SCARS: A Scalable Context-Aware Recommendation System

Suman Datta*, Joydeep Das†, Prosenjit Gupta* and Subhashis Majumder*
*Dept. of Computer Sc. and Engg, Heritage Institute of Technology, Kolkata, WB, India
Email: sumandatta94@gmail.com
Email: subhashis.majumder@heritageit.edu
Email: prosenjit_gupta@acm.org
†The Heritage Academy, Kolkata, WB, India
Email: joydeep.das@heritageit.edu

***Abstract*—Recommender Systems (RS) are used to provide personalized suggestions for information, products and services that are not already used or experienced by a user, but are very likely to be preferred by him/her. Most of the existing RS employ variations of *Collaborative Filtering* (CF) for suggesting items relevant to users' interests. However, CF requires similarity computations that grows polynomially with the number of users and items in the database. In order to handle this scalability problem and speeding up the recommendation process, we propose a clustering based recommendation method. The proposed work utilizes the different user attributes such as age, gender, occupation, etc. as contextual features and then partitions the users' space on the basis of these attributes. We divide the entire users' space into smaller clusters based on the context, and then apply the recommendation algorithm separately to the clusters. This helps us to reduce the running time of the algorithm as we avoid computations over the entire data. In this work, we present a scalable CF framework that extends the traditional CF algorithms by incorporating users context into the recommendation process. While recommending to a target user in a specific cluster, our approach uses the ratings of the target user as well as the rating history of the other users in that cluster. One of the main objectives of our work is to reduce the running time without compromising the recommendation quality much. This ensures scalability, allowing us to tackle bigger datasets using the same resources. We have tested our algorithm on the MovieLens dataset, however, our recommendation approach is perfectly generalized. Experiments conducted indicate that our method is quite effective in reducing the running time.**

***Keywords*—Collaborative Filtering, Recommender Systems, Data Clustering, Context Awareness, Scalability.**

I. Introduction

Recommender Systems (RS) are software tools and techniques providing suggestions for items to be of use to a user. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, what movies to watch or what online news to read. Recommendation algorithms are extensively adopted by both research and e-commerce applications, in order to provide an intelligent mechanism to filter out the excess of information available in a domain [18]. Examples of some popular RS include product recommendation in Amazon[1], movie recommendation in MovieLens[2], and music recommendation in Last.fm[3].

Collaborative Filtering (CF) [2], [19] is one of the most widely studied and widely adapted techniques behind recommendation algorithms. It tries to recommend items to users based on user-user or item-item similarities computed from existing data, often in the form of ratings given by users. Typical CF-based RS associates a user with a community of like minded users based on their preferences (e.g., ratings) over all the items, and then recommends the target user the items liked by others in that community. Most of the earlier approaches to recommendation algorithms involve two types of objects, users and items, and a function that estimates the rating for a previously unrated item. Such RS focus on recommending the most relevant items to users and do not consider any contextual information, such as time, place, and company of other people. However, in many applications the most relevant information for a user may not only depend on her preferences, but also in her context. Incorporating the contextual information into recommendation process may improve the overall recommendation quality. In E-commerce applications, the intent of a purchase [13] made by a customer can be used as contextual information. For example, the same customer may buy from the same online store different products, like, a book for improving her personal skills, a book as a gift, or an electronic device for her entertainment. Similarly in data mining applications, context may sometimes be defined as those events that characterize the life stages of a customer and which can influence a change in his/her preferences [5]. Contextual information (e.g., time, location, mood, weather, etc.) has been recognized as an important factor that influences the accuracy of recommendations. For instance, John may prefer watching action films with his brothers, but would rather choose a romantic film with his girlfriend. In this case, the companions (brothers vs. girlfriend) is the key contextual information for movie recommendation. Several context-aware RS have been proposed [4], [14] to incorporate contextual information into the existing recommendation algorithms. However, the existing context-aware RS cannot efficiently combine different types of contextual information, and also suffer from high computational complexity.

