Employing Sparsity Removal Approach and Fuzzy C-Means Clustering Technique on a Movie Recommendation System

Noor Ifada, Eko Hadi Prasetyo, Mula'ab
Informatics Department
University of Trunojoyo Madura
Bangkalan, Madura, Indonesia
noor.ifada@trunojoyo.ac.id, praztainfor@gmail.com, mulaab@trunojoyo.ac.id

***Abstract*—Collaborative Filtering (CF) approach has been used in many recommendation systems. Despite its popularity, CF faces several challenges such as data sparsity and scalability. In this paper, we propose a novel clustered item-based CF to solve both problems. To overcome the sparsity issue of rating data, we propose a novel sparsity removal approach that employs the combination of rating and movie genre similarities. To overcome the scalability issue, we apply the use the Fuzzy C-Means clustering technique to create groups of movies. Evaluating the proposed method on a real-world movie dataset, we show that our proposed method produces a dense rating data, is scalable for high dimensional data, and improves the recommendation quality of the traditional item-based CF method.**

Keywords*—*Collaborative Filtering, Sparsity, Scalability, Rating similarity, Genre similarity, Fuzzy C-Means Clustering

I. INTRODUCTION

Recommendation systems have been facilitating users in choosing items that they might like, out of a large set of items available on the web. In the websites that offer rating features, users are allowed to express their level of preference to items they have chosen. To sort the users' priorities, prediction models calculate the users' level of preferences to items they have not seen based on their rating history [1]. In other words, recommendation systems are trying to understand the users' needs and expectation, presenting a small set of items that suit their interest, and saving the users from having to spend too much time looking for them.

Collaborative Filtering (CF) is a very popular approach in recommendation systems [1-3]. It can be implemented based on either the memory-based or model-based approach. Yet, since the generated recommendations of the former have more understandable reasoning and explanation than the later [4], we use the memory-based approach as the general framework of this paper.

The memory-based approach consists of two models: user-based and item-based [4]. The main difference between these two is on how the similarities of rating patterns are employed to generate the recommendations. The user-based model assumes that a target user is influenced by users that have like-minded rating pattern with him and therefore users' similarities are used for generating the recommendations. On the other hand, the item-based model uses the items similarities since it assumes that a target user tends to choose items that are similarly rated to the ones he liked in the past. Given that the item-based model generally performs better than the user-based [2, 4], we implement the model to build our proposed method.

CF suffers from two common problems: sparsity and scalability. Sparsity is the condition where unobserved rating entries are a lot more dominant than the observed ones. In other words, the data has too many missing values. Meanwhile, scalability is the situation when the dimension size of the data causes expensive computational cost.

Sparsity is limiting the performance of CF since the sparse rating data means that users do not share a sufficient number of similar items and the system is unable to calculate the users or items similarities [2]. A recommendation system with such condition will result in poor recommendation performance [3]. Researchers have proposed to solve the sparsity problem by implementing clustering [5] or CBR [6] techniques. However, those methods focused on using the rating data only and did not consider the benefit of item attribute, which has shown to be beneficial in predicting item rating [7]. Scalability becomes an issue in CF when a recommendation system is struggling to scale to a large dataset [2, 8]. Clustering has been a practical solution to solve such problem by creating groups of items [9] or users [5, 6, 10].

In this paper, we are focusing our work on a movie recommendation system to address the two challenges of CF, by proposing a novel sparsity removal approach to fill in the unobserved rating entries and implementing a clustered item-based CF method to scale the system to a large data. At the first stage of our work, we solve the sparsity issue by combining the similarities of rating data and movie genre as the item attribute to calculate the values of unobserved rating entries, to generate a dense rating data. On the next stage, we employ the Fuzzy C-Means clustering technique, on the dense rating data, to create groups of movies. Followed by implementing the item-based CF method per cluster, i.e. clustered item-based CF method, in which the movie similarities calculation is decomposed to make it scalable for large dataset.

Experimental results on real-world dataset show that the recommendation quality of our proposed method outperforms the traditional item-based method. This finding establishes that the proposed sparsity removal approach efficiently solves the sparsity problem and assists the clustered item-based CF method in producing quality recommendations.

