A Distributed Hybrid Recommendation Framework to Address the New-User Cold-Start Problem

Jenq-Haur Wang and Yi-Hao Chen
Department of Computer Science and Information Engineering
National Taipei University of Technology
Taipei, Taiwan
jhwang@csie.ntut.edu.tw, augus790302@gmail.com

***Abstract*—With the development of electronic commerce, recommender system is becoming an important research topic. Existing methods of recommender systems such as collaborative filtering and content analysis are suitable for different cases. Not every recommender system is capable of handling all kinds of situations. In this paper, we bring forward a distributed hybrid recommendation framework to address the new-user cold-start problem based on user classification. First, current user characteristics, user context and operating records are used to classify the user type. Then, suitable recommendation algorithms are dynamically selected based on the current user type, and executed in parallel. Finally, the recommendation results are merged into a consolidated list. In the experiment on movie ratings dataset MovieLens, the proposed framework can enhance the accuracy of recommendation system. Also, a 100% coverage can be achieved in the case of new users. This shows the potential of the proposed framework in addressing new-user cold-start problem.**

Keywords—User classification; recommender systems; new-user cold-start problem

I. INTRODUCTION

With the development of electronic commerce on the Internet, recommender system is becoming an important research topic. Some algorithms consider the content that users are interested to learn the topical preferences, while others obtain potential clues from similar users who liked the same items, or similar items which have the same rating as the target item. These existing methods are all based on previous rating history. Thus, content-based can be used when there are enough content descriptions for the items, while collaborative filtering methods are commonly used when the system contains many existing items rated by many existing users. However, various methods might be suitable for different cases. Not every recommender system is capable of handling all kinds of situations. In some cases where we have little information about the user or the item or both, existing methods might not be able to provide suitable recommendation. For example, new users that have just registered in the system, or new items that have never been rated are among the typical *cold-start problem*.

Existing hybrid recommender systems combine the content-based similarity with collaborative filtering methods to recommend items even when there's no rating information about the new item. It's difficult to know which people like the item or not. However, in the case of new users, the issue becomes more challenging since there's no preference information about the user so that it's not possible to define similar users.

In this paper, we bring forward a distributed hybrid recommendation framework to address the new-user cold-start problem based on user classification. First, current user characteristics, user context and operating records are used to classify the user type. Then, suitable recommendation algorithms are dynamically selected based on the current user type. Finally, individual recommendation lists are then merged into a consolidated list as the final result. In the experiment, we use MovieLens 1M dataset as the experimental data which includes about 1,000,000 rating data of 3,900 movies marked by 6,040 users on the rating of 5-star scale. We conducted 10-fold cross validation, and the average of 20 runs was evaluated. As the experimental results show, the hybrid recommendation architecture can enhance the accuracy of recommendation system. Also, the cases of new users can be successfully recommended. This shows the potential of the proposed framework in addressing the new-user cold-start problem. Further investigation is needed to validate the performance in different datasets.

II. RELATED WORK

With the huge amount of Internet content, recommender system is becoming more important for satisfying user information need. The simplest method to recommend suitable items for users is information filtering based on demographics [1]. The assumption is people with similar gender, age or occupation tend to share common interests. To learn more information about individual user preferences from her previous interests, content-based recommendation techniques [2] are used based on content features or topics of the items previously purchased or viewed. With the huge amount of previous history of user purchasing or rating records,

978-1-4673-7211-4/15 $31.00 © 2015 IEEE
DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.307

collaborative filtering [3, 4] is among the most commonly used techniques. Generally, two types of collaborative filtering techniques can be used: user-based, item-based. User-based collaborative filtering discovers similar users who had similar purchasing or rating behaviors on items. People are likely to purchase what similar users bought. On the other hand, we can also utilize item-based collaborative filtering, in which items that get similar ratings from different users are likely to be good candidates. People are likely to purchase similar items with their previous interests.

However, when there are not enough existing system records of user ratings on items, collaborative filtering techniques might not be able give useful recommendations since correlation between user ratings and items are not robust with sparse data. Thus, hybrid recommender systems [5] combine collaborative filtering with content-based methods to exploit the text descriptions of items even when there're few ratings about the item.

The extreme cases of rating sparseness problem happen when new users register in the system [6, 7], or when new items enter the system [8, 9]. It's called the *cold-start problem* since it's not possible to give recommendations due to the lack of user ratings on items.

