Context Aware Scalable Collaborative Filtering

Joydeep Das*, Subhashis Majumder† and Kalyani Mali‡
*The Heritage Academy, Kolkata, WB, India
Email: joydeep.das@heritageit.edu
†Dept. of Computer Sc & Engg, Heritage Institute of Technology, Kolkata, WB, India
Email:subhashis.majumder@heritageit.edu
‡Dept. of Computer Sc & Engg, University of kalyani, Kalyani, WB, India
Email:kalyanimali1992@gmail.com

Abstract—**Traditional Collaborative Filtering (CF) based Recommender Systems (RS) typically do not consider the contextual attributes of users or items while making recommendations. However, there are plenty of applications in day to day life, where the choices made by a person may not only depend on his or her earlier preferences, but more on the context. In this work, we incorporate the contextual attributes to cluster the original user-item rating matrix and then apply CF based recommendation algorithms to the clusters independently. It helps to bring down the runtime as we can bypass computations over the entire data and also generate more accurate recommendations as the partitioned clusters hold similar ratings, which produce greater impacts on each other. In order to cluster the rating matrix, we use both centroid based (k-means) and density based (DBSCAN) clustering techniques and finally compare the results. One of the main objectives of our work is to make it more scalable by reducing the runtime without compromising the recommendation quality much. Experiments performed on two publicly available datasets: MovieLens-1M and MovieLens-100K show that our approach is scalable and accurate.**

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

I. Introduction

In recent years, due to the exponential growth of online available information, people face difficulties in making their choices from an overwhelming set of choices. Recommender System (RS) [2] have been developed to alleviate the problem of information overload by suggesting relevant items according to the preferences of a querying user. RS plays an important role in e-commerce (e.g., Netflix, Amazon), social networks (e.g., LinkedIn, Foursquare), review sites (e.g., Epinions, Movielens), mobile applications, information retrieval and so on [1], [12].

Most of the existing RS rely on Collaborative Filtering (CF) techniques [2], [16], which predict a user's interest in an item by utilizing the past rating history of other similar users and/or items. CF based systems typically use only the ratings of users on items while recommending without taking into consideration any other influential information available in an application area. However, in many applications, contextual information (e.g., time, location, weather, mood, etc.) has been recognized as an important factor that may influence the recommendation quality. For instance, Alice may like watching action films with his brothers, but would rather prefer a comedy film with his parents. In this case, the companions (brothers vs. parents) is the key contextual information for movie recommendation. Similarly a travel recommender system would provide a vacation package recommendation in summer which can be very different from the one in winter. Therefore, integration of contextual information with the conventional rating based system may improve the overall recommendation quality. Over the last decade, different context-aware RS have been proposed [4], [19] to incorporate contextual information in the existing CF framework, e.g., matrix factorization models [10]. However, most of the existing context-aware recommender systems fail to integrate multiple types of contextual information (e.g., discrete versus continuous values [9]). Some of them also have high computational complexity (e.g., matrix factorization model [19]). So, a more scalable architecture is required which can effectively integrate the different contextual information with the existing approaches to further improve the recommendation quality.

In this paper, we propose a scalable clustering based CF framework that partitions the entire user-item-rating matrix on the basis of the various contextual information. Since the generated clusters contain similar ratings, therefore the missing ratings predicted from the clusters are more accurate and personalized than those directly obtained using the original rating matrix. In this work, we apply CF based recommendation algorithms separately to the clusters with the idea of reducing the costly similarity computations involved in the CF process. Our main objective is scalability, i.e., to speed up the recommendation task so that larger datasets can be tackled with existing computational infrastructure while maintaining an acceptable recommendation quality. Experimental results show that our context aware CF significantly improves the performance of the traditional CF based RS. In the rest of the paper, we use the term CASCF (Context Aware Scalable Collaborative Filtering) to represent our model.

The rest of the paper is organized as follows: In section II, we provide background information and also review some of the past works related to CF and context-aware RS. Section III outlines our contribution while section IV presents our experimental setup. In section V, we report and analyze the experimental results and in section VI, we conclude discussing our future research directions.