To the best of our knowledge, this is the first work on rating data that employs the combination of item/movie and genre similarities as a sparsity removal approach, to create a dense rating data. Our contributions are summarized as follows: (1) a novel sparsity removal approach that fills in the unobserved rating entries, handling the sparsity problem by employing the combination of rating and genre similarities, and (2) a clustered item-based CF method that implements the Fuzzy C-means clustering technique on an item-based CF method to deal with the scalability issue.

The organization of the rest of this paper is as follows. Section II describes the related work. Section III details the proposed clustered item-based CF method that consists of the sparsity removal approach and the Fuzzy C-Means clustering technique. Section IV presents the experimental results based on a real-world movie dataset. Section V concludes the paper.

II. Related Work

Collaborative Filtering (CF) approach has been successfully used in recommendation systems. The memory-based approach is an implementation of the CF approach which recommendation generation process is based on the calculation of similarities among users or items [4]. The work of Sarwar et al. [2] had analyzed various similarities technique used in item-based systems. They showed that the Adjusted Cosine similarity gives the best results in compared to others such as Cosine-based Similarity and Pearson Correlation Similarity.

The work of Gong [3] discussed the sparsity issue commonly occurs in CF due to the over dominance of unobserved rating entries. He proposed to solve the problem by employing the combination of user and user attribute similarities. Meanwhile, the recommendation generation process was conducted by implementing the combination of item and item attribute similarities. Though the work seemed promising, the method implemented a user-based model which ignored the fact that, in general, the item-based model performs better [2, 4].

The work of Chen et al. [7] also tried to solve the sparsity issue by calculating the item attribute similarity using the Jaccard similarity. They combined the prediction function with the time weight function to increase the accuracy of recommendations. This method, however, ignored the use of clustering technique when generating the recommendations and therefore it suffered from the scalability problem.

Alternatively, Kumar & Fan [6] focused on solving both the sparsity and scalability problems of CF. They used the Case-based Reasoning (CBR) to fill in the vacant cells in the rating matrix and implemented a clustering technique using SOM optimized with GA afterward. However, the CBR approach was a disadvantage in this case since it disregarded the benefit of the item attribute. Moreover, the SOM clustering can be outperformed by another technique, i.e. Fuzzy C-Means clustering [5].

In this paper, we propose a novel clustered item-based CF method that employs a novel sparsity removal approach which combines the rating and genre similarities, to solve the sparsity problem in rating data, and the Fuzzy C-Means clustering technique, to address the scalability problem in the generating recommendation stage.

III. Clustered Item-based CF Method

A. Overview

Our proposed method is focusing on handling the sparsity and scalability problems of CF approach. To address the sparsity issue, we propose a sparsity removal approach that estimates the unobserved rating entries by combining the similarities of rating and genre via relative weighting approach, results in a dense rating data. To address the scalability issue, we use the Fuzzy C-Means clustering technique to create groups of movies. This approach is to make sure that only a subset of movies needs to be considered during the stage of recommendation generation in which the item-based CF is implemented. The framework of our proposed method is illustrated in Fig. 1.

