Adaptive Collaborative Filtering Based on Scalable Clustering for Big Recommender Systems*

O-Joun Lee†, Min-Sung Hong†, and Jason J. Jung‡

School of Computer Engineering, Chung-Ang University
Heukseok-Dong, Dongjak-Gu, 156-756, KS013, Seoul, Korea
E-mail: {concerto34, minsung87, j3ung}@cau.ac.kr

Juhyun Shin

Department of Control and Measuring Robot Engineering, Chosun University
Seoseok-dong, Dong-gu, 501-759, KS008, Gwangju, Korea
E-mail: jhshinkr@chosun.ac.kr

Pankoo Kim

Department of Computer Engineering, Chosun University
Seoseok-dong, Dong-gu, 501-759, KS008, Gwangju, Korea
E-mail: pkkim@chosun.ac.kr

Abstract: The large amount of information that is currently being collected (the so-called "big data"), have resulted in model-based Collaborative Filtering (CF) methods to encountering limitations, e.g., the sparsity problem and the scalability problem. It is difficult for model-based CF methods to address the scalability-performance trade-off. Therefore, we propose a scalable clustering-based CF method in this paper that can help provide a balance by re-locating elements in the cluster model. The proposed method is evaluated by performing a comparison against existing methods in terms of measurements for the Mean Absolute Error (MAE) and response time to assess the performance and scalability. The experimental results show that the proposed method improves the MAE and the response time by 50.79% and 48.25%, respectively.

* This paper is significantly extended from an earlier version presented at the 3rd International Conference on Smart Media Applications in December 2014.

† These authors contributed equally to this work as the first author.

‡ Corresponding author.

Keywords: Big data; Recommender System; Adaptive System; Clustering-based Collaborative Filtering; Scalable System

1 Introduction

Collaborative filtering (CF) can be used to find a set of the relevant items that are assumed to be the most appropriate for a target user. Most CF approaches have analyzed user ratings to discover the various relationships between users, between items, and between users and items [1-4]. Since these methods mainly focus on collecting more user ratings (without integrating much external knowledge of a specific domain), we have realized that they have some fundamental limitations. The first limitation is the sparsity problem (which is also called the cold-start problem and first-users/items problem). If the density of the user rating matrix is too low and cannot represent users' preferences, the performance of the CF method will decrease [2].

The second limitation lies in the scalability problem (particularly, the big data issue). As the number of users and the number of items in a CF-based recommender system increase, the computation time to build user/item subsets exponentially increases [5].

A number of methods have been proposed to solve these problems, and these methods can be categorized into two main groups: model-based CF methods and hybrid CF methods [6-7]. Table 1 shows the advantages and disadvantages of these [8].

Table 1
Advantages and disadvantages of model-based CF and hybrid CF

	Advantages	Disadvantages
Model-based CF	· better addresses the sparsity, scalability and other problems · improves performance	· expensive model-building process · trade-off between performance and scalability
Hybrid CF	· improves performance · overcomes CF problems, such as sparsity and gray sheep	· increased complexity · needs external information that is usually not available

As can be seen, these cannot solve all the fundamental problems of the CF method, i.e., data sparsity problem and scalability problem. For the Model-based CF, these methods exhibit a trade-off between the predictive performance and the scalability since a reduced coverage results in a rating table that is relatively sparse. Furthermore, the cold-start problem still exist. The cold-start problem and the first-user/item problem are simply caused by the absence of 1data, so the model-based CF cannot be a fundamental solution.

Hence, the various hybrid CF methods that combine model-based CF and Content-Based Filtering (CBF) have been proposed in order to address these problems. However, most of these methods are too complicated to implement and external knowledge or data are required. If an excess of external data is necessary, then the applicable domain for the system will be restricted.

This paper proposes Adaptive Collaborative Filtering Based on Scalable Clustering (ACFSC) which is focusing on solving scalability problem by reducing time complexity to compose neighborhood. Also, it improves the data sparsity problem by making users' and items' feature vectors incrementally learning.

This method adopts three major policies: the use of minimal external data, combining, and re-locating. First, to maximize the adaptable domain of the system, we use the minimal external data, such as the user profile and the item metadata. Second, rating data and external data are combined to solve the cold-start problem and the first user/item problem. Third, newly-arrived rating data is used to re-locate users and items to form more appropriate clusters in order to solve the reduced coverage issues.

Based on the fore-mentioned policies, implementation of this method is composed of four parts, as shown in Fig. 1. The parts interact with each other following a cyclic architecture. It makes user ratings gradually improve the cluster models.