II. Background and Related Work

A. Collaborative Filtering based Recommender Systems

Collaborative Filtering (CF) is a recommendation technique that identifies similarities between users (or items), based on their ratings in order to select neighbors and then compute predictions for the active user on the basis of the ratings of its neighbors. According to Breese et al. [5], CF algorithms can be divided into two classes: memory based and model based algorithms. Memory based algorithms identify the top-*K* most similar users (neighbors) to the active user, and then use a weighted sum of the ratings of the neighbors to predict missing ratings for the active user. A general user-based formulation of the weighted sum scheme can be [5]:

$$p_{a,j} = \bar{r}_a + k \sum_{i=1}^{n} w(a,i)(r_{i,j} - \bar{r}_i)$$

where n is the size of the neighborhood and $\bar{r}_a$ and $\bar{r}_i$ are the average ratings for the active user a and neighbor user i respectively. $w(a,i)$ is the correlation between user a and user i. k is a normalizing factor. The most important part for memory based algorithms is the similarity computation. Given two users, u and v, a similarity function based on Pearson's correlation coefficient can be defined as follows.

$$Sim(u,v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$

where the $i \in I$ summations are over the items that both the users u and v have rated and $\bar{r}_u$ is the average rating of the co-rated items of the u'th user. Memory based algorithms are easy to implement and use. In this approach, new data can be added easily and incrementally, and the technique also scale well with co-rated items. However, its performance decreases when data gets sparse and also fails to recommend for new users and items.

Model based algorithms implement data mining or machine learning algorithms on the training data to estimate or learn a model to make predictions for an active user. Some well-known model based CF algorithms include clustering CF models [15], [6], latent semantic models [8], and regression-based models [17]. Model based algorithms handle the sparsity and scalability problems better than the memory based algorithms. It also improves the prediction performance. The disadvantages with this technique are in costly model building and updating. One can lose useful information due to dimensionality based reduction techniques.

B. Context-Aware Recommendation Systems

The incorporation of contextual information in RS has been a popular research area in the last decade. The importance of including and using the contextual information in recommendation systems has been discussed by Adomavicius et al. [1], where the authors presented a multidimensional approach that can provide recommendations based on contextual information and also demonstrated that the use of contextual information helps to increase the quality of recommendations in certain settings. We can obtain contexts by explicitly capturing data from relevant users or items, by implicitly deriving from data or environment, by inferring using statistical methods, or data mining/machine learning, etc. [3].

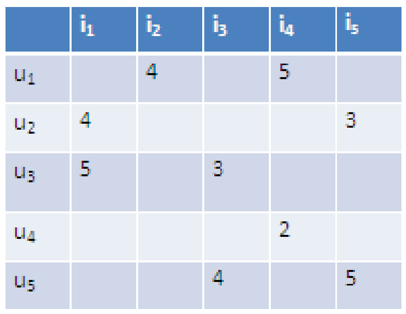
Recently Xiaoyao et al. [18] incorporated contextual information into a CF approach based on matrix factorization method, and at the same time used data clustering techniques to optimize their proposed system. Similarly, Liu et al. [11] integrated social network information with contextual attributes and proposed a context-aware recommender system. They used random decision trees to partition the original user-item-rating matrix and then applied matrix factorization based CF algorithm in the partitioned matrix. Zhong et al. [19] proposed a contextual collaborative filtering algorithm (called RPMF) with the assumption that contextual information is encoded in or reflected by the user-specific and item-specific latent factors. On the basis of this, tree based partitioning is randomly applied to split the user-item-rating matrix by forming clusters of items and users having similar contexts, and finally factorizing the sub-matrices corresponding to each of those clusters.

III. Our Recommendation Model

Conventional RS normally consider only two dimensions - users and items to make recommendations. However, in certain applications, rich contextual information is available which can be used as an additional dimension to enhance the quality of recommendation. In Fig. 1(a) we have depicted a sample user-item-rating matrix while its context aware variant user-item-context-rating matrix is shown in Fig. 1(b). Contexts can be defined as something that influences users decision making process. We can often notice that people from one age group may prefer items that are vastly different from the people of another age group. Here age is the contextual feature that influences users decision. Similarly, people engaged in similar professions may have similar tastes and preferences. In CASCF model, we classify contextual information into two categories: (1) static context, which represents the characteristics of a user, such as age, gender, profession, etc.; (2) dynamic contexts, which represents instantaneous information associated with a rating, such as user's mood, companion, location or time when he/she rates an item. In this work, we incorporate both static and dynamic contexts into a traditional CF based recommender system.