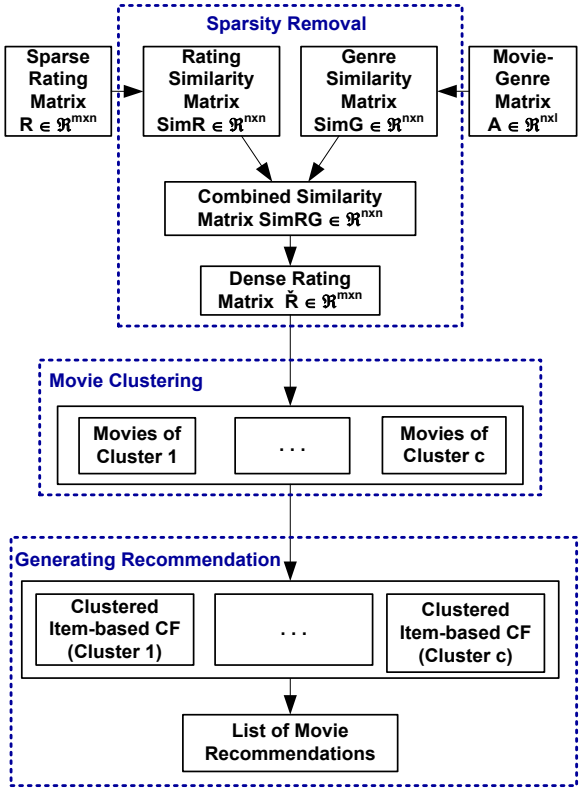

Fig. 1. Framework of the proposed clustered item-based CF method

B. Preliminaries

The rating data consists of observed entries which form the binary correlations between users and movies. The rating scores show the users level of preferences toward movies. The movie-genre data includes the list genres of each movie. The movie-genre binary scores show whether the movies are within the genres or not.

Let $U = \{u_1, u_2, u_3, \dots, u_m\}$ be the set of m users, $I = \{i_1, i_2, i_3, \dots, i_n\}$ be the set of n movies, and $G = \{g_1, g_2, g_3, \dots, g_l\}$ be the set of l genres. The binary correlation within rating data can be modeled as a rating matrix of $R \in \mathbb{R}^{m \times n}$ where each cell, r_{ui}, represents the rating of movie i given by user u. Whereas U_i indicates the set of users who have rated movie i. Meanwhile, the genre data can be modeled as a movie-genre matrix of $A \in \mathbb{R}^{n \times l}$ where each cell, a_{ig}, is set to 1 if movie i has genre g or 0 otherwise. Whereas G_i is the number of genres of movie i.

Fig. 2 and Fig. 3 respectively present the toy examples of a rating matrix $R \in \mathbb{R}^{3\times4}$ and a movie-genre matrix $A \in \mathbb{R}^{4\times5}$ where $U = \{u_1, u_2, u_3\}$, $I = \{i_1, i_2, i_3, i_4\}$, and $G = \{g_1, g_2, g_3, g_4, g_5\}$. Therefore, $U_1 = \{1\}$, $U_2 = \{1\}$, $U_3 = \{2\}$, and $U_4 = \{1, 3\}$. Meanwhile, $G_1 = 2$, $G_2 = 2$, $G_3 = 1$, and $G_4 = 1$.

		Movie			
		i_1	i_2	i_3	i_4
User	u_1	1	5	0	2
	u_2	0	0	2	0
	u_3	0	0	0	4

Fig. 2. Rating matrix R

		Genre				
		g_1	g_2	g_3	g_4	g_5
Movie	i_1	0	0	0	1	1
	i_2	0	1	1	0	0
	i_3	0	1	0	0	0
	i_4	0	0	0	0	1

Fig. 3. Movie-Genre matrix A

C. Sparsity Removal

This section explains the stage where we solve the sparsity problem by filling the unobserved cells in the rating matrix, creating a dense data. The ideas are that user tends to rate movies that are similar to other movies the user has liked [2], and that movies peer group usually are movies of similar genres [3, 4]. In this case, we can combine the similarities of rating and genre via relative weighting approach and use them to calculate the vacant rating.

There are a number of ways to compute the rating similarity between movies. In this paper, we use the Adjusted Cosine similarity due to its outperformance compared to others [2, 4]. The rating similarity between movie i and j is computed as follows:

$$SimR(i,j) = \frac{\sum_{u\in U_i\cap U_j}(r_{ui}-\mu_u)\cdot(r_{uj}-\mu_u)}{\sqrt{\sum_{u\in U_i\cap U_j}(r_{ui}-\mu_u)^2}\cdot\sqrt{\sum_{u\in U_i\cap U_j}(r_{uj}-\mu_u)^2}} \quad (1)$$

where r_{ui} is the rating of movie i by user u and μ_u is the average rating of user u.

The genre similarity between movie i and j is computed using the Jaccard similarity [7] as follows:

$$SimG(i,j) = \frac{\sum_{k\in l}(a_{ik}\cdot a_{jk})}{G_i+G_j-\sum_{k\in l}(a_{ik}\cdot a_{jk})} \quad (2)$$

where a_{ik} is a binary value of genre k on movie i, G_i is the number of genres of movie i, and l is the total number of genres.