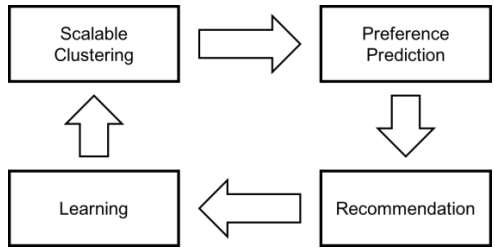

Figure 1
Conceptual Composition of the ACFSC

1) *Scalable clustering*: this step creates a cluster model for users/items that is based on feature vectors of them. It reduces the time complexity to produce user/item subsets. Feature vectors of these are composed by combining user profile data, item metadata, and user ratings. This step is intended to solve the cold-start and first-user/item problems.

2) *Recommendation*: this step recommends the top-N items that are selected according to the preference of the users who are included in the same user subsets. In order to minimize the system load, most tasks to create subsets proceed as in the first step.

3) *Preference prediction*: this step predicts users' missing preference information by using clusters of users and items to resolve the problems caused by the sparsity of the user rating matrix.

4) *Learning*: this step resolves the difficulty in quantifying the qualitative characteristic of the users and items through the use of learning feature vectors of users and items.

The remainder of this paper is organized as follows. In Section 2, we present an outline of related work. In Section 3, we introduce the Scalable clustering method to create the recommendation model as described in the first step above. In Section 4, the other steps are described to implement the Scalable Collaborative Filtering techniques. In Section 5, we evaluate the performance by comparing the results for the proposed method against those of others, and we discuss the results of the experiments. Finally, the conclusion provides a summary of the proposed algorithm and outlines future work.

2 Related Work

2.1 Model-based CF

In order to solve the scalability problem, various model-based CF methods based on machine learning or data mining models have been proposed, e.g., Bayesian belief nets, clustering models, latent semantic indexing, sparse factor analysis, and dimensional reduction. These use rating data to estimate or learn a model to predict users' preferences and can partially mitigate the scalability problem and the sparsity problem and improve the performance of the recommender systems. However, these have not yet been able to overcome the trade-off between performance and scalability and the cold-start problem still exists. The cold-start problem and the first-user/item problem are caused by the absence of data. Thus, model-based CF is not enough to solve them. For the moment, we introduce two representative approaches for these methods: clustering-based methods and dimensional reduction-based methods.

Clustering-based methods that use a cluster model to reduce the time complexity have been proposed. These methods build a cluster model by using correlations and similarities and use clusters that are built as subsets of users or items. Li and Murata [22] presented a CF method that is based on multi-dimensional clustering which involves clustering user/item profiles that are generated as background data, followed by clustering pruning and preference prediction through the weighted average of neighbors. This method has an advantage in that it maintains the balance in the performance of the recommendations, even when a growing diversity of items is handled. However, it still has the same limitations as the model-based CF method. Bellogin and Parapar [23] implemented in a normalized cut (N-Cut), a graph cut-based clustering solution, to facilitate the formation of similar user groups. Despite the improvement in performance over existing CF methods, the solution was unable to address the reduced coverage issue. Although

these kinds of methods are effective in solving the scalability problem of CF, they have two additional limitations. First, the costs of building cluster models are higher than for CF, so the models are hard to update dynamically. Second, since the target areas to compose the user/item subsets are restricted by the clusters, a reduced coverage problem occurs and the performance of the system deteriorates.

Dimensional reduction-based methods have also been proposed. These methods aim to cancel the noise in the rating matrices. Buried correlation information is generated between the users and the items underline. Zhong and Li [10] suggested a unified method that combines the latent and external features of users/items to ensure the accuracy in the predictive preference. A probabilistic latent semantic analysis is used to extract the latent features of the historical rating data. Luo et al. [11] implemented an incremental CF recommender system based on Regularized Matrix Factorization. This method supports incremental updates for the trained parameters as new ratings arrive. These methods partially solve the sparsity problem, including the cold-start problem and the first-user/item problem. However, the scalability problem becomes aggravated.

2.2 Combining Model-based CF and Content-based Filtering

The data sparsity problem in the model-based CF can be addressed through hybrid CF methods, which have been widely studied by combining model-based CF and Content-Based Filtering (CBF). These methods use external data to address the cold-start problem and the first-user/item problem of the model-based CF.