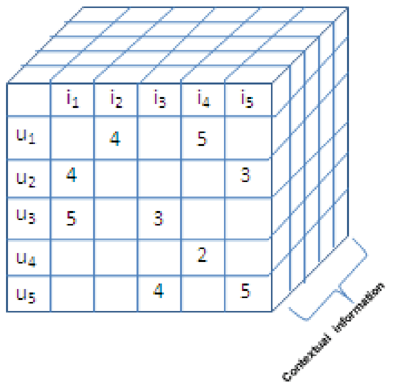
In our work, we use contextual information as the parameter to cluster the original user-item rating matrix and then then apply CF based recommendation algorithm to the resultant sub-matrices individually. This helps to reduce the complexity as the similarity computations involved in CF are confined within the cluster as well as generate more accurate recommendations since ratings with similar contexts are grouped in a cluster. We consider that each rating R_i is associated with a set of contextual information denoted by vector $C = \{c_1, c_2, \cdots, c_l\}$, where vector component c_x value is in {0,1}. For example, c_1 may denote age (teenage), c_2 may denote gender (male) and c_3 may denote time (Weekday). In this context, the contextual information that "John aged 30 went to see the movie 'Star Wars' on Wednesday" can be represented as $\{0, 1, 1\}$. In this way, we represent all the ratings as vectors in the contextual space. We use discrete values to represent contextual information and then cluster the contextual vectors such that contextual information having similar values are grouped in one cluster. In this work, we use two different clustering techniques - one centroid based (k-means) and one density based (DBSCAN) [7] to cluster the original user-item rating matrix using contextual information and finally compare the results.

Suppose that the rating dataset has N ratings. We represent each of the ratings as contextual vectors using the method defined above (see Table I). Let us denote the contextual information for user u_i to select item i_j by the vector $C_{ij} = \{c_{ij}^1, c_{ij}^2, \cdots, c_{ij}^l\}$, and the contextual information for user u_m



	i_1	i_2	i_3	i_4	i_5
u_1		4		5	
u_2	4				3
u_3	5		3		
u_4				2	
u_5			4		5

(a) User-Item-Rating matrix



(b) User-Item-Context-Rating matrix

Figure 1: Context-Aware Recommendation

to select item i_n by the vector $C_{mn} = \{c^1_{mn}, c^2_{mn}, \cdots, c^l_{mn}\}$. Then the distance between two contextual vectors can be defined as [18]:

$$d(C_{ij}, C_{mn}) = \sum_{x=1}^{l} \delta\left(c^x_{ij}, c^x_{mn}\right) \tag{1}$$

where

$$\begin{aligned} \delta\left(c^x_{ij}, c^x_{mn}\right) &= 0, \quad c^x_{ij} = c^x_{mn} \\ &= 1, \quad c^x_{ij} \neq c^x_{mn} \end{aligned}$$

Table I: Example of Contextual Vectors

Contexts			
Teenager	Male	Weekday	Contextual Vectors
Yes	No	Yes	{1, 0, 1}
Yes	Yes	Yes	{1, 1, 1}
No	Yes	Yes	{0, 1, 1}
Yes	No	No	{1, 0, 0}

A. *k-means based clustering*

In order to implement the k-means algorithm, we need to define k, the number of clusters. We use the term object to denote a contextual vector. k objects are randomly selected as the initial centers of the k clusters. For each of the remaining objects, we calculate the distance between the object and all the cluster centers and assign the object to the nearest cluster. Next, iteratively the cluster centers are recalculated until the centers do not change any more. This clustering process is shown in its algorithmic form in Algorithm 1.

Algorithm 1: k-means clustering algorithm

```
Input: A dataset containing $N$ contextual vectors
       (objects), number of clusters $k$
Output: A set of $k$ clusters with similar contexts
1 Randomly select $k$ objects from the dataset as the initial
  centers for $k$ different clusters.
2 For the remaining objects, calculate the distance
  between the object and the cluster centers and assign
  the object to the cluster whose center is nearest to that
  object, ties being broken arbitrarily.
3 Update the cluster centers, i.e. recalculate the mean
  value of the objects for each cluster.
4 Repeat step 2 and step 3 until the cluster centers do not
  change any more.
```

B. *DBSCAN based clustering*

In this work, we also implement a density based clustering technique (DBSCAN). DBSCAN [7] algorithm requires two parameters: (i) ε-neighborhood (eps), which defines the radius of the neighborhood of a given object and (ii) *MinPts*, the minimum number of objects required to form a cluster. We implement the clustering algorithm using the same contextual vectors defined earlier and the distance between two vectors is calculated using equation 1. The clustering algorithm is detailed in Algorithm 2.