New-item cold-start problem has been addressed by Schein et al. [10]. Since recommendations are required for items with no rating, pure collaborative filtering alone cannot help in the new-item cold-start problem. Thus, content information can be used to make recommendation for new items whose content are similar to items that were already rated.

On the other hand, the new-user cold-start problem is more difficult [11] since there's no content information available as in new items. In Bobadilla et al. [11], a linear combination of simple user similarity measures such as overlap between rated items, number of items that users have rated in the same score, and a process of optimization based on neural network learning are used. Hyung [12] proposed a new similarity measure for collaborative filtering to alleviate the new user problem. Instead of Pearson's correlation and cosine, they proposed a new measure, which considers three factors of similarity: Proximity, Impact, and Popularity. In this paper, we classify user types based on demographic information and system experiences. Different recommender algorithms can be assigned based on user types.

Regarding the high computational complexity of recommender systems such as collaborative filtering, [13] proposed to implement user-based collaborative filtering based on Hadoop, a popular cloud computing platform. On the other hand, [14] scaled up item-based collaborative filtering algorithms with MapReduce, a popular functional programming model for large-scale distributed computing. In our proposed framework, we utilized similar ideas in dividing the different recommendation algorithms into separate tasks. Each task is independent to the others and will be dynamically invoked as scheduled by user type classification.

III. THE PROPOSED METHOD

In the proposed framework of hybrid recommender system, there are three major components: user classification, recommender algorithm, and result merging. The overall architecture of the proposed approach is illustrated in Fig. 1:

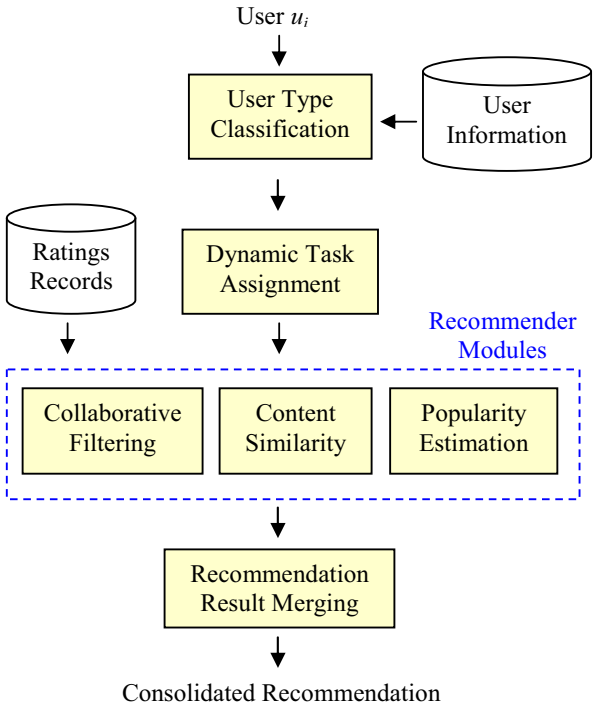

Fig. 1. The system architecture of the proposed approach of hybrid recommender system based on user classification.

First, given any user u_i, we classify the user into various types according to her characteristics. Then, according to different user type, we dispatch tasks to suitable recommender modules in parallel. Finally, given the recommendation results from different recommenders, we merge them into one consolidated list as the final recommendation for users. In the following subsections, we describe each component in details.

A. User Type Classification

One of the important modules in the proposed approach is user classification. To classify users based on user characteristics, we include user information such as demographic information and user experiences in the system. We define the type of user u_i by her demographic information $dem(u_i)$ and system experience $exp(u_i)$, that is, $UserType(u_i) = f(dem(u_i), exp(u_i))$. For demographic information, user age groups, gender, occupation, and address are collected for all users in the system. User age and gender can be possible clues of user preferences. For example, young teenagers can have higher interests in popular events or topics than elders. Male users could also have different interests from female. Analyzing user interests from all demographic groups give us more information on similar users for recommendation. For

user experiences, information such as user operation log, registration, and usage time are collected. For example, new users might have different information needs from experienced users. It's not possible to learn any preferences for new users. Ordinary recommendation algorithms such as pure collaborative filtering are not applicable due to the cold start user problem. Thus, we need different recommendation algorithms for new users.