Parallel methods that use model-based CF and CBF at the same time have been suggested. These methods print out top-N items by integrating the results of model-based CF and CBF. Park et al. [13] suggested single-scaled hybrid filtering that uses a weight decided through an experiment, and this method showed a slight improvement in performance. Shen et al. [14] presented a hybrid filtering method that applied an optimized weight. This method used a simple learning algorithm based on user feedback to find more optimized weights. These methods are effective in solving the remaining problems for model-based CF. However, these methods do not provide improvements in terms of the trade-off between performance and the scalability, and also introduce two additional problems. First, they depend on the CBF at the initial time and for the first-user/item, so the overall performance can be comparatively decreased. Second, they need to dynamically update the weights as the data grows, but it is difficult to dynamically allocate time to update or optimize the system.

At the same time, other methods have been proposed to use external data to build corresponding models. These methods initially use external data when rating data does not yet exist, and the system functions by using rating data after the model has been built. Cho et al. [9] proposed a map-based personalized recommender system that uses Bayesian Networks built by an expert with a parameter that was learned from the dataset. Contextual information was collected (e.g., location,

time, weather), and the user request was provided a mobile device. Campos et al. [17] employed Bayesian networks to combine the characteristics of both CBF and CF and to generate more accurate recommendations by using probabilistic reasoning to compute the probability distribution over the expected rating. These methods are remarkable in solving the cold-start problem. However, it is hard to find external data that can be adapted to the appliance domain. If the external data is not adaptable, the performance of the system will severely decline.

On the other hand, other methods that integrate external data with user rating data have been presented. These methods use external data as the rating data or transform the external data into rating data. Bogers and Bosch [16] combined CF and CBF by using tags in social bookmarking websites. They examined how to incorporate tags and other metadata into a hybrid CBF/CF algorithm by overlapping the traditional user-based and item-based similarity measures with the tags. Hu and Pu [18] proposed a method that combines the personality characteristics of the users into traditional rating-based similarity computations for user-based collaborative filtering systems. Kim et al. [19] proposed a new approach to model users in a collaborative manner by using user-generated tags. This can be exploited in a recommender system by leveraging user-generated tags as preference indicators. These methods therefore effectively improve the sparsity problem. However, logistic evidence to transform external data as rating data are insufficient, and in addition, the scalability problem is not solved.

3 Scalable clustering with Data Streams

The scalable clustering method is conducted in three steps. In the first step, each cluster model for users and items is created according to their feature vectors. This step creates their clusters using the Expectation Maximization (EM) algorithm. The maximum likelihood is estimated based on the Gaussian-Bayesian Probabilistic Model and the cosine similarity, and it is formulated as

$$L_{C_i,x_j} = f(x_j, \mu_i, \sigma_i) \times \frac{1}{N_{C_i}} \sum_{l=1}^{N_{C_i}} \frac{R_j \cdot R_l}{\|R_j\| \, \|R_l\|}, \quad (1)$$

where C_i indicates the i-th cluster, x_j indicates the j-th element, μ_i and σ_i represent the average and the standard deviation of the elements in the i-th cluster C_i, and R_j is a rating vector of element j. N_{C_i} represents the number of elements included in the i-th cluster. $f(x_j, \mu_i, \sigma_i)$ indicates the probability that the j-th element is included in the i-th cluster. L_{C_i,x_j} indicates the maximum likelihood of the j-th element for the i-th cluster.

In the second step, the inter-cluster preferences between each of the user-clusters and item-clusters are estimated. This indicates the representative value of the preference of a particular user cluster for particular item-cluster, and it is formulated as

$$CP_{i,j} = \sum_{l}^{N_{uc_i}} \sum_{n}^{N_{cc_j}} W_{UC_i,u_l} \times W_{CC_{j,}c_n} \times R_{u_{l,}c_n}, \quad (2)$$

where UC_i and CC_j indicate the i-th user-cluster and the j-th item-cluster. u_l and c_n indicate the l-th user and the n-th item. $R_{u_{l,}c_n}$ indicates the rating for user u_l and item c_n, W_{UC_i,u_l} represents the likelihood of the user u_l for the user-cluster UC_i, and $W_{CC_{j,}c_n}$ is the likelihood of the item c_n for the item-cluster CC_j. $CP_{i,j}$ is the inter-cluster preference for the user-cluster UC_i for item-cluster CC_j.