Algorithm 2: DBSCAN clustering algorithm

```
Input: A dataset containing $N$ contextual vectors
       (objects), MinPts, $\varepsilon$-neighborhood
Output: A set of clusters with similar contexts
1 Select an arbitrary starting object that has not been
  visited.
2 Extract the neighborhood of this object using $\varepsilon$ (All
  objects which are within a distance of $\varepsilon$ are in its
  neighborhood).
3 If the number of objects present in $\varepsilon$-neighborhood $\geq$
  MinPts, a cluster is started. Otherwise, the object is
  labeled as noise (Later this object can become the part
  of another cluster).
4 If a object is found to be part of a cluster then its
  $\varepsilon$-neighborhood is also part of that cluster and the
  above procedure from step 2 is repeated for all
  $\varepsilon$-neighborhood objects. This is repeated until all
  objects in the cluster is determined.
5 Then a new unvisited object is retrieved and processed,
  leading to the discovery of another cluster or noise.
6 This process continues until all points are marked as
  visited.
```

C. *Recommendation with Clusters*

We now describe how to integrate the clusters obtained in the previous sections with the existing CF models. As already discussed, we apply the recommendation algorithm individually to the clusters with the aim of reducing the computational complexity as well as better recommendation accuracy. In case of conventional CF methods like Matrix Factorization (MF) or user-based CF, the user-item matrix is the only required input and the output is the prediction score for each rating that is absent in the matrix. In CASCF, for each cluster C_k, we can actually get a submatrix from the original big user-item rating matrix R, with only users, items and ratings belonging to that cluster. Therefore we can directly apply any CF method without any modification. The pseudocode of the *top-N* recommendation is shown in Algorithm 3.

Algorithm 3: *Top-N* Recommendation Algorithm

```
Input: Rating matrix $R$, a set of clusters
       $\{C_1, \cdots, C_L\}$, a chosen CF method, and the
       number of items in recommendation list
Output: Recommendation list for each user.
1  for $k \leftarrow 1$ to $L$ do
2      Extract submatrix $R_k$ from rating matrix $R$ with
       users and items belonging to cluster $C_k$;
3      Apply the CF recommendation method with $R_k$ as
       input and predict missing ratings.
4      for $i \leftarrow 1$ to $s$ do
5          Form a Item Set for user $u_i \in C_k$;
6          Item set contains only those items whose
           $predicted\ rating \geq threshold\, value$;
7          Sort the items in the Item Set according to the
           descending order of rating frequency (no. of
           users rating the item);
8          Recommend a list of top-N items from the Item
           Set to user $u_i$;
9      end
10 end
```

IV. Experimental Settings

A. Data Description

Our experiments are performed on the MovieLens-1M and MovieLens-100K datasets[1]. MovieLens-1M data consists of 1,000,209 anonymous ratings (1-5) of approximately 3,900 movies made by 6,040 MovieLens users where each user has rated at least 20 movies. MovieLens-100K dataset contains 943 users providing 100,000 anonymous ratings (1-5) on 1682 movies.

B. Evaluation Metric Discussion

In order to evaluate the prediction accuracy of CASCF, we use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [16]. The objective of any prediction algorithm is to minimize the MAE and RMSE values. Since we are interested in generating a list of *Top-N* recommended items, therefore MAE and RMSE are not sufficient to evaluate the recommendation quality. Therefore, we evaluate the quality of the ranked list using F1 measure and MAP (Mean Average Precision) [13]. In Table II, we have shown the different possible combinations of recommendation. A recommendation is considered as relevant if the customer gives a rating of 4 or 5 to that item (in a scale of 1 to 5). True Positive (Negative) or t_p (t_n) denotes the case where a product which is recommended (not recommended) by the system is relevant (irrelevant) to a customer. Similarly, False Positive (f_p) denotes the case where an item irrelevent to a customer has also been recommended by the system. Finally False Negative (f_n) is the case where the system fails to recommend an item that a customer likes.

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

Table II: Combinations of Recommendation

	Relevant (rating = 4 or 5)	Irrelevant (rating = 1, 2 or 3)
Recommend	t_p	f_p
Do not recommend	f_n	t_n

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

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

$$Precision = \frac{t_p}{t_p + f_p}, \quad Recall = \frac{t_p}{t_p + f_n}$$

F1 measure combines Precision and Recall with equal priority, which makes it easy to compare algorithms across datasets.

MAP: Given a ranked list with n items, for each user u, we denote *prec*(j) as the precision at rank position j, and *pref*(j) as the preference indicator of item at position j. If the item at position j is rated by user u, then *pref*$(j) = 1$, otherwise *pref*$(j) = 0$. Average Precision (AP) is computed as the average of precisions computed at each position in the ranked list. MAP is the mean of AP for all users.

$$AP(u) = \frac{\sum_{j=1}^{n} prec(j) \times pref(j)}{n}$$

$$MAP = \frac{1}{|U|} \sum_{u \in U} AP(u)$$

Higher F1 and MAP imply better recommendation performance.