The relative weighting to combine the similarities of rating and genre is computed as:

$$SimRG(i,j) = \omega \cdot SimI(i,j) + (1-\omega)\cdot SImG(i,j) \quad (3)$$

where ω is the relative weight coefficient that determines the importance of each similarity. In this case, the similarity is fully based on the rating similarity when $\omega = 1$ and that of genre when $\omega = 0$.

A list of top-J neighbors $Q_i(u)$ of movie i, for which the user u has specified ratings, is then sorted based on their similarity values and used to calculate the value of unrated cells $\hat{r}_{ui}$ of user u to movie i by using the following weighted average approach:

$$\hat{r}_{ui} = \frac{\sum_{j\in Q_i(u)} r_{uj}\cdot SimRG(i,j)}{\sum_{j\in Q(u)}|SimRG(i,j)|} \quad (4)$$

where $|Q_i(u)| \le J$. The complete sparsity removal algorithm is presented in Fig. 4.

Algorithm: Sparsity Removal

Input: Sparse Rating Matrix $R \in \mathbb{R}^{m\times n}$, Movie-Genre Matrix $A \in \mathbb{R}^{n\times l}$, relative weight coefficient ω, neighbor size J

Output: Dense Rating Matrix $\hat{R} \in \mathbb{R}^{m\times n}$

Process:

1. Generate the item similarity matrix $SimR \in \mathbb{R}^{n\times n}$ from R using Equation (1)
2. Generate the genre similarity matrix $SimG \in \mathbb{R}^{n\times n}$ from A using Equation (2)
3. Generate the weighted relative combination similarity matrix $SimRG \in \mathbb{R}^{n\times n}$ using Equation (3)
4. Based on $SimRG \in \mathbb{R}^{n\times n}$, generate the list of item neighbors $Q_i(u)$ of size J for each user u
5. Generate the Dense Rating matrix $\hat{R} \in \mathbb{R}^{m\times n}$ using Equation (4)

Fig. 4. Sparsity removal algorithm

Algorithm: Fuzzy C-Means Clustering

Input: Dense matrix $\hat{R}$, cluster size C, $w = 2$, $MaxIter$, ε, $t = 1$, $F(0) = 0$

Output: Cluster

Process:

1. Initialize $C = [\mu_{ij}]$ matrix, where $1 = \sum_{k=1}^{C}\mu_{ij}$
2. Calculate the centers of cluster k on attribute j:

   $V_{kj} = \frac{\sum_{i\in n}((\mu_{ik})^w\cdot X_{ij})}{\sum_{i\in n}(v_{ik})^w}$
3. Calculate Objective function at t^{th} iteration $F(t)$:

   $F(t) = \sum_{i\in n}\sum_{k\in C}\left(\left[\sum_{j\in m}(X_{ij}-V_{kj})^2\right]\cdot {\mu_{ik}}^w\right)$
4. Update $C = [\mu_{ij}]$ matrix:

   $\mu_{ij} = \left(\sum_{k\in C}\left(\frac{\|X_i-C_i\|}{\|X_i-C_i\|}\right)^{\frac{2}{w-1}}\right)^{-1}$
5. If $\|F(t)-F(t-1)\| < \varepsilon$ or $(t > MaxIter)$ then STOP Else $t \leftarrow t+1$, return to Step 2

Fig. 5. Fuzzy C-Means clustering algorithm

D. Movie Clustering using Fuzzy C-Means

This section details the stage where we solve the scalability problem by applying a clustering technique to reduce the next stage computation. After the sparsity removal stage, we now have a dense rating matrix $\hat{R} \in \mathbb{R}^{m \times n}$. We implement the Fuzzy C-Means clustering algorithm [11, 12] to cluster movies into several groups based on the rating data [9], as shown in Fig. 5.