In the third step, a user-item preference matrix is created. This matrix marks the preferences that are predicted using the proposed method, and it is different from the rating matrix that simply marks the rating score entered by the users. The prediction of the user-item preference is estimated according to the inter-cluster preferences and the user ratings, and it is formulated as

$$P(i_j|u_i)(predicted) = P(UC_l|u_i)P(IC_m|i_j)CP_{l,m}R_{i,j}, \quad (3)$$

where u_i and i_j indicate the i-th user and the j-th item, the user-cluster UC_l includes the u_i, and the item-cluster IC_m includes the i_j. In addition, $P(UC_l|u_i)$ is the probability that u_i belongs to the UC_l, and $P(IC_m|i_j)$ is the probability that i_j belongs to the IC_m. $CP_{l,m}$ is the inter- cluster preference of the UC_l for the IC_m. $P(c_j|u_i)(predicted)$ represents the predicted preference for the u_i corresponding to i_j.

4 Scalable Collaborative Filtering

The scalable collaborative filtering is composed of three modules. 1) The first module recommends the top-N items based on the user-item preference matrix, 2) the second module predicts the missed rating scores in the rating matrix, and 3) the third module enables the feature vectors of the users and the items that are learned by using feedback from the users.

4.1 Recommendation Module

This module chooses the top-N items in order to recommend a particular user. To improve scalability, a ranking of the item-clusters is referenced for a user-cluster that includes the user. This process consists of two steps.

1) In the first step, the item-clusters are ordered according to the inter-cluster preferences for the user-cluster that includes the target user.
2) In the second step, the items are added to the top-N list sequential search of the ordered item-clusters. The items that previously received a high preference are selected, and if no items received a preference in the search range, the search range is shifted to the next item-cluster.

4.2 Preference Prediction Module

This module predicts the missed rating scores through a hybrid preference prediction. The prediction is performed by using the weighted average of two widely used prediction methods: user-oriented prediction and item oriented prediction. This process is composed of three steps.

In the first step, similarity matrices are created for both the users and the items. Each component of the matrices has a similarity between the users or items, and the similarity is measured by using a cosine coefficient of the rating vectors.

In the second step, the two previously mentioned prediction methods make each of the prediction results by using the weighted average of the similarities. This step is formulated as

$$p_{a,m}(u) = \overline{R_a} + \frac{\sum_{n \in UC_a} (R_{n,m} - \overline{R_a}) \times uw_{a,n}}{\sum_{n \in UC_a} uw_{a,n}} \tag{4}$$

$$p_{a,m}(i) = \frac{\sum_{l \in CC_m} R_{a,l}\, cw_{m,l}}{\sum_{l \in CC_m} cw_{m,l}}, \tag{5}$$

where UC_a indicates the user-cluster that includes user a, and CC_m indicates the item-cluster that includes item m. $uw_{a,n}$ and $cw_{m,l}$ represent the similarities between users a and n and between items m and l. $\overline{R_a}$ represents the average of the ratings inserted by user a. $R_{n,m}$ and $R_{a,l}$ indicate ratings of user n for the item m and of user a for the item l, respectively. $p_{a,m}(u)$ and $p_{a,m}(i)$ indicate the predicted rating scores for user a and item l estimated by the user-oriented and item oriented method, respectively.

In the third step, the hybrid preference prediction method deducts the predicted rating score by combining the results of the preceding step according to the weighted average. The weighting is dynamically decided based on the consistency of the source datasets. This step is formulated as

$$p_{a,m}(hybrid) = \alpha \times p_{a,m}(u) + (1 - \alpha) \times p_{a,m}(i) \tag{6}$$

$$\alpha = \sigma(CC_m)/\sigma(UC_a) + \sigma(CC_m), \tag{7}$$

where $\sigma(UC_a)$ and $\sigma(CC_m)$ are standard deviations of the clusters that include user a and the item m, respectively. $p_{a,m}(hybrid)$ is the predicted rating score for user a and item m, and α is the dynamic weighting.

4.3 Learning Module

This module provides the feature vectors of users and items that are learned according to the rating data. This process consists of three steps: normalizing, learning, and re-locating.

In the first step, the newly-arrived ratings are normalized by the Gaussian Probability Model that is composed of the historical user ratings. Since standard points and measures of the rating scores vary across individuals, we need to unify these. This step is formulated as

$$preR_{a,m} = (F_{a,m} - \overrightarrow{F_a})/\sigma(F_a) \tag{8}$$

$$R_{a,m} = \begin{cases} 1, \ if \ preR_{a,m} \geq 10 \\ preR_{a,m}/20 + 0.5, \ if \ |preR_{a,m}| < 10 \\ 0, \ if \ preR_{a,m} \leq -10 \end{cases} \tag{9}$$

where $F_{a,m}$ represents a newly arrived rating for user a and item m, $\overrightarrow{F_a}$ and $\sigma(F_a)$ are the average and standard deviation of the historical ratings for user a, respectively, $preR_{a,m}$ indicates the preprocessed and non-normalized rating, and $R_{a,m}$ indicates the normalized rating.