In this paper, we propose a scalable clustering-based CF

[1]http://www.amazon.com/

978-1-4799-4445-3/15/$31.00

[2]http://www.movielens.org/

[3]http://www.last.fm/

method that partitions the data on the basis of different user contexts. While recommending, our algorithm consider the preferences of the target user as well as the preference ratings of the other users in the neighborhood of the target user. The decomposition algorithm partitions the entire users' space into smaller clusters, and we offer recommendations to the users in those clusters independently. One of the motivations of our work is to reduce the quadratic complexity, typically associated with CF algorithms.

In CF, finding similarity amongst N users is an O(N^2) process. If N is large then similarity computation becomes quite expensive. Decomposition avoids this quadratic blowup and allows us to process bigger data sets even with limited computational resources. As for example, if we partition a region with n users into k partitions with nearly equal sizes, then the overall time required for performing collaborative filtering in all those k partitions will be proportional to $k.(n/k)^2 = (n^2/k^2).k = n^2/k$. So we can achieve a k order speed up by dividing the users' space into k partitions. Note that though we decomposed the users' space into smaller clusters and applied the recommendation algorithm to the clusters independently, it does not mean that two distant users cannot have high correlation in rating. It may also happen that the recommendation quality degrades, as we recommend only using the data of a particular cluster, and not the entire dataset. Our goal is to partition the space into smaller manageable clusters and in turn reduce the overall running time without sacrificing recommendation quality. This ensures scalability, allowing us to tackle bigger datasets using the same hardware resources.

The rest of the paper is organized as follows: In section II, we provide background information and also review some of the past work related to collaborative filtering and context aware recommender systems. Section III outlines our recommendation framework while sections IV and V present our decomposition and recommendation algorithms respectively. In section VI, we report and analyze the experimental results. We conclude discussing our future research directions in section VII.

II. Background and Related Work

A. Collaborative Filtering based Recommender Systems

Recommender systems (RS) can be broadly classified into two groups - (i) Content-Based filtering systems [7] that predict preferences based on the content of the items and the interests of the users, and (ii) Collaborative Filtering (CF) systems [9] that finds a group of users who have similar taste and preferences over items to that of the active user, and recommends to the user those items that were enjoyed by others in the group. The basic premise behind CF is that users who agreed on the past tend to agree in the future also. CF applications typically involve very large datasets, and often it suffers from sparsity and scalability issues.

Most of the existing CF methods based on correlation criteria [15], singular value decomposition (SVD) [16] and non-negative matrix factorization (NNMF) [10] provide highly accurate predictions of ratings. However, these CF techniques suffer from high computational complexity. The correlation-based techniques use similarity measures such as Pearson correlation [6] and cosine similarity [6] to determine a neighborhood of like-minded users for each user and then predict the user's rating for a product as a weighted average of ratings of the neighbors. But, they are computationally very expensive as the correlation between every pair of users needs to be computed during the training phase.

To address the *scalability* issue, Sarwar et al. [17] clustered the complete user set on the basis of user-user similarity and used the cluster as the neighborhood. In contrast, O'Conner et al. [11] used clustering algorithms to partition the item set on the basis of user rating data. Breese et al. [7] utilized Bayesian network and clustering approaches in their recommendation process to address the scalability issue. Xu et al. [21] proposed a multiclass co-clustering model that extends the traditional CF algorithms by incorporating correlated user-item subgroups information in their recommendation algorithm.

In this work, we use a clustering based user partitioning technique to address the scalability problem associated with the CF process.

B. Context-Aware Recommender Systems

In many application domains [13], including recommender systems, contextual information has proved to be useful for providing more accurate predictions. Relevant contextual information does matter in recommender systems and therefore nowadays it is becoming a common practice to incorporate contextual information into the recommender algorithms. Contexts can be obtained in several ways, such as by explicitly gathering from relevant users/items, by implicitly deriving from data or environment, or by inferring using statistical methods, or data mining/machine learning, etc.. Adomavicius and Tuzhilin [3] incorporated the relevant contextual features that may effect the preferences of the user into their recommendation algorithm. They developed a context aware RS that uses both the user preferences as well as the current context in their recommendation process. Adomavicius et al. [1] presented a multidimensional recommendation model based on multiple dimensions, i.e., user/item dimension as well as various contextual information. Similarly, Oku et al. [12] incorporated additional contextual dimensions (such as time, companion, and weather) into the recommendation process and used machine learning techniques to provide recommendations in a restaurant recommender system.