During the system operation, user status could also be changing, such as age, occupation, and experience in the system. In order to timely reflect the changing user information need, our system can automatically detect the changes in user demographic information and usage habits. Then, users will be reclassified into more suitable user types. For example, as users become more experienced in the system, more usage logs can be utilized in recommendation. Thus, users can be reclassified as experienced users. Dynamically changing the user types can better reflect the real situations with more suitable recommendation methods.

B. Dynamic Task Assignment

After determining user type, instead of considering all factors into one integrated similarity measure between users, suitable recommendation modules will be dynamically invoked depending on user type. Then, the recommendation tasks will be dynamically assigned to the corresponding queue of the recommendation module. The idea is to consider only appropriate factors for each user type.

In this paper, the problem of dynamically dispatching tasks to recommendation modules and then merging the results is very similar to distributed retrieval [15] when searching multiple data sources. In the proposed framework, we consider each module as one possible source of information which can give partial recommendation based on its algorithm and capability. Thus, each recommendation module can be independently run for different user needs. In order to facilitate more flexible and efficient use of computing resources, each module can be run in parallel with different user profiles and requirements. Even if some recommendation does not give satisfactory results, the other modules could possibly compensate the results. From the perspective of resource distribution, the concept of dynamic task assignment is the same as distributed retrieval.

The dynamic distribution of recommendation tasks is crucial to system performance when the number of users and the number of recommendation modules increase.

C. Recommendation Modules

In the proposed approach, we incorporate the most popular and effective recommender algorithms, including: collaborative filtering, content similarity, and popularity estimation.

1) Collaborative Filtering

In collaborative filtering, we look for similar users that have the same rating behavior, or similar items that have the same ratings as the target item. Specifically, user-based collaborative filtering takes into account the previous rating history for all users. Items rated by users with similar rating behaviors can be recommended. The idea is that: people who bought similar items as the user might also buy those items.

For two users u_i and u_j, we define their similarity by their ratings record, that is, $sim_{user}(u_i, u_j)=sim(rating_{row}(u_i), rating_{row}(u_j))$, where $rating_{row}(u_i)$ is the vector of rating records of user u_i on all items. The notation of vector $rating_{row}(u_i)$ corresponds to the row vector in a user-item rating matrix. One possible simplification is to consider only the overlap between positively rated items. Let $N(u_i)$ be the set of positively rated items by user u_i, the similarity between users u_i and u_j can be defined by Jaccard coefficient as in Eq.(1):

$$sim_{user}(u_i, u_j) = \frac{|N(u_i) \bigcap N(u_j)|}{|N(u_i) \bigcup N(u_j)|} \quad (1)$$

After calculating the similarity between users, the interest of user u_i on an item d_j can be defined as in Eq.(2):

$$p(u_i, d_j) = \sum_{u_k \in S_k(u_i) \bigcap R(d_j)} sim_{user}(u_i, u_k) * r(u_k, d_j) \quad (2)$$

where $r(u_k, d_j)$ is the rating of user u_k on item d_j, $S_k(u_i)$ is the set of k-nearest neighbor users of user u_i, and $R(d_j)$ is the set of users that positively rated d_j.

Item-based collaborative filtering tries to identify item-to-item similarity in terms of their correlation in ratings. Items with similar ratings can be recommended. The idea is that: items that get similar ratings as the item might also be interesting to the user. This can be formalized as in Eq.(3):

$$sim_{item}(d_i, d_j) = \frac{|R(d_i) \bigcap R(d_j)|}{|R(d_i) \bigcup R(d_j)|} \quad (3)$$

where $R(d_j)$ is the set of users that positively rated d_j.

After calculating the similarity between items, the interest of user u_i on an item d_j can be defined as in Eq.(4):

$$p(u_i, d_j) = \sum_{d_k \in S_k(dj) \bigcap R(u_i)} p(u_i, d_k) * sim_{item}(d_k, d_j) \quad (4)$$

where $p(u_i, d_k)$ is the interest of user u_i on item d_k, $S_k(d_j)$ is the set of k-nearest neighbor items of item d_j, and $R(u_i)$ is the set of items that were positively rated by u_i.

For ordinary users, we have many ratings records on items, collaborative filtering methods can be helpful in recommending the most suitable items. For expert users, since they have much more ratings records, we can learn more about their preferences. Thus, user interest-based collaborative filtering can be used to recommend for expert users.

2) Content Similarity

In addition to similar users or item ratings, we could also use content similarity to find items that best suit users' needs.

For example, user that like to watch certain types of movies can be learned as his preference.