In the second step, mutual complementary learning between the feature vector of the user who inserted the rating and the item which was rated by user is performed. This step uses the normalized rating as a weighting and is formulated as

$$\overrightarrow{UV_a} = R_{a,m} \times CV_m + (1 - R_{a,m}) \times pre\overrightarrow{UV_a} \tag{10}$$

$$\overrightarrow{CV_m} = R_{a,m} \times UV_a + (1 - R_{a,m}) \times pre\overrightarrow{CV_m}, \tag{11}$$

where $\overrightarrow{UV_a}$ and $\overrightarrow{CV_m}$ are feature vectors for user a and item m.

In the third step, the user who inserted the rating and the item which is rated by user are re-located to more suitable clusters. This step does not rebuild the cluster model, but rather just searches the clusters with greater likelihood following the change in the feature vectors.

5 Evaluation and Discussion

In this section, we present the results for two experiments that aimed to investigate two different issues: the cold-start problem and the scalability problem. In the first experiment, ACFSC is compared against existing methods to assess whether the proposed method provides better performance with a sparse rating table. Our conjecture is that the ACFSC should show better performance during the initial stage, and the gap between the ACFSC and the other methods should subsequently become smaller.

In the second experiment, ACFSC is compared against existing methods to verify whether these have an improved robustness for a high-load environment. Our hypothesis is that when a larger scale is obtained, this system should be more robust than the others. In summary, the experiments address the following research questions.

- Q1: Can ACFSC improve the performance of an initial system that has an extremely sparse rating matrix?
- Q2: Is ACFSC robust for use in the high-load environment of a large-scale system?

5.1 Experimental Environment

The experiments were conducted on representative service over 6 months based on the Ameba recommendation engine that was developed by the authors for a service providing wellness content. As a comparison group, we selected the following hybrid CF methods: Single-Scaled Hybrid Filtering (SSF) [13], Hybrid Recommendation System Based on Usage frequency (HFUF) [21], and Reinforcement Learning Algorithm Based Hybrid Filtering (RLHF) [14]. We then implemented a simulator based on Ameba for the subject methods.

The environment for the experiments consisted of an Android application and Windows server. A server was implemented with Apache Tomcat 7.0 and Windows 7. The DBMS of the server was MySQL 5.5. The integrated development environment (IDE) of the server was Visual Studio 2010, and the language used was Visual C++. A client was implemented on Android, and Eclipse Indigo was the IDE for the client with JAVA as the language for the Android SDK.

For the experimental settings, the subject user group was composed so that the age of the subjects was evenly distributed. This group consisted of the 150 people between 20 and 60 years of age who were randomly selected from students and faculty members of Dankook University. The subject item set was composed in such a way for the characteristics of the subjects to be spread over various areas. The set consisted of the 347 wellness content items, including cultural, tourism, and leisure content that were spread over the metropolitan areas of South Korea.

5.2 Improvement of Cold-start Problem

In order to assess the improvement in the cold-start problem, we measured the Means Absolute Error (MAE) as the number of users increased. The MAE is widely used to measure the performance of recommender systems [20, 21]. It is an average of the absolute deviations between a predicted ranking and an actual ranking for the recommended items. This measure is formulated as

$$MAE = \sum_{i=1}^{N} |r_i - p_i| / N, \tag{12}$$

where N is the number of items, and p_i and r_i represent the predicted ranking and the real ranking of i-th item. To obtain the experimental data, we make the subject user group, excluding the ordinary users, insert rating scores for all items that were recommended.

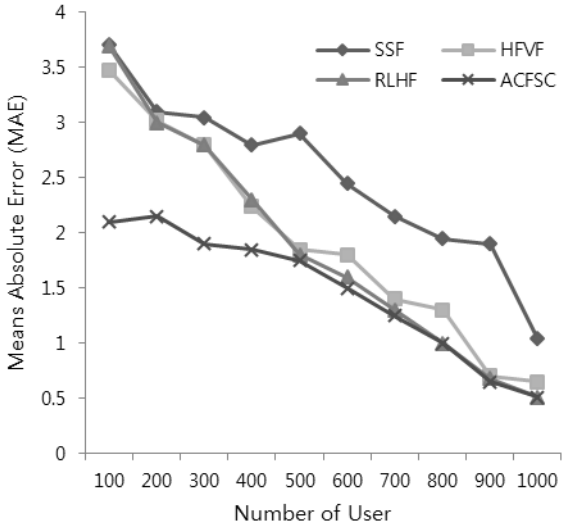

Figure 2
MAE of each Method according to users' number

Fig. 2 shows the MAE for each method with respect to the number of users. In addition, Table 2 represents the average, standard deviation, and range of the MAE for the selected methods.