Recent works have focused on building models that directly integrate contextual information with traditional {user, item, rating} relations. For instance, Zhong et al. [22] proposed a contextual collaborative filtering algorithm (called RPMF) to support context-aware recommendation. The assumption behind this model is that contextual information is encoded in or reflected by the user-specific and item-specific latent factors. Li et al. [20] integrated GPS into Recommender System to create a location-aware recommender system. They explored the increasing demand of mobile commerce and developed a recommender system for mobile commerce in tourism. Brown et al. [8] introduce another interesting application that allows tourists interactively share their sightseeing experiences with remote users, demonstrating the value that context-aware techniques can provide in supporting social activities.

III. Our Framework

In order to address the computational complexity of the CF task, we propose a clustering based context aware collaborative filtering framework that utilizes the different contextual features to partition the users' space. Traditional recommender systems normally only consider the ratings of items by users to make recommendations. However, in many systems, rich contextual information is available, providing a new information dimension for recommendation. We define context as something that influences user's decision making process. As for example, users from one age group may prefer items that are vastly different from another age group. Here age is the contextual feature that influences user's decision. Similarly people engaged in similar professions may have similar tastes and preferences. In this work, we use different contexts, such as gender, age and occupation to cluster the entire users' space into smaller homogeneous clusters. The primary objective of our work is to deal with the computational complexity associated with the traditional CF algorithms. After clustering the users' space, we apply the recommendation algorithm separately to the individual cluster. This allows us to reduce the running time of the algorithm as we avoid similarity computations over the entire user data. However, it may degrade recommendation quality as we do similarity computations only over a partial data (not the entire rating data). Our primary objective is to reduce the running time while maintaining an acceptable recommendation quality.

In this work, we have used the MovieLens-1M dataset to test our recommendation algorithm. MovieLens is a Collaborative Filtering recommender system developed by GroupLens Research Group. The dataset contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. Ratings are on a five star (integral) scale from 1 to 5. We use this dataset because it has the contextual attributes like gender, age, occupation, etc. that are required to implement our recommender system. In order to demonstrate the applicability of our method, we have developed a context-aware movie recommendation system that recommend movies according to the context of the target user.

IV. Context Based Decomposition

In this work, we cluster the entire users' space according to gender, age and occupation of the user. Let U represents the entire set of users. We partition the set U into p partitions $U_1, U_2, \cdots, U_p$, where $U_i \cap U_j = \phi$ for $1 \leq i, j \leq p$; and $U_1 \cup U_2, \cdots \cup U_p = U$. For any user u, if $u \in U_i$ then the recommendation algorithm use the entire cluster U_i as the neighborhood.

Our clustering algorithm is implemented using MovieLens dataset. The dataset has demographic information like gender, age, occupation and location (zip-code) about the users. In this work, we use gender, age and occupation attributes of the user to cluster the users' space hierarchically. Our clustering technique is stated below.

We propose two alternative approaches (*Scheme1* and *Scheme2*) to partitioning the users of the dataset as defined below.

- In *Scheme1*, We first divide the users on the basis of the gender into two gender clusters (*Male* and *Female*). Then the gender clusters are further divided into four age clusters (*A_18*, *A_25*, *A_35*, and *A_Oth*). Cluster *A_18* contains users having age less than equal to 18. *A_25* contains users having age between 19 and 25. Users having age between 26 and 35 are put in the cluster *A_35* and the rest of the users (age more than 35) are merged in *A_Oth*. We pictorially represent this scheme in Fig. 1.
- In *Scheme2*, like *Scheme1*, users are first split into gender clusters, and then the gender clusters are further divided into occupation clusters as shown in Fig. 2. In Fig. 2, we can see that there are 21 occupation clusters (O:0, O:1, ... O:21) for both the Male and Female clusters. The occupations correspond to the actual occupation of the users present in the MovieLens dataset.