For items d_i and d_j, since only short item descriptions are available, typical document similarity measure such as cosine cannot be used. Thus, the content similarity can be defined by the overlap between their descriptions. For example, we can define the content similarity by Dice coefficient as in Eq.(5):

$$sim_{content}(d_i, d_j) = \frac{2*|desc(d_i) \bigcap desc(d_j)|}{|desc(d_i)| + |desc(d_j)|} \quad (5)$$

Since item content is helpful when we have little information on the user interest, it's mainly used to recommend for newbies.

3) Popularity Estimation

The third type of recommendation algorithm is based on popularity. In general, items that most people are interested are usually a safe choice when you don't know her particular preferences. In the case of new user, popularity becomes one possible solution to the cold start problem.

Instead of personalized recommendation that targets at user preferences, we include the top-ranked popular items as the candidates. Popularity of an item d_j can be defined as the number of users that have rated d_j. That is, $sim_{popularity}(d_j, d_j) = |R(d_i)|-|R(d_j)|$. The idea is that: the more users have rated the item, the more popular it is.

To get more refined estimation of item popularity, the rating records of an item by all users can be compared, that is, $sim_{popularity}(d_i, d_j)=sim(rating_{col}(d_i), rating_{col}(d_j))$, where $rating_{col}(d_j)$ is the vector of rating records of item d_j by all users. The notation of vector $rating_{col}(d_j)$ corresponds to the column vector in a user-item rating matrix.

One possible simplification is to count the number of users that give similar ratings. For example, the idea in [11] can be used, which includes the number of users that give exactly the same scores, the number of users that give different polarity, and the number of users that give ratings with differences of 1, 2, 3, and so on.

In this paper, we include variations of the popularity estimation methods by user demographics. For example, gender-related popular recommendation calculated popularity respectively on males and females. Similarly, occupation-related popular recommendation calculates popularity for each occupation separately. Age-related popularity considers different age groups such as children, teenagers, adults, and elders. Since popularity estimation is helpful when we know little about the user, it's mainly used to recommend for newbies.

In addition to the recommender algorithms, we implemented five similarity measures including: Pearson's correlation coefficient, Euclidean distance, cosine similarity, adjusted cosine similarity, and Spearman correlation.

D. Recommendation Result Merging

After related recommendation modules have completed the assigned tasks, the results will be merged into a consolidated recommendation list. As mentioned in the previous section, we utilized the idea of distributed retrieval [15] in which several data sources are searched. Generally, each module will give a list of recommended items with the corresponding scores. This module will do normalization on each list, reweighting, and then sorting to form a consolidated result.

In particular, different recommendation modules are given different weights regarding various user types. For example, popularity estimation is more important for new users, which are assigned higher weights than content-based methods if there are much fewer items rated by the user. Similarly, user preference-based collaborative filtering should have higher weights for experienced users since much more ratings are available to learn the preferences. In this paper, we focus on demographic information for popularity estimation, content-based, and collaborative filtering.

IV. EXPERIMENTS

To evaluate the effectiveness of the proposed hybrid recommender system, the MovieLens dataset was used in our experiment. There are four versions of MovieLens dataset, in which we used the MovieLens 1M dataset with about one million ratings from 6,040 users about 3,900 movies. The ratings are on a 5-star scale.

In order to compare the performance, we conducted leave-one-out [16] 10-fold cross validation to avoid the effects of specific data. Also, the experiment was run 20 times to avoid overfitting.

Before conducting the experiments, preprocessing tasks are needed to extract the user demographic information and experiences in the system. MovieLens dataset contains demographic information such as user age, occupation, and gender, but experiences information is not available. Thus, we further calculated the user experiences based on their total number of ratings since registration in the system. This can be obtained from the timestamp of each rating record. Here we dynamically divided users into three types according to their current experiences in the system: newbie, ordinary, and expert. From the observation of MovieLens dataset, we simply use the number of ratings to classify the three. Specifically, we consider newbies as the users whose number of ratings are less than or equal to 5, and experts as those with 20 or more ratings. Those users who give between 6-19 ratings are ordinary users.