As shown in Fig. 2, ACFSC exhibits its greatest performance during the initial time and also shows a comparatively steady performance after that. However, the gap between ACFSC and the other methods gradually decreases as the number of users increases. When the number of users is greater than 500, the methods show a performance similar to that of RLHF, which shows a comparatively higher performance than the other two methods. Also, when the number of users is higher than 1000, the proposed method exhibits a lower performance than RLHF.

As shown in Table 2, ACFSC presents a more stable performance than the other methods and also shows a slight improvement on average. ACFSC shows an improvement of 50.79% relative to RLHF over the given range. Also, the average MAE improved by 22.48%. In addition, we can make sure that the performance is stable for the proposed method in terms of the standard deviation.

Table 2
Average, Standard Deviation and Range of MAE for selected methods

	SSF	HFVF	RLHF	ACFSC
Average	2.516	1.956	1.899	1.472
Standard Deviation	0.778	0.907	1.038	0.562
Range	2.678	2.650	3.132	1.540

With respect to Q1, we can claim that the proposed method improves the cold-start problem. Also, since the problem of the first-user/item is caused by similar reasons as the cold-start problem, we can say that the proposed method probably can resolve the first-user/item problem as well. Nevertheless, we find that the proposed method is not able to improve the performance of the CF in an environment without the influence of cold-start problem. If we proceeded with the experiment for over 1000 users, the proposed method would not show an improvement in performance in this manner.

5.3 Robustness for Large Scale Service

In order to compare the robustness in a high-load environment, we measured the average response time as the number of users increased. The response time is a critical requirement for web-based services, such as a search engine and a recommender system. This method is defined as the amount of time that is required to provide a service. The gradient in the response time for the number of users can reveal how scalable a recommender system actually is. The average of the response time is then calculated based on the historical log of the server.

Fig. 3 represents the average for the response time of each method according to the number of users. Table 3 shows the average, standard deviation, and the range of the response time for the selected methods.

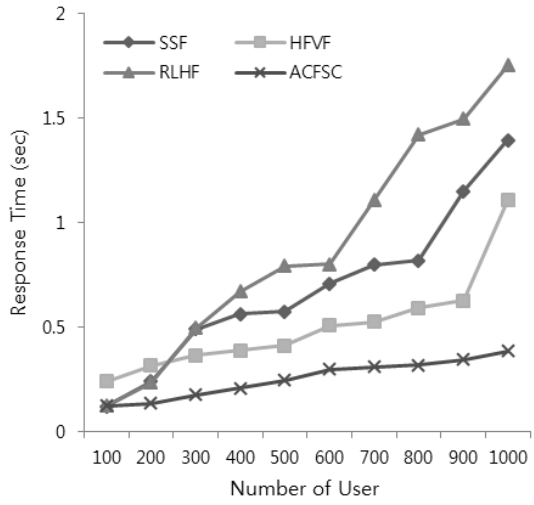

Figure 3
Average Response Time of each method according to the number of users

As shown in Fig. 3, we find that ACFSC shows a much shorter response time than the other methods by excluding the initial time. In addition, it exhibits a remarkably low and steady gradient. When the number of users is greater than 100, it has much shorter response time than HFVF, which shows a shorter response

time than the other two methods. Furthermore, when the number of users is greater than 900, ACFSC shows a stable gradient while the response time of HFVF starts to exponentially increase. In particular, as mentioned above, when the number of users is greater than 500, RLHF shows similar values as MAE with the proposed method while we can see that the response time for RLHF exponentially increases. On the other hand, ACFSC shows a linear increase at that time.

In addition, as shown in Table 3, ACFSC presents a more stable response time than the other methods, and also shows a remarkable improvement on average. Our method improves by 48.25% on average, as compared with HFVF. Also, the range improved by 74.18%, and furthermore, we can see clear improvement in the standard deviation.