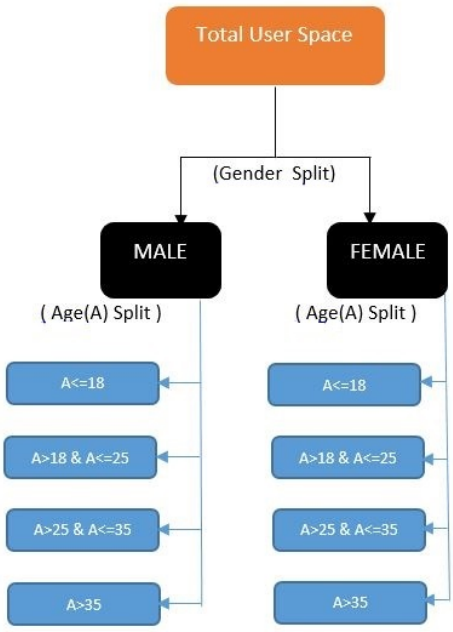

Figure 1. Clustering by gender and age

The decomposition process partitions the users' space into smaller clusters on the basis of the different attributes of the users. We now apply the recommendation algorithm individually to the clusters, and compare the performance of the algorithm applied to the clusters separately, with the corresponding performance while applied to the entire user dataset (without decomposition). We will also compare the overall performance of the recommendation algorithm applied to clusters formed by *Scheme1* with the performance in the corresponding clusters formed as a result of *Scheme2*.

V. The Recommendation Algorithm

As we have already clustered the users into smaller cells, so we can start our recommendation process. We use CF algorithms for recommendation generation. The main idea is to apply the CF algorithm separately to the clusters and only use the ratings of that cluster. In this work, We recommend

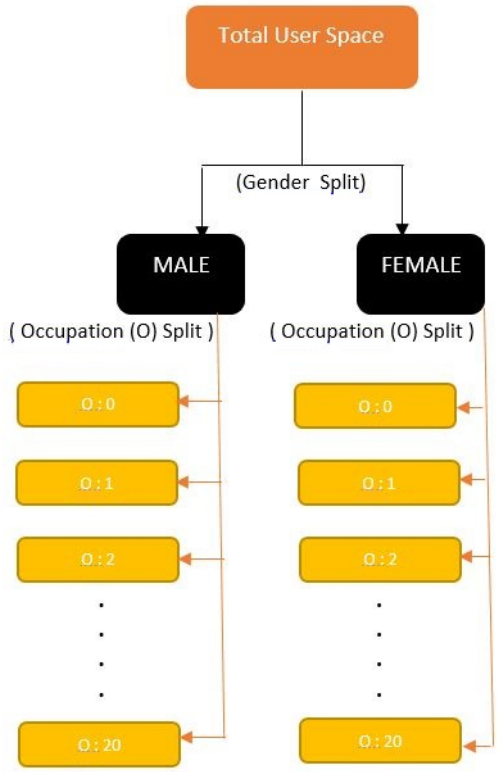

Figure 2. Clustering by gender and occupation

using a user based CF approach. In CF, we need to define a similarity metric that computes the similarity between two users. Our work uses *cosine similarity* metric [6] to compute similarities between users. The cosine similarity between two users is higher if the two users have purchased a larger set of common items. Our system provide recommendations for the following two categories of users.

- For an existing user in a cluster, the algorithm first finds her collaborative users based on the similarity score and then recommend *Top-N* items using the taste and preferences of the collaborative users.

- For a new user of the system, the algorithm use the contextual attributes like gender, age and occupation of the user to place her in the destined cluster. The *Top-N* items highly rated by the users in the cluster are recommended to the user.

We find the *collaborative users* for the target user by calculating the similarity score between the target user and all other users in the cluster. We choose the *top-K* collaborative users according to the similarity score. Next, we form a set of top rated items (movies) by using the ratings of the *top-K* collaborative users. This set include only those items whose average rating from all the K similar users is more than a threshold value. Then the items in this *item set* is again ranked in order of their rating frequency (no. of users rating the item). The system recommends to the target user the *top-N* items from the *item set* not rated by the user. We implemented the algorithm with K = 100 and N = 10.