E. Generating Recommendation using Clustered Item-based CF

After the clustering stage, we implement the clustered item-based CF to calculate movie similarities. Note that in the traditional item-based CF, the similarities are calculated for all movies, causing the scalability issue, while our clustered item-based CF conducts the calculations within the cluster only. The similarity calculation is formula formulated as follows:

$$Sim(i,j) = \frac{\sum_{u \in U_i \cap U_j} (\hat{r}_{ui} - \hat{\mu}_u) \cdot (\hat{r}_{uj} - \hat{\mu}_u)}{\sqrt{\sum_{u \in U_i \cap U_j} (\hat{r}_{ui} - \hat{\mu}_u)^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} (\hat{r}_{uj} - \hat{\mu}_u)^2}} \quad (5)$$

where $\hat{r}_{ui}$ is the rating of movie i by user u and $\hat{\mu}_u$ the average rating of user u from matrix $\hat{R}$.

Using the top-K nearest neighbors of movie i of user u, $T_i(u)$, the predicted rating p_{ui} of user u to movie i is calculated as:

$$p_{ui} = \frac{\sum_{j \in T_i(u)} \hat{r}_{uj} \cdot Sim(i,j)}{\sum_{j \in T_i(u)} |Sim(i,j)|} \quad (6)$$

where $|T_i(u)| \leq K$. Based on the predictions, the movies with high predicted ratings are recommended to each target user u as a top-N list of movie recommendations $Top(N)$.

IV. Empirical Analysis

A. Dataset and Experiment Design

For the experiments, we use the real-world MovieLens dataset from the GroupLens corpus[1] detailed in Table 1. It consists of 943 users, 1682 movies, and 100000 rating where each user has rated at least 20 movies. The rating is of five level of preferences from 1 to 5, indicating the lowest to the highest scores. The total number of available genres is 19.

TABLE I. Description of MovieLens Dataset

Data	Description
$u.data$	The rating data: user id, item id, rating, timestamp
$u.info$	The number of users, items, and ratings
$u.item$	Information about the items: movie id, movie title, release data, video release date, and movie's genres
$u.genre$	A list of the genres
$u.user$	Demographic information about the users: user id, age, gender, occupation, zip code
$u.occupation$	A list of the occupations

Note that, in this paper, the rating matrix R is generated from the $u.data$ data while the movie-genre matrix A is generated from the $u.item$ and $u.genre$ data.

Cross-validation is used to evaluate the results. We implement the 5-fold cross validation where each fold is randomly divided into an 80% training set D_{train} and a 20% test set D_{test}. The recommendation quality is then reported as the average results of all folds.

B. Evaluation Metrics

The recommendation task is to predict the top-N movies for all target users in D_{test}. For each target user u, the top-N list of recommended movies $Top(N)$ are compared to the ground truth movies GT in D_{test}.

In this paper, we use the F1-Score for measuring the recommendation quality of each target user u:

$$F1\text{-}Score(N) = \frac{2 \cdot (Precision(N) \cdot Recall(N))}{Precision(N) + Recall(N)} \quad (7)$$

where the Precision and Recall of each target user u are calculated as follows:

$$Precision(N) = 100 \cdot \frac{|Top(N) \cap GT|}{N} \quad (8)$$

$$Recall(N) = 100 \cdot \frac{|Top(N) \cap GT|}{|GT|} \quad (9)$$

The reported F1-Scores are the average scores of all users in the D_{test}.

C. Experiment Results

Impact of the Relative Weight Coefficient ω: The ω is the relative weight coefficient used to combine the rating and genre similarities (Equation (3)). Our experiments use a total grid of relative weight coefficients between 0 and 1 with an interval of 0.1, resulting $\omega = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Fig. 6 shows that the best result is achieved when $\omega = 0.9$. This value indicates that the rating similarity is 90% impacting the recommendation quality while the rest is determined by the genre similarity.