C. Comparison

In this work, we implement two memory based CF algorithms - user-based and item-based and one matrix factorization method of SVD. Then we combine these recommendation methods with our framework to verify whether their performance is improved.

User-based: In order to implement the user-based algorithm, we use Pearson's correlation coefficient [16] as the similarity metric for finding user-user similarities and use the user-based model in [16].

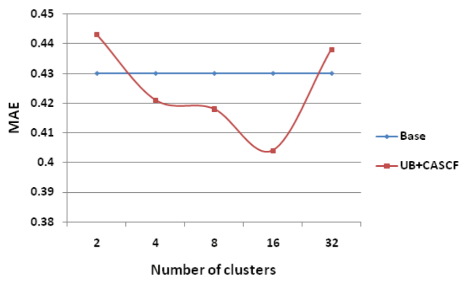
Item-based: For item-based method, we use Cosine-Based similarity metric [16] to compute the item-item similarities and use the item-based model defined in [16].

SVD: In SVD, the user-item matrix is decomposed into three matrices with k features: $R = U_k S_k V_k^T$. Then the prediction score for the i-th customer on the j-th product is given by $P_{i,j} = \bar{r_i} + U_k \sqrt{S_k}^T(i).\sqrt{S_k} V_k^T(j)$, where $\bar{r_i}$ is the i-th row average according to [14]. In this work, we use the value of k as 6.

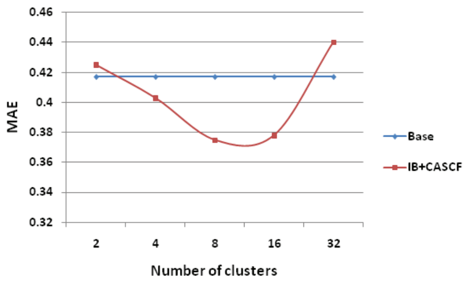
V. Results and Discussion

A. Performance on MovieLens-1M dataset

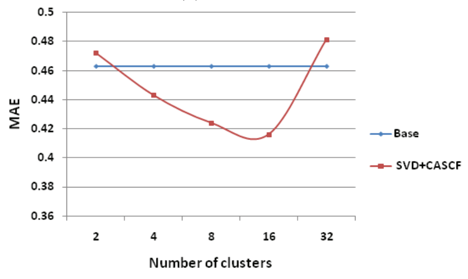
In this work, we randomly split the user ratings of the dataset into two sets - 80% for training purpose and 20% for testing.



(a) User-based

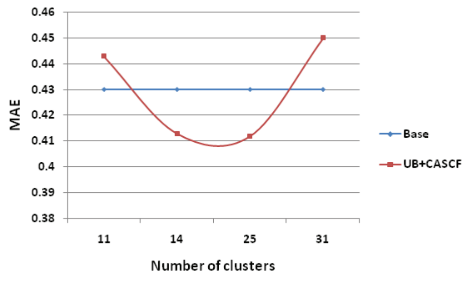


(b) Item-based

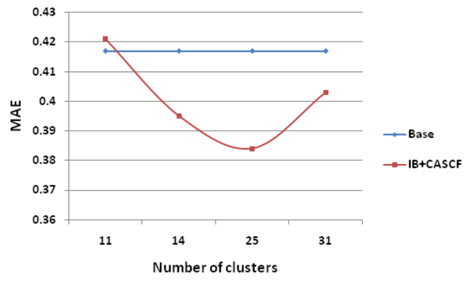


(c) SVD

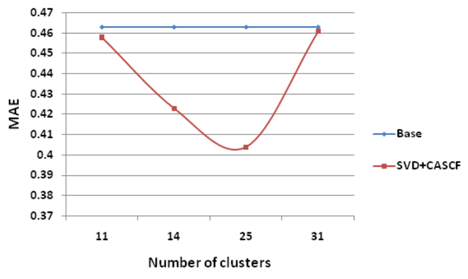
Figure 2: Comparisons of the prediction performance on MovieLens-1M Dataset in terms of MAE by k-means method.



(a) User-based



(b) Item-based



(c) SVD

Figure 3: Comparisons of the prediction performance on MovieLens-1M Dataset in terms of MAE by DBSCAN method.