Algorithm Recommend_Item

step1: Select a target user for recommendation.

step2: Assign the user to a cluster according to her contextual attributes (gender, age, occupation).

step3: Find *top-K collaborative users* of the target user by calculating *cosine similarity* score.

step3.1: Find users with *cosine similarity* value = 1 (highly correlated).

step3.2: If this number exceeds K, consider all the users.

step3.3: If not, sort (descending) the users according to their similarity value and select users from this sorted list such that number of *collaborative users* equals K.

step4: Form a top rated *item set* by using the ratings of the *top-K collaborative users*.

step5: Recommend *top-N* items from the *item set* that are not rated by the target user.

VI. Experiments and Results

We have conducted several experiments to evaluate the effectiveness of the proposed method. In this section, we report the result of the experiments performed and also make an empirical analysis of the results. We have tested our recommendation algorithm on the MovieLens-1M dataset to validate our scheme. The user ratings of the dataset are randomly split into two sets - observed items (80%) for training and held-out items (20%) for testing. Ratings for the held-out items were to be predicted.

A. Evaluation Metric Discussion

Two commonly used metrics for evaluating the prediction accuracy of traditional Collaborative Filtering algorithms are Mean Absolute Error (MAE) [6] and Root Mean Square Error (RMSE) [6]. In this work, we use MAE and RMSE to evaluate the prediction accuracy while quality of the recommendation is measured using the *Precision*, *Recall* and *F1 score* metric.

MAE: MAE is defined as the average of the absolute error. Absolute error is the difference between the predicted rating and actual rating. Let the actual user ratings be, $\{r_1, r_2, \cdots, r_n\}$, and predicted ratings are, $\{p_1, p_2, \cdots, p_n\}$, where n is the number of items. Then Absolute error,

$$E = \{e_1, e_2, \cdots, e_n\} = \{p_1 - r_1, p_2 - r_2, \cdots, p_n - r_n\}$$

and

$$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

Any prediction algorithm tries to minimize the MAE.

RMSE: RMSE is similar to MAE and is biased to provide more weights to larger errors.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}}$$

We have depicted the different combinations of recommendation that can be generated in a typical recommendation problem in Table I. Note that a customer likes an item (movie) if he has given a rating of 4 or 5 to that item (in a scale of 1 to 5), otherwise dislikes it, i.e., his rating is 1, 2 or 3. A recommendation is positive if recommended rating coincides with the actual rating given by the customer.

Table I. POSSIBLE RECOMMENDATIONS

	Customer Likes (rating = 4 or 5)	Customer Dislikes (rating = 1, 2 or 3)
Recommend	True positives	False positives
Do not recommend	False negatives	True negatives

Precision: Precision measures the degree of accuracy of the recommendations produced by the algorithm. In our system, Precision measures what fraction of the recommended items are liked by the customers.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall: The Recall metric is also known as the hit rate, which is widely used for evaluating *top-K* recommender systems. In our recommender system, Recall measures what fraction of the items liked by the customers, has been recommended by the algorithm.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

F1 measure: F1 measure or F1 score is the harmonic mean of Precision and Recall.

$$F1 = \frac{2 * Precision\ * Recall}{Precision + Recall}$$

B. Recommendation performance on MovieLens-1M dataset

We apply the recommendation algorithm individually to each cluster. Our objective is to execute the algorithm only using the ratings of a particular cluster, which permits us to avoid the similarity computations over the entire dataset. Our experiments are performed on the MovieLens-1M dataset. While recommending items to a user, the algorithm takes into account only the $\{user, item, rating\}$ triplets of the *collaborative users* of the target user. We execute the algorithm with $K = 100$ and $N = 10$, as discussed in section V. That is we consider a maximum of 100 *collaborative users* and recommend *top-10* items to the user.

To evaluate the quality of overall recommendation, we have tested the recommendation algorithm for all the clusters formed as a result of the decomposition process. As mentioned in section IV, we have clustered the users' space using two alternative approaches - *Scheme1* and *Scheme2*. We report the results of the recommendation algorithm applied to the clusters formed by *Scheme1* approach in Table II and that of *Scheme2* approach in Table III. In the Tables, we make a comparative analysis of the recommendation performance using different evaluation metrics. Here *base performance* indicates the performance of the algorithm using the entire users' space (without decomposition). We compare the overall performance in the clusters formed by the context based decomposition method with the *base performance*. We use MAE and RMSE to evaluate the prediction accuracy and also use Precision@K, Recall@K and F1@K to evaluate the quality of the *top-K* recommended items. Note that, we present Precision (P@10), Recall (R@10) and F1 ($F1$@10) score on position 10. The bold numbers indicate that its value has an obvious improvement than the base value.

