A Recommendation System for Repetitively Purchasing Items in E-commerce Based on Collaborative Filtering and Association Rules

Yoon Kyoung Choi, Sung Kwon Kim

School of Computer Science and Engineering, Chung-Ang University, Republic of Korea
ykchoi@baewha.ac.kr, skkim@cau.ac.kr

Abstract

In this paper, we are to address the problem of item recommendations to users in shopping malls selling several different kinds of items, e.g., daily necessities such as cosmetics, detergent, and food ingredients. Most of current recommendation algorithms are developed for sites selling only one kind of items, e.g., music or movies. To devise efficient recommendation algorithms suitable for repetitively purchasing items, we give a method to implicitly assign ratings for these items by making use of repetitive purchase counts, and then use these ratings for the purpose of recommendation prediction with the help of user-based collaborative filtering and item-based collaborative filtering algorithms. We also propose associate item-based recommendation algorithm. Items are called associate items if they are frequently bought by users at the same time. If a user is to buy some item, it is reasonable to recommend some of its associate items. We implement user-based (item-based) collaborative filtering algorithm and associate item-based algorithm, and compare these three algorithms in view of the recommendation hit ratio, prediction performance, and recommendation coverage, along with computation time.

Keywords: Recommendation system, Collaborative filtering, e-commerce, Association rules

1 Introduction

With the growing purchase through online shopping malls, recommendation systems based on purchasing preferences of users have been widely used [1-5]. Recommendation systems predict the preferences of a user and recommend items that the user might like based on the predictions. One of the most widely used algorithms in recommendation systems is collaborative filtering [6-8]. Collaborative filtering recommendation systems produce a list of recommended items for a user based on similar decisions made by other users. These algorithms are devised usually for "homogeneous" shopping malls in the sense that they sell only one kind of items, e.g., music or movies.

Nowadays, as mobile devices are widely used and delivery services have been expanded, online purchases to buy daily necessities such as cosmetics, detergent, and food ingredient have increased. Since they are consumables, those purchases tend to be repetitive periodically and users usually buy several different items at the same time. For shopping malls selling daily necessities, it is difficult to directly apply collaborative filtering recommendation as they are "heterogeneous," i.e., they are selling several different kinds of items and as they usually record purchasing frequency and purchasing time of items by users but not their ratings for items.

In collaborative filtering algorithms, recommendations are usually made by the process of predicting target users' preferences based on the ratings of items that existing users gave in the past. For repetitively purchasing items, the ratings by users for items, needed by collaborative filtering algorithms, can be implicitly assigned by using the *repetitive purchase counts* or *purchase frequency*. With these implicitly assigned ratings, traditional collaborative filtering recommendation algorithms can be used to recommend items to users.

Since a set of items are purchased together at a same shopping basket, we can also consider the associate items as recommended items, where *associate items* of an item *i* are the items which are frequently bought at the same time in a same basket with item *i* by users. To find associate items we can use association rule mining algorithms [9-10].

Choi and Kim [11] present a user-based collaborative filtering algorithm by using *purchase frequency* as rating criteria for each repetitively purchasing item, where purchase frequency represents how often a user buys it repeatedly. Since for general e-commerce shopping malls this recommendation algorithm for repetitive items can be used as an auxiliary in addition to ordinary recommendation engines, our recommendation algorithm must be simpler and more efficient. Choi and Kim [12] simplify [11] by using *repetitive purchase counts* as rating criteria, instead of using purchase

*Corresponding Author: Sung Kwon Kim; E-mail: skkim@cau.ac.kr
DOI: 10.3966/160792642018111906006

frequency which needs many efforts to calculate. However, the repetitive purchase counts used in [12] have a large variation in each item and in each user, it is necessary to correct these deviations when calculating the similarity or recommendation prediction.

In this paper, we use *normalized repetitive purchase counts* as rating criteria, which is normalized by considering the size and distribution of the dataset. And we implement user-based and item-based collaborative filtering algorithms, and analyze the performance of these recommendation algorithms. We also propose *associate item-based* recommendation algorithm, and compare its performance with the two collaborative filtering algorithms. As experiment data, we use e-commerce sales data from *Happycoop Co*., one of Korean consumer cooperative companies [13].

In Section 2, we review related work on recommendation algorithms. Section 3 describes our recommendation algorithms for repetitively purchasing items composed of off-line preparation process and on-line recommendation process. In Section 4, we evaluate the performance of our recommendation algorithms, and we conclude in Section 5.