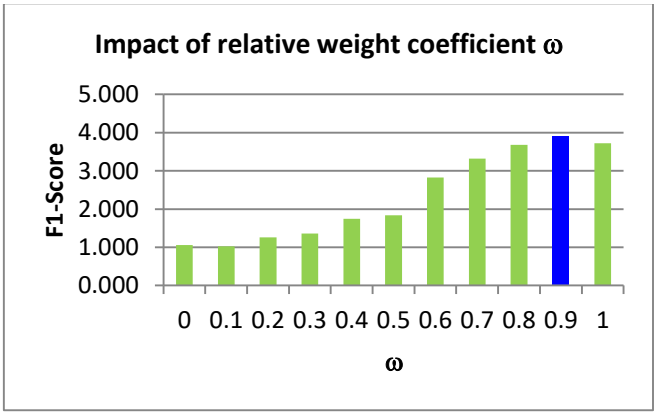

Fig. 6. Impact of the relative weight coefficient ω

[1] http://files.grouplens.org/datasets/movielens/ml-100k.zip

Impact of the Neighbor Size J: The J is the size of item neighbor to get $Q_i(u)$ used in Equation (4). We varies the item neighbor size as $J = \{5, 10, 15, 20, 25, 30, 40, 50, 100\}$. Results in Fig. 7 show that the best recommendation quality is achieved when $J = 30$ and is gradually decreasing when $J > 30$. The result indicates that larger neighbor size is not always linear to the recommendation quality.

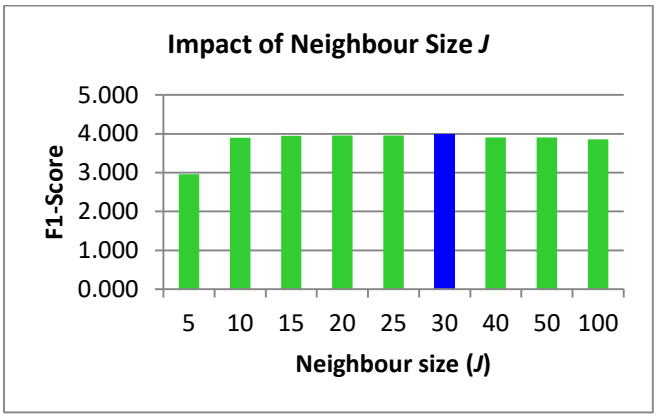

Fig. 7. Impact of neighbor size J

Sparsity Removal: Table 2 lists the comparison of rating matrix densities, of D_{train} of each fold, between the sparse rating matrix R and dense rating matrix $\hat{R}$. The statistics confirm that the sparsity removal approach has successfully generated dense rating matrices ($\hat{R}$) out of sparse matrices (R).

TABLE II. COMPARISON OF RATING MATRIX DENSITIES

Fold	Density of D_{train} (%)	
	Sparse Rating Matrix $R \in \mathbb{R}^{m \times n}$	Dense Rating Matrix $\hat{R} \in \mathbb{R}^{m \times n}$
1	5.04	100
2	5.04	100
3	5.04	100
4	5.04	100
5	5.04	100

Scalability of the Movie Clustering: The purpose of implementing movie clustering is to overcome the scalability issue in the item-based CF. We measure the running time of method at various number of cluster $C = \{1, 2, 3, 5, 7, 10, 15, 30, 50\}$. Note that item-based CF method without clustering technique is implemented when $C = 1$. Fig. 8 shows that the running time of movie clustering ($C > 1$) is a lot faster compared to that of the non-clustering ($C = 1$). This result points out that clustering technique implementation is able to solve the scalability problem.

Impact of the Number of Cluster C: The C is the number of cluster used when implementing the Fuzzy C-Means Clustering Algorithm (Fig. 5). We implement a various number of cluster $C = \{1, 2, 3, 5, 7, 10, 15, 30, 50\}$ and measure their resulted recommendation qualities. Fig. 9 shows that the highest F1-Score is achieved when at least $C = 2$. It is also worthwhile to notice that the worst result is achieved when $C = 1$, which indicates that the implementation of the clustering technique can also improve the recommendation quality, adding its advantage in solving the scalability issue.

Impact of the Neighbor Size K: The K is the size of item neighbor to get $T_i(u)$ used in Equation (6). Our experiments use a variety of item neighbor size $t = \{5, 10, 15, 20, 25, 30, 40, 50, 100\}$. Results in Fig. 10 show that the recommendation quality is linear to the neighbor size only until $K = 15$, i.e. the size when the best performance is achieved. When $K > 15$, the method performance is gradually decreasing. In other words, the larger the size of neighbor does not always give better performance since it possibly forcing items which actually are not similar enough to the target item to be in the neighborhood.