Table 3
Average, Standard Deviation and Range of Response Time for selected methods

	SSF	HFVF	RLHF	ACFSC
Average	0.672	0.485	0.894	0.251
Standard Deviation	0.359	0.267	0.531	0.093
Range	1.131	0.980	1.587	0.253

With respect to Q2, we can make sure that the proposed method improves the robustness of the system for use in a large-scale service. At the initial time, ACFSC shows a similar response time as the others, but this is caused by the fact that all of the other methods have extremely sparse rating matrices at that time.

5.4 Discussion

As comparing with SSF and RLHF, the proposed method (ACFSC) outperforms the existing contents-based CF methods with respect to the data sparsity problem and the scalability problem. SSF and RLHF tried to improve the data sparsity problem by building CBF model based on metadata of items (i.e., genre, running time, and so on). However, as shown in Fig. 2, the metadata cannot improve the data sparsity problem enough since it can only reflect superficial features of items. The proposed method solved this issue based on mutual learning between the feature vectors of items and users. It makes an accuracy of the feature vectors gradually better.

Also, in terms of the scalability problem, ACFSC overcomes a limitation of the existing contents-based CF methods. As shown in Fig. 3, SSF and RLHF cannot improve the scalability problem since they estimate user preference by combining two independent filtering methods which are CF and CBF. On the other hand, the proposed method improved the scalability problem based on cluster models of items and users. It reduces time complexity to compose neighborhoods of items or users. In case of original CF methods, the time complexity to build model is

directly proportional to the number of items and users. However, in case of the proposed method, the time complexity to build model is directly proportional to the number of clusters of items and users.

Furthermore, as comparing with HFVF, ACFSC outperforms the existing rule-based CF methods with respect to the fore-mentioned two problems. HFVF used usage frequencies of items to improve the scalability problem. It extracted association rules from the usage frequency and applied them to reduce the number of target items to recommend (called as a coverage). As shown in Figs. 2 and 3, it can improve the scalability problem partially, but it cannot improve the data sparsity problem. Also, the proposed method reduces the coverages for both items and users. However, it improves both the data sparsity problem and scalability problem, since it reduces the coverages based on the cluster models built by preferences of users and improves them gradually.

Conclusion

The exponential increase in information (the so-called "Big Data paradigm") has caused difficulties in searching for desirable information and in addressing the increase in content. Therefore, the necessity of developing scalable recommender systems is on the rise. In this paper, we have proposed a highly scalable method (Adaptive Collaborative Filtering Based on Scalable Clustering) to guarantee stable performance and robustness of a recommender system for use in a large-scale system.

With respect to the two research questions that were previously mentioned, the proposed method performed outstanding improvements. With respect to Q1, we can claim that the proposed method improves system performance when there is a cold-start problem. Also, since the first-user/item problem is caused due to reasons similar to those of the cold-start problem, we can say that the proposed method can probably resolve the first-user/item problem, too. With respect to Q2, the proposed method was found to improve robustness for use in large-scale services.

Nevertheless, we also find that the proposed method cannot improve performance in a CF environment that does not have the influence of the cold-start problem. Our future work will therefore improve the clustering algorithm to improve system performance not only during the initial time.

Acknowledgement

This Research was supported by the Chung-Ang University Research Scholarship Grants in 2015. Also, this work was supported by the Human Resource Training Program for Regional Innovation and Creativity through the Ministry of Education and National Research Foundation of Korea (NRF-2014H1C1A1066494).

References

[1] Lee, O. J., Hong, M. S., Lee, W. J., and Lee, J. D.: "Scalable Collaborative Filtering Technique based on Scalable Clustering," Journal of Intelligence and Information Systems, Vol. 20, No. 2, 2014, pp. 73-92 (in Korean)

[2] Adomavicius, G., and Tuzhilin, A.: "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, 2005, pp. 734-749

[3] Pirasteh, P., Hwang, D., and Jung, J. J.: "Exploiting Matrix Factorization to Asymmetric User Similarities in Recommendation Systems," Knowledge-Based Systems, Vol. 83, 2015, pp. 51-57

[4] Pham, X. H., Nguyen, T. T., Jung, J. J., and Nguyen, N. T.: "<A, V>-Spear: A New Method for Expert Based Recommendation Systems," Cybernetics and Systems, Vol. 45, No. 2, 2014, pp. 165-179

[5] Bhosale, N. S., and Pande, S. S.: "A Survey on Recommendation System for Big Data Applications." Data Mining and Knowledge Engineering, Vol. 7, No. 1, 2015, pp. 42-44