Table II. PERFORMANCE COMPARISONS ON MOVIELENS-1M DATASET USING SCHEME1 DECOMPOSITION APPROACH.

	No. of Clusters	P@10 (Avg)	R@10 (Avg)	$F1$@10 (Avg)	MAE (Avg)	$RMSE$ (Avg)
Base performance	1	0.970	0.736	0.815	0.429	0.536
Gender_Split	2	0.858	**0.830**	**0.828**	0.439	0.551
Male_Age_Split	4	0.896	**0.829**	**0.842**	**0.405**	**0.514**
Female_Age_Split	4	0.903	**0.766**	0.804	0.436	**0.521**

Table III. PERFORMANCE COMPARISONS ON MOVIELENS-1M DATASET USING SCHEME2 DECOMPOSITION APPROACH.

	No. of Clusters	P@10 (Avg)	R@10 (Avg)	$F1$@10 (Avg)	MAE (Avg)	$RMSE$ (Avg)
Base performance	1	0.970	0.736	0.815	0.429	0.536
Gender_Split	2	0.858	**0.830**	**0.828**	0.439	0.551
Male_Occup_Split	21	0.881	**0.839**	**0.859**	**0.412**	0.541
Female_Occup_Split	21	0.911	**0.854**	**0.842**	0.432	**0.511**

In the Tables II and III, we have reported the performance of our recommendation algorithm averaged over all the clusters. As for example, for Gender_Split case, we have an average Precision, Recall, F1, MAE, RMSE of 0.858, 0.830, 0.828, 0.439 and 0.551 respectively averaged across the two clusters (Male and Female). Similarly for Male_Age_Split case, we have an average Precision, Recall, F1, MAE, RMSE of 0.896, 0.829, 0.842, 0.405 and 0.514 averaged across all the 4 age based clusters. Here we can see that the algorithm performs better in terms of Recall, F1, MAE and RMSE while in terms of Precision, the *base performance* is slightly better. In table III, we can note that for Male_Occup_Split case, we have better Recall, F1 and MAE values over base while *base performance* is better in terms of Precision and RMSE. Since we executed the algorithm only using the ratings of a particular cluster, it may sometimes compromise our recommendation quality as two users in two different clusters may have similarity in the rating patterns. However, from the above tables, it is clear that our algorithm always performs better (in terms of Recall) than the *base performance*, while for the other evaluation metrics it has values which are nearly equal to the base.

C. Scalability

We report the running time of our algorithm for *Scheme1* and *Scheme2* based recommendation approaches in Tables IV and V respectively. Here we record the overall time required for testing the algorithm (in minutes) in all the clusters formed by the decomposition process. Note that, the running time comprises of both the cluster formation time and recommendation generation time for all the users of a cluster. Here

base represents the entire dataset without decomposition. Our experiments are run on a computer with Core i3 - 2100 @ 3.10GHz x 4 CPU and 4 GB RAM.

Table IV. RUNNING TIME COMPARISONS ON MOVIELENS-1M DATASET USING SCHEME1 BASED RECOMMENDATION APPROACH.

	No. of Clusters	$Time(Minutes)$
Base	1	1233.3
Gender_Split	2	845.20
Male_Age_Split	4	510.43
Female_Age_Split	4	491

Table V. RUNNING TIME COMPARISONS ON MOVIELENS-1M DATASET USING SCHEME2 BASED RECOMMENDATION APPROACH.