In order to evaluate the effect of number of clusters on recommendation accuracy, we vary the value of k for k-means algorithm from 2 to 32. In DBSCAN algorithm, we do not require to specify the number of clusters in the data a priori, and the number of clusters depend on the values of ε-neighborhood (eps) and $MinPts$. In this work, we use the following pairs of eps and $MinPts$: $\{1, 100\}$, $\{1, 200\}$, $\{2, 100\}$, $\{2, 200\}$ and accordingly got 25, 11, 31 and 14 clusters respectively. Next, we report the prediction accuracy of the recommendation algorithm in terms of MAE for k-means and DBSCAN methods in Figures 2 and 3 respectively. Here we compare the MAE values of our method using the three CF approaches described earlier. In each subgraph of Fig. 2 and Fig. 3, Base represents the original performance of that method, i.e., without clustering. We compare the Base performance with the performance of our model CASCF. In Fig. 2(a), (b) and (c), we can see that our model has a lower predicted error when k value is 8 or 16, and the values of MAE grow quickly when k is more than 16. Similarly in Fig. 3, we note that our model has lower MAE values when number of clusters lie between 14 and 25. Analyzing Fig. 2 and Fig. 3, we can conclude that our method has outperformed the base values in most of the cases.

The other three evaluation metrics: F1@10, MAP, and RMSE are recorded in Tables III and IV for k-means and DBSCAN clustering techniques respectively. In the tables, Base represents the original performance of the different CF methods, and we compare the Base performance with the performance of the methods using our clustering model. The bold numbers indicate that those values have improved over the Base values. From the results presented in Table III and Table IV, we again find that irrespective of the CF methods, in most of the cases the values of F1, MAP and RMSE using our clustering based approach are better than the corresponding Base values.

Table III: Performance Comparisons on MovieLens-1M Dataset using k-means

# of clusters	F1@10			MAP			RMSE		
	User based	Item based	SVD	User based	Item based	SVD	User based	Item based	SVD
1 (Base)	0.314	0.376	0.421	0.384	0.415	0.443	0.615	0.584	0.558
2	0.285	0.358	0.406	0.379	0.396	0.415	**0.604**	0.594	0.576
4	**0.326**	**0.386**	**0.432**	0.381	**0.438**	**0.461**	0.624	**0.563**	**0.549**
8	**0.346**	**0.401**	**0.442**	**0.414**	**0.442**	**0.483**	**0.576**	**0.554**	**0.536**
16	**0.364**	**0.423**	**0.463**	**0.436**	**0.475**	**0.503**	**0.563**	**0.568**	**0.527**
32	0.298	0.348	**0.430**	0.381	0.382	0.432	0.635	0.615	0.568

B. *Performance on MovieLens-100K dataset*

The ratings of the MovieLens-100K dataset is divided into five different training/testing sets. We repeat our experiment with each set and average the results. In order to implement

Table IV: Performance Comparisons on MovieLens-1M Dataset using DBSCAN

# of clusters	F1@10			MAP			RMSE		
	User based	Item based	SVD	User based	Item based	SVD	User based	Item based	SVD
1 (Base)	0.314	0.376	0.421	0.384	0.415	0.443	0.615	0.584	0.558
11	0.296	0.348	0.418	**0.404**	**0.429**	**0.458**	**0.603**	0.613	0.571
14	**0.335**	**0.391**	**0.442**	0.368	0.383	**0.461**	**0.587**	**0.577**	**0.532**
25	**0.364**	**0.443**	**0.475**	**0.428**	**0.452**	**0.483**	**0.573**	**0.541**	**0.513**
31	0.305	**0.435**	**0.451**	0.362	**0.463**	**0.451**	0.636	**0.558**	**0.547**

k-means clustering, we set the value of k from 2 to 32 and for DBSCAN, following pairs of eps and $MinPts$: {1, 50}, {1, 75}, {2, 50}, {2, 75} are used which results in 20, 8, 27 and 12 clusters respectively. Since the number of ratings in MovieLens-100K is lesser than MovieLens-1M, therefore here we use smaller values for $MinPts$. We report the prediction accuracy of the recommendation algorithm in terms of MAE for k-means and DBSCAN method in Figures 4 and 5 respectively. Similar to the results for MovieLens-1M data, our model achieves best MAE values when the number of clusters are between 8 and 16 for k-means method (Fig. 4) and between 12 and 20 for DBSCAN method (Fig. 5). The other three evaluation metrics: F1@10, MAP, and RMSE are reported in Tables V and VI for k-means and DBSCAN clustering techniques respectively. As already discussed, the bold numbers indicate that its value has an improvement over the Base value. In Table V and Table VI, we can observe that in terms of the different evaluation metrics our clustering based approach performs far better than the Base results in most of the cases.