2 Related Work

Collaborative filtering (CF) is one of the most widely used techniques for building recommender systems [14]. For each user, *user-based CF* recommendation algorithm uses historical information to identify a neighborhood of people that in the past have exhibited similar behavior, and then analyze this neighborhood to identify new pieces of information that will be liked by the user [6].

The input data in the collaborative filtering algorithm is a collection of past purchase histories of m users and n items, and normally represented by $m \times n$ user- item matrix R.

An important step in the user-based CF algorithm is to compute the similarity between users. The most commonly used measures of user similarity are Pearson similarity, cosine similarity, and Jaccard similarity [3, 15].

The next step of the CF recommendation algorithm is to recommend the top-N items with the highest *prediction value* among the items purchased or rated by k similar users of the user u.

Item-based CF recommendation algorithm calculates the similarity between the various items, rather than between users, and then uses them to identify the set of items to be recommended [16-19]. In item-based CF algorithms, *cosine similarity* is commonly used as a standard metric. The cosine similary between two items a and b, viewed as the corresponding rating vectors $\vec{a}$ and $\vec{b}$ in user space, that is, as the corresponding column vectors in user- item matrix R, is defined as follows [8]:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \times |\vec{b}|} \quad \textbf{(1)}$$

Where $\vec{a} \cdot \vec{b}$ means the dot product of the two vectors, and $|\vec{a}|$ means the length of each vector. That is, $\vec{a} \cdot \vec{b}$ is calculated as $\sum_{k=1}^{m}(r_{ka} \times r_{kb})$, and $|\vec{a}|$ is calculated as $\sqrt{\sum_{k=1}^{m} {r_{ka}}^2}$.

The *frequent item-sets problem* is that of finding sets of items that appear in many of the same shopping baskets. This problem is often viewed as the discovery of "*association rules*" [10]. Finding association rules between a set of co-purchased items is one of the most commonly used data mining techniques for e-commerce [3].

The form of an association rule is $I \rightarrow j$, where I is a set of items and j is an item. The implication of this association rule is that if all of the items in I appear in some basket, then j is "likely" to appear in that basket as well.

The *support* for a set of items I is defined as a ratio of the number of baskets that contains I to the number of baskets. The notion of "likely" is formalized by defining the *confidence* of the rule $I \rightarrow j$ to be the ratio of the support for $I \cup \{j\}$ to the support for I. That is, the confidence c states that c% of baskets that contain I also contains j [10].

If we have found all item-sets that meet a support threshold s, then we can find within them all the association rules that have both high support and high confidence. To find *frequent* item-sets, an algorithm known as "*A-Priori*" is often used. *Apriori* is proposed by [9] and it is the most classic and widely used algorithm from which many variants have been developed [20-23]. For each item in frequent item-sets, we call each other an *associate item*.

Recommendation algorithms for periodically repetitive purchasing shopping malls were first implemented by Choi and Kim [11]. Once a new purchase occurs, the recommendation system in [11] updates *purchase frequency* of the corresponding item by off-line, which is followed by re-calculating a *favorite-items set* for each user, and re-calculating *user similarity* based on the favorite-items set. The *favorite-items set* is defined as the set of frequently purchased items for that user. Whenever a user logs in, items in *favorite-items sets* of other users who have similar purchase patterns are recommended using a user-based collaborative filtering method. They also implemented *automatic selection system*, which selects and recommends items that it is time to purchase by examining purchase frequency of candidate items. This algorithm recommends only items which have been purchased several times repeatedly in the past by the user.

Generally, in e-commerce shopping malls, only some items are purchased repeatedly, so the algorithm of [11] can be used as an *auxiliary* to general recommendation engine. To do this, the recommendation algorithm should be simpler and more efficient. Choi and Kim[12] simplified and improved the user-based CF recommendation algorithm of [11] by using *repetitive purchase counts* as rating criteria, instead of using purchase frequency which needs many efforts to calculate.

3 Recommendation System for Repetitively Purchasing Items

A diagram of our recommendation system proposed is shown in Figure 1, consisted of two parts: online and offline processes. When a new purchase occurs, since the user-item matrix is changed, the user-similarity etc. must also be changed. Therefore, such data processing needs to be performed periodically, as a pre-computing task. This process is described in the *offline* box in Figure 1.