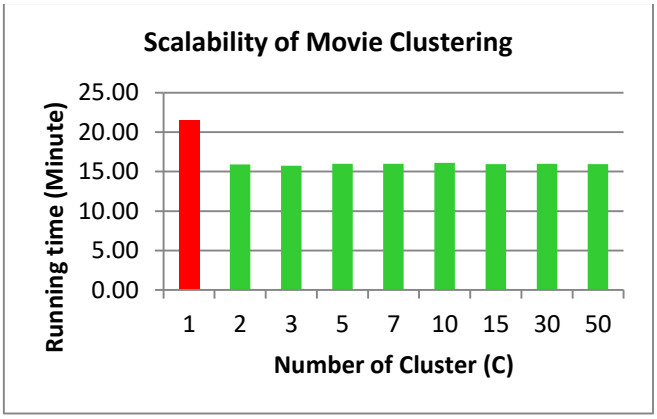

Fig. 8. Scalability of movie clustering

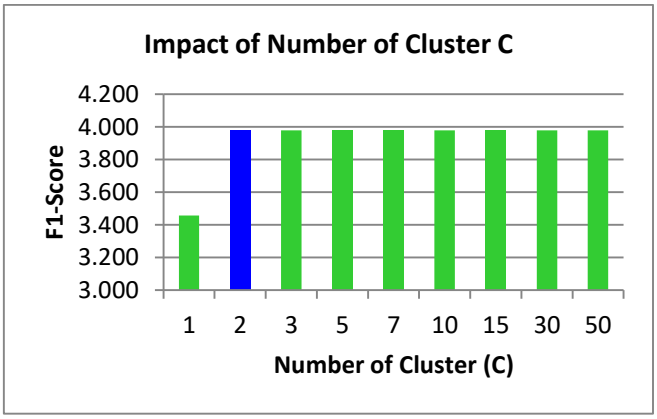

Fig. 9. Impact of number of cluster C

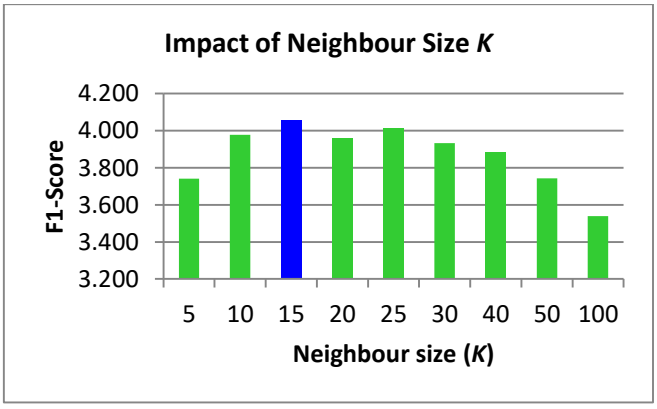

Fig. 10. Impact of neighbor size K

Recommendation Performance Comparison: For the purpose of comparison, we use the traditional item-based CF method [2] as our benchmark method. In this case, we compare its recommendation quality to that of ours, i.e. the clustered item-based CF. It is to be noted that, based on our experiments, the neighbor size of the traditional item-based CF is set as $K = 5$ to achieve its best performance. The comparison shown in Fig. 11 displays that our method outperforms the benchmark method on any top-N list of recommendations. This ascertains that our clustered item-based CF method that implements the sparsity removal approach and the Fuzzy C-means clustering technique on an item-based CF method can improve the quality of recommendations.