A. Experiment on Accuracy of kNN Recommendation

To evaluate the accuracy of kNN recommendation, we utilized two evaluation metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). They are defined as in Eqs.(6) & (7):

$$MAE = \sqrt{\frac{\sum_{(u_i,d_j)\in T} |r(u_i,d_j) - r'(u_i,d_j)|}{|T|}} \quad (6)$$

$$RMSE = \sqrt{\frac{\sum_{(u_i,d_j)\in T}(r(u_i,d_j) - r'(u_i,d_j))^2}{|T|}} \quad (7)$$

where T is the set of test ratings, $r(u_i, d_j)$ is the actual rating of user u_i on item d_j, and $r'(u_i, d_j)$ is the predicted rating of user u_i on item d_j.

MAE measures the absolute differences between the actual rating and the predicted rating, where RMSE punishes more in the cases of inaccurate predictions. Thus, RMSE is stricter in evaluating the system performance.

To compare the proposed hybrid recommendation framework with traditional item-based and user-based collaborative filtering, we first evaluated the accuracy in k-Nearest Neighbor (kNN) recommendation when k = 20. The performance comparison with collaborative filtering in terms of MAE and RMSE is listed in Table I.

TABLE I. PERFORMANCE COMPARISON WITH COLLABORATIVE FILTERING (K=20)

Method	Performance Metrics	
	MAE	***RMSE***
Item-based	0.9582	1.2321
User-based	0.8532	1.1014
The Proposed Method	**0.8085**	**0.9370**

As shown in Table I, the proposed method outperforms either item-based or user-based collaborative filtering methods in both MAE and RMSE.

To further observe the effects of k on the performance, we further checked the performance for k=20 – 200, in the interval of 20 as shown in Figure 2 & Figure 3.

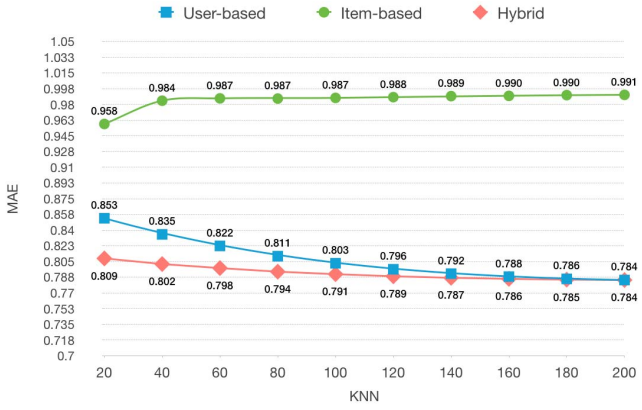

Fig. 2. The kNN recommendation performance comparison in terms of MAE.

As shown in Figure 2, the performance of item-based collaborative filtering becomes worse as the value of k increases, while the performance of user-based collaborative filtering improves as k increases. The proposed method consistently outperforms both item-based and user-based collaborative filtering methods in terms of MAE. The difference in MAE between user-based collaborative filtering and our proposed hybrid framework reduces as k increases. Next, we measured the performance in terms of RMSE.

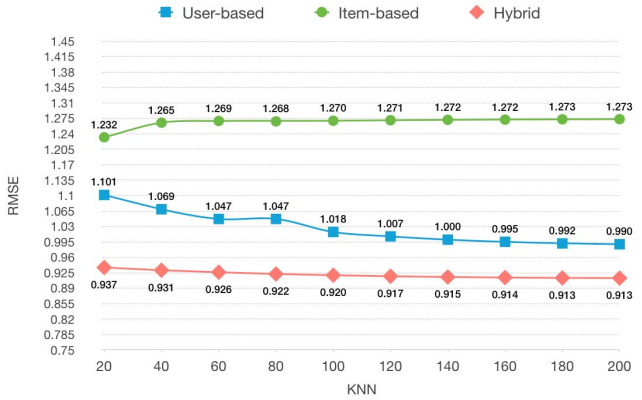

Fig. 3. The kNN recommendation performance comparison in terms of RMSE.

As shown in Figure 3, the performance of item-based collaborative filtering becomes worse as the value of k increases, while the performance of user-based collaborative filtering improves as k increases. The proposed method still consistently outperforms both item-based and user-based collaborative filtering methods in terms of RMSE. The difference in RMSE between user-based collaborative filtering and our proposed hybrid framework reduces as k increases, but the proposed method still holds the lead. We can observe a good recommendation performance even when there are fewer number of neighbors. As Figure 3 shows, the proposed hybrid recommendation framework is more stable in different number of nearest neighbors. This shows the advantage of the proposed recommendation framework which is less vulnerable to number of neighbor rating records.