[6] Lee, N., Jung, J. J., Selamat, A., and Hwang, D.: "Black-Box Testing of Practical Movie Recommendation Systems: A Comparative Study," Computer Science and Information Systems, Vol. 11, No. 1, 2014, pp. 241-249

[7] Ren, X., Lin, J., Yu, X., Khandelwal, U., Gu, Q., Wang, L., and Han, J.: "ClusCite: Effective Citation Recommendation by Information Network-based Clustering," Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 821-830

[8] Su, X., and Khoshgoftaar, T. M.: ″A Survey of Collaborative Filtering Techniques," Advances in Artificial Intelligence, Vol. 2009, 2009, Article 421425

[9] Cho, S.B., Hong, J. H., and Park, M. H.: "Location-based Recommendation System using Bayesian User's Preference Model in Mobile Devices," Proceeding of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC 2007), Lecture Notes in Computer Science, Vol. 4611, Hong Kong, China, July 11-13, 2007, pp. 1130-1139

[10] Zhong, J., and Li, X.: "Unified Collaborative Filtering Model based on Combination of Latent Features," Expert Systems with Applications, Vol. 37, No. 8, 2010, pp. 5666-5672

[11] Luo, X., Xia, Y., and Zhu, Q.: "Incremental Collaaborative Filtering Recommender based on Regularizad Matrix Factorization," Knowledge-Based Systems, Vol. 27, 2012, pp. 271-280

[12] Cacheda, F., Carneiro, V., Fernandez, D., and Formoso, V.: "Comparison of Collaborative Filtering Algorithms: Limitations of Current Techniques and Proposals for Scalable, High-Performance Recommender Systems," Journal ACM Transactions on the Web, Vol. 5, No. 1, 2011, Articl 2

[13] Park, K. S., Choi, J. M., and Lee, D. H.: "A Single-Scaled Hybrid Filtering Method for IPTV Program Recommendation," International Journal of Circuits, Systems and Signal Processing, No. 1, Vol. 4, 2010, pp. 161-168

[14] Shen, Y., Shin, H. C., Kim, D. G., Hong, Y. H., and Rhee, P. K.: "Reinforcement Learning Algorithm Based Hybrid Filtering Image Recommender System," Journal of the Institute of Webcasting, Internet and Telecommunication, Vol. 12, No. 3, 2012, pp. 75-81

[15] Tewari, A. S., Kumar, A., and Barman, A. G.: "Book Recommendation System Based on Combine Features of Content Based Filtering, Collaborative Filtering and Association Rule Mining," Proceedings of the 2014 IEEE International Advance Computing Conference (IACC) 2014, pp. 500-503

[16] Bogers, T., and Van Den Bosch, A.: "Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites," Proceedings of the ACM RecSys'09 Workshop on Recommender Systems & the Social Web, 2009, pp. 9-16

[17] Campos, L. M., Fernandez-Luna, J. M., Huete, J. F., and Rueda-morales, M. A.: "Combining Content-based and Collaborative Recommendations: a Hybrid Approach based on Bayesian Networks," International Journal of Approximate Reasoning, Vol. 51, No. 7, 2010, pp. 785-799

[18] Hu, R., and Pu, P.: "Using Personality Information in Collaborative Filtering for New Users," Proceedings of the 2nd ACM RecSys'10 Workshop on Recommender Systems & the Social Web, 2010, pp. 17-24

[19] Kim, H. N., Alkhaldi, A., El Saddik, A., and Jo, G. S.: "Collaborative User Modeling with User-generated Tags for Social Recommender Systems," Expert Systems with Applications, Vol. 38, No. 7, 2011, pp. 8488-8496

[20] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C.: "Eigentaste: a Constant Time Collaborative Filtering Algorithm," Information Retrieval, Vol. 4, No. 2, 2001, pp. 133-151

[21] Kim, Y., and Moon, S. B.: "A Study on Hybrid Recommendation System Based on Usage Frequency for Multimedia Contents," Journal of the Korean society for information management, Vol. 23, No. 3, 2006, pp. 91-125 (in Korean)

[22] Li, X., and Murata, T.: "Using Multidimensional Clustering-based Collaborative Filtering Approach Improving Recommendation Diversity," Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. Intell. Agent Technol., (2012) 169-174

[23] Bellogin, A., and Parapar, J.: "Using Graph Partitioning Techniques for Neighbor Selection in User-based Collaborative Filtering". Proceedings of the 6th ACM conference on Recommender systems, ACM, (2012) 213-216