	No. of Clusters	$Time(Minutes)$
Base	1	1233.3
Gender_Split	2	845.20
Male_Occup_Split	21	258.32
Female_Occup_Split	21	221.20

In Table IV, we can observe that running time improves significantly when we divide the entire dataset into smaller cells and apply the algorithm individually to the cells. As for example, using *Scheme1* approach, for Male_Age_Split case, the overall time required for recommending all the users in the 4 clusters is 510.43 minutes while that of the entire dataset is 1233.3 minutes. Thus the running time reduces by about 58% over the base performance. Similarly for the Female_Age_Split case, running time reduces by about 61%. In Table V, we can also notice similar results, that is, clustering approach always allow us to achieve significantly improved runtime.

From the above analysis we can conclude that our method is effective in reducing the running time. We have also seen in the previous section that the recommendation algorithm produces quality recommendations when applied to the clusters separately. Thus we achieve our goal of partitioning the users' space into smaller cells and in turn reduce the overall running time without sacrificing the recommendation quality much. This ensures scalability and establishes that our method will be effective to deal with even bigger datasets.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a clustering based scalable context aware recommender system. We have implemented a decomposition technique that divides the users' space into smaller clusters on the basis of the different contextual attributes. Experimental analysis using real datasets show that our model is efficient and scalable. Our proposed approach deals with the *Scalability* problem of the CF based recommendation methods by applying the recommendation algorithm separately to the regions. Online experimentation of the clustering algorithm and the recommendation algorithm will be the focus of our future work. We will also try to use other metrics for finding similarities between users with the aim of optimizing the splitting technique and the recommendation algorithm.

REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM transactions on information systems*, vol. 23, no. 1, pp. 103–145, 2005.

[2] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, pp. 734–749, 2005.

[3] ——, "Context-aware recommender systems," in *Recommender Systems Handbook*, 2011, pp. 217–253.

[4] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proceedings of the fifth ACM conference on Recommender systems*, 2011.

[5] M. J. Berry and G. Linoff, *Data mining techniques: for marketing, sales, and customer support*. John Wiley & Sons, Inc. New York, NY, USA, 1997.

[6] B. Bhasker and K. Srikumar, *Recommender Systems in e-Commerce*. McGraw-Hill Education, 2010.

[7] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*, 1998, pp. 43–52.

[8] B. Brown, M. Chalmers, M. Bell, M. Hall, I. MacColl, and P. Rudman, "Sharing the square: collaborative leisure in the city streets," in *Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work*, 2005, pp. 427–447.

[9] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, 2000, pp. 241–250.

[10] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM transactions on information systems*, vol. 22, no. 1, pp. 89–115, 2004.

[11] M. O'Connor and J. Herlocker, "Clustering items for collaborative filtering," in *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 1999.

[12] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura, "Context-aware svm for context-dependent information recommendation," in *Proceedings of the 7th International Conference on Mobile Data Management*, 2006, p. 109.

[13] C. Palmisano, A. Tuzhilin, and M. Gorgoglione, "Using context to improve predictive modeling of customers in personalization applications," *IEEE transactions on knowledge and data engineering*, vol. 20, no. 11, pp. 1535–1549, 2008.

[14] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrival*.

[15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of ACM conference on Computer Supported Cooperative Work*, 1994, pp. 175–186.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender systems a case study," in *WebKDD Workshop*, 2000.

[17] ——, "Recommender systems for large-scale e.commerce: Scalable neighborhood formation using clustering," in *Proceedings of the Fifth International Conference on Computer and Information Technology*, 2002, pp. 158–167.

[18] J. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," in *Proceedings of the 1st ACM conference on Electronic Commerce (EC-99)*, 1999, pp. 158–166.

[19] X. Su and T. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009.

[20] L. Xinyu, M. Zhongchun, Z. Zhenmei, and J. Wu, "A location-aware recommender system for tourism mobile commerce," in *Proceedings of the 2nd International Conference on Information Science and Engineering (ICISE)*, 2010, pp. 1709–1711.

[21] B. Xu, J. Bu, C. Chen, and D. Cai, "An exploration of improving collaborative recommender systems via user-item subgroups," in *Proceedings of the 21st international conference on World Wide Web (WWW' 2012)*, 2012, pp. 21–30.

[22] E. Zhong, W. Fan, and Q. Yang, "Contextual collaborative filtering via hierarchical matrix factorization," in *Proceedings of the SIAM International Conference on Data Mining*.