B. Experiment on Coverage of Recommendation

In this experiment, we further checked the coverage of our proposed approach in recommendation. The coverage of the recommendation is defined as the ratio of successfully recommended users to all users. The comparison of coverage is shown in Table II.

TABLE II. COVERAGE COMPARISON WITH COLLABORATIVE FILTERING

Method	Coverage
Item-based	86%
User-based	98%
The Proposed Method	**100%**

As shown in Table II, the proposed method outperforms either item-based or user-based collaborative filtering methods in terms of coverage. This shows the advantage of our proposed hybrid framework in the case of new-user cold-start recommendation.

From the above observations, we can see the overall good performance of our proposed framework in accurately recommending items even in the case of new users. This was validated by the 100% coverage of the proposed approach compared to pure item-based and user-based collaborative filtering. The overall accuracy can be better than either item-based or user-based collaborative filtering with a much lower error at different values of k in kNN recommendation. This validates the practical use of our proposed approach in new-user cold-start situations.

V. CONCLUSION

In this paper, we have proposed a hybrid recommender system based on user classification. By dynamically assigning suitable recommender algorithms according to user type, our system can exploit the most appropriate algorithms. From the experimental results on MovieLens 1M dataset, our proposed method can achieve better performance in terms of RMSE or MAE. Also, among different types of users, our proposed method can successfully recommend in all cases. This shows the potential of our proposed method in real applications when there's cold start problem for new users. Further investigation is needed to evaluate the performance of our proposed approach in other datasets.

REFERENCES

[1] B. Krulwich, "Lifestyle Finder: Intelligent User Profiling using Large-Scale Demographic Data," Artificial Intelligence Magazine, 18(2), pp. 37-45, 1997.

[2] M. Pazzani, D. Billsus, "Content-based Recommendation Systems," The Adaptive Web, pp. 325-341, 2007.

[3] J.L. Herlocker, J.A. Konstan, J.T. Riedl, L.G. Terveen, "Evaluating Collaborative Filtering Recommender Systems," ACM Transactions on Information Systems, 22(1), pp. 5-53, 2004.

[4] X. Su, T.M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," Advances in Artificial Intelligence, pp. 1-19, 2009.

[5] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," User Modeling and User-Adpated Interaction, 12(4), pp. 331-370, 2002.

[6] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A.Konstan, J. Riedl, "Getting to Know You: Learning New Users Preferences in Recommender Systems," Proceedings of the International Conference on Intelligent Users Interfces (IUI 2002), pp. 127-134, 2002.

[7] A.M. Rashid, G. Karypis, K. Riedl, Learning Preferences of New Users in Recommender Systems: an Information Theoretic Approach," SIGKDD Explorations, 10(2), pp. 90-100, 2008.

[8] Y.J. Park, A. Tuzhilin, "The Long Tail of Recommender Systems and How to Leverage it," Proceedings of ACM Conference on Recommender Systems 2008, pp. 11-18, 2008.

[9] S.T. Park, W. Chu,"Pairwise Preference Regression for Cold-Start Recommendation," Proceedings of ACM Conference on Recommender Systems 2009, pp. 21-28, 2009.

[10] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, "Methods and Metrics afor Cold-Start Recommendation," Proceedings of the 25th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 2002), pp. 253-260, 2002.

[11] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal, "A Collaborative Filtering Approach to Mitigate the New Usaer Cold Start Problem," Knowledge-based Systems, 26, pp. 225-238, 2012.

[12] J.A. Hyung, "A New Similarity Measure for Collaborative Filtering to Alleviate the New User Cold-Starting Problem," Information Sciences, 178, pp. 37-51, 2008.

[13] Z.D. Zhao, M.S. Shang, "User-based Collaborative-Filtering Recommendation Algorithms on Hadoop," Proceedings of 2010 Third International Conference on Knowledge Discovery and Data Mining (WKDD 2010), pp. 478-481, 2010.

[14] J. Jiang, J. Lu, G. Zhang, G. Long, "Scaling-up Item-based Collaborative Filtering Recommendation Algorithm based on Hadoop," Proceedings of 2011 IEEE World Congress on Services, pp. 490-497, 2011.

[15] J. Callan, Distributed Information Retrieval, Advances in Information Retrieval, pp. 127-150, 2000.

[16] P. Massa, P. Avesani, "Trust-aware Recommender Systems," Proceedings of 2007 ACM Conference on Recommender Systems, pp.17-24, 2007.