Table V: Performance Comparisons on MovieLens-100K Dataset using k-means

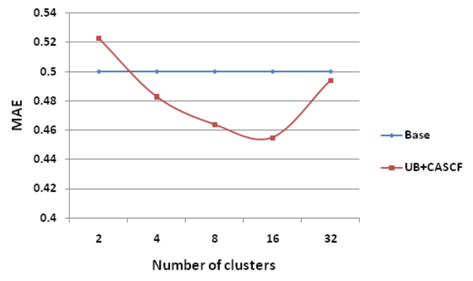
# of clusters	F1@10			MAP			RMSE		
	User based	Item based	SVD	User based	Item based	SVD	User based	Item based	SVD
1 (Base)	0.287	0.336	0.385	0.354	0.395	0.413	0.665	0.614	0.588
2	0.265	0.308	0.365	0.332	0.376	0.392	0.684	0.635	0.618
4	0.282	**0.352**	**0.402**	**0.376**	**0.418**	**0.426**	0.671	**0.593**	**0.572**
8	**0.317**	**0.368**	**0.416**	**0.396**	**0.442**	**0.448**	**0.642**	**0.582**	**0.566**
16	**0.338**	**0.384**	**0.433**	**0.413**	**0.432**	**0.453**	**0.631**	**0.562**	**0.537**
32	0.277	0.325	0.371	0.343	0.387	**0.422**	**0.653**	0.625	**0.576**

Table VI: Performance Comparisons on MovieLens-100K Dataset using DBSCAN

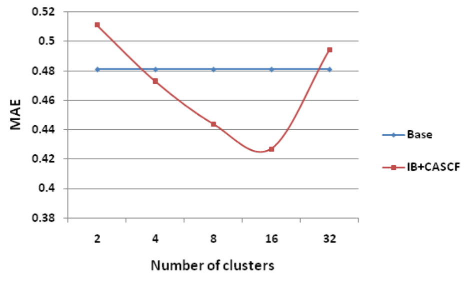
# of clusters	F1@10			MAP			RMSE		
	User based	Item based	SVD	User based	Item based	SVD	User based	Item based	SVD
1 (Base)	0.287	0.336	0.385	0.354	0.395	0.413	0.665	0.614	0.588
8	0.254	0.327	**0.394**	0.332	**0.415**	**0.428**	0.683	**0.603**	0.593
12	**0.305**	**0.364**	**0.417**	**0.375**	**0.437**	**0.453**	**0.646**	**0.575**	**0.556**
20	**0.337**	**0.394**	**0.423**	**0.396**	**0.461**	**0.474**	**0.639**	**0.563**	**0.538**
27	0.281	**0.341**	0.376	**0.367**	0.379	**0.432**	0.692	0.637	**0.571**

C. Scalability

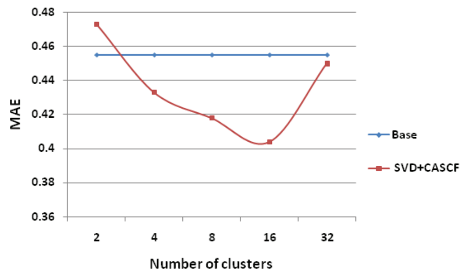
We report the runtime of our algorithm for MovieLens-1M and MovieLens-100K datasets in Fig. 6. Here, Base represents the recommendation time per user using the entire rating matrix R (without clustering). Our experiments are run on a computer with Core i3 - 2100 @ 3.10GHz x 4 CPU and 4 GB RAM. We compare the runtime of our clustering based model with the Base performance. From Fig. 6(a) and (b), we can clearly observe that the recommendation time reduces significantly when we cluster the original matrix and run the CF algorithms separately to the clusters. Moreover, note that, the recommendation time per user decrease as we increase the number of clusters.



(a) User-based



(b) Item-based



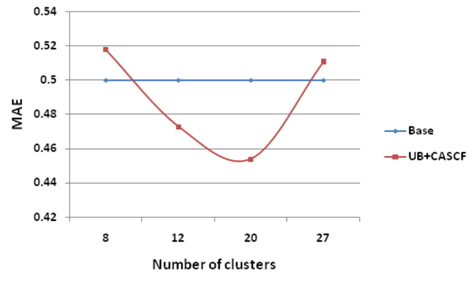
(c) SVD

Figure 4: Comparisons of the prediction performance on MovieLens-100K Dataset in terms of MAE by k-means method.

