS 9734442, and IIS 9978717 with additional funding by Net Perceptions Inc. This work was also supported by NSF CCR-9972519, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. We also thank anonymous reviewers for their valuable comments.

References

[1] Billsus, D., and Pazzani, M. J. (1998). Learning Collaborative Information Filters. In *Proceedings of ICML '98*. pp. 46-53.

[2] Borchers, A., Leppik, D., Konstan, J., and Riedl, J. (1998). Partitioning in Recommender Systems. *Technical Report CS-TR-98-023,* Computer Science Dept., University of Minnesota.

[3] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence,* pp. 43-52.

[4] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*. December.

[5] Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of ACM SIGIR'99*. ACM press.

[6] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of CHI '95*.

[7] Jain, A. K., and Dubes, R. C. (1988). Algorithms for Clustering Data. *Prentice Hall Publishers*. Englewood Cliffs, NJ.

[8] Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J. (1997). GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM,* 40(3), pp. 77-87.

[9] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of CSCW '94*, Chapel Hill, NC.

[10] Resnick, P., and Varian, H. R. (1997). Recommender Systems. Special issue of Communications of the ACM. 40(3).

[11] Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., and Riedl, J. (1998). Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In *Proceedings of CSCW '98*, Seattle, WA.

[12] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000). Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of the ACM EC'00 Conference*. Minneapolis, MN. pp. 158-167

[13] Shardanand, U., and Maes, P. (1995). Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In *Proceedings of CHI '95*. Denver, CO.

[14] Steinbach, M., Karypis, G., and Kumar, V. (2000). A Comparison of Document Clustering Techniques. In *Text Mining Workshop (ACM KDD'00)*.

[15] Ungar, L. H., and Foster, D. P. (1998) Clustering Methods for Collaborative Filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*.