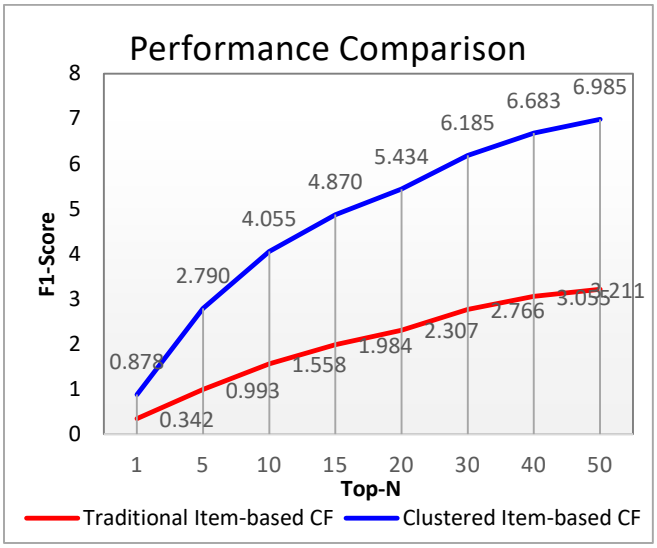

Fig. 11. Recommendation performance comparison

V. Conclusion and Future Work

We have presented our proposed clustered item-based CF method that implements the sparsity removal approach and the Fuzzy C-Means clustering technique to respectively deal with the sparsity and scalability problems. The sparsity removal approach employs the combination of rating and genre similarities to fill in the unobserved rating entries, creating a dense rating data. The Fuzzy C-Means clustering technique creates groups of movies so that the movie similarities used for generating the list of recommendations are calculated per cluster only, making it scalable for a large dataset.

The empirical analysis shows that our proposed method does not only solve the sparsity and scalability problems. Yet, it also outperforms its benchmark, i.e. the traditional item-based CF method. All of these confirm that the clustered item-based CF, which implements the sparsity removal approach and the Fuzzy C-means clustering technique, can improve the recommendation quality. For future work, we would like to investigate the potential of modifying the combination of rating and genre similarities for sparsity removal.

References

[1] J. A. Konstan and J. Riedl, "Recommender systems: from algorithms to user experience," *User Modeling and User-Adapted Interaction,* vol. 22, pp. 101-123, 2012.

[2] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in *The 10th International Conference on World Wide Web*, Hong Kong, 2001, pp. 285-295.

[3] S. Gong, "Employing User Attribute and Item Attribute to Enhance the Collaborative Filtering Recommendation," *Journal of Software,* vol. 4, pp. 883-890, 2009.

[4] C. C. Aggarwal, *Recommender Systems: The Textbook*: Springer, 2016.

[5] H. Koohi and K. Kiani, "User based Collaborative Filtering using fuzzy C-means," *Measurement,* vol. 91, pp. 134-139, 2016.

[6] N. P. Kumar and Z. Fan, "Hybrid User-Item Based Collaborative Filtering," *Procedia Computer Science,* vol. 60, pp. 1453-1461, 2015.

[7] Q. Chen, W. Li, and J. Liu, "Collaborative Filtering Algorithm Based on Item Attribute and Time Weight," in *The 2016 International Conference on Automatic Control and Information Engineering*, Hong Kong, 2016, pp. 12-15.

[8] S. L. Jadhav and M. P. Mali, "Pre-Recommendation Clustering and Review Based Approach for Collaborative Filtering Based Movie Recommendation," *International Journal of Information Technology and Computer Science,* vol. 8, pp. 72-80, 2016.

[9] S. Wei, N. Ye, S. Zhang, X. Huang, and J. Zhu, "Collaborative filtering recommendation algorithm based on item clustering and global similarity," in *The Fifth International Conference on Business Intelligence and Financial Engineering (BIFE)*, Lanzhou, China, 2012, pp. 69-72.

[10] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Recommender Systems for Large-scale E-Commerce: Scalable Neighborhood Formation Using Clustering," in *The fifth International Conference on Computer and Information Technology*, Dhaka, Bangladesh, 2002, pp. 291-324.

[11] R. Suganya and R. Shanthi, "Fuzzy c-means algorithm - A review," *International Journal of Scientific and Research Publications,* vol. 2, pp. 440-442, 2012.

[12] A. Shafeeq, "Optimal Clustering Using Modified Fuzzy C-Means Clustering Algorithm," in *International Conference on Information Technology & Society*, Kuala Lumpur, Malaysia, 2015 pp. 417-424.