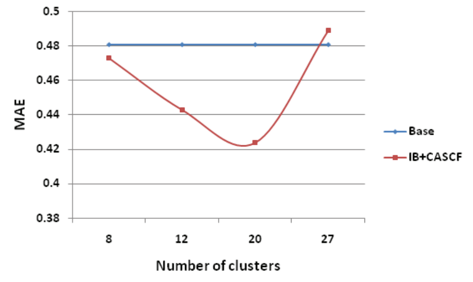
Analyzing the results of the experiments performed on the two MovieLens datasets, we can conclude that our recommendation model CASCF is efficient in reducing the runtime without sacrificing the recommendation quality much. This proves that our method is scalable and it can be used to deal with even bigger datasets with similar computing resources.

VI. Conclusion and Future Work

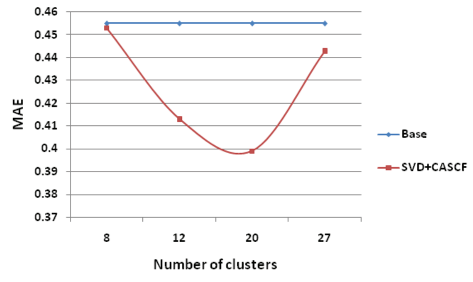
In this paper, we have presented a scalable context-based clustering algorithm that utilizes the contextual information available in the rating data to improve the recommendation accuracy. We have implemented data clustering techniques to partition the original user-item-rating into homogeneous clusters and then run CF algorithms separately on the clusters. We have conducted extensive experiments on two publicly available datasets MovieLens-1M and MovieLens-100K, and the experimental results show that our method enjoys higher recommendation accuracy over the existing CF methods as well as brings down the runtime considerably. The focus of our future work is to use other clustering and CF techniques in



(a) User-based

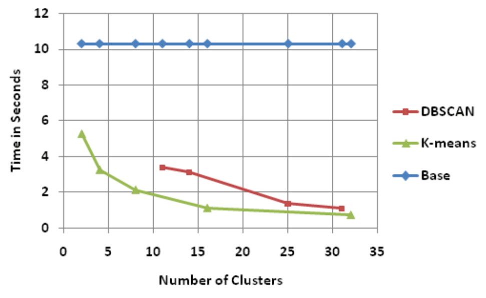


(b) Item-based

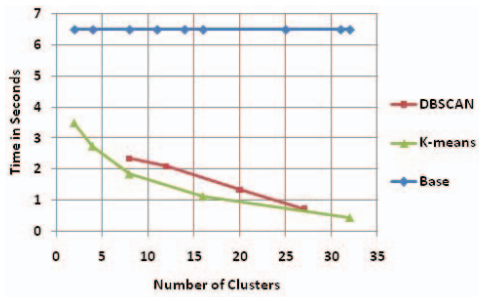


(c) SVD

Figure 5: Comparisons of the prediction performance on MovieLens-100K Dataset in terms of MAE by DBSCAN method.



(a) MovieLens-1M



(b) MovieLens-100K

Figure 6: Recommendation Time per User

order to generate much faster and accurate recommendations. How this can be leveraged is a matter of future research.

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*, vol. 17, no. 6, 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] 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.

[6] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Proceedings of the Fifth IEEE International Conference on Data Mining*. IEEE Computer Society, 2005.

[7] J. Han and M. Kamber, *Data Mining Concepts and Techniques*. Morgan Kaufmann Publishers, 2006.

[8] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[9] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 79–86.

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer Society*, vol. 42, no. 8, pp. 30–37, 2009.

[11] X. Liu and K. Aberer, "Soco: a social network aided context-aware recommender system," in *Proceedings of the 22nd international conference on World Wide Web (WWW' 13)*, 2013, pp. 781–791.

[12] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey," *Decision Support Systems*, vol. 74, no. C, pp. 12–32, 2015.

[13] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

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

[15] ——, "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.

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

[17] S. Vucetic and Z. Obradovic, "Collaborative filtering using a regression-based approach," *Knowledge and Information Systems*, vol. 7, no. 1, pp. 1–22, 2005.

[18] Z. Xiaoyao, L. Yonglong, S. Liping, and C. Fulong, "A new recommender system using context clustering based on matrix factorization techniques," *Chinese Journal of Electronics*, vol. 25, no. 2, pp. 334 – 340, 2016.

[19] E. Zhong, W. Fan, and Q. Yang, "Contextual collaborative filtering via hierarchical matrix factorization," in *Proceedings of the SIAM International Conference on Data Mining*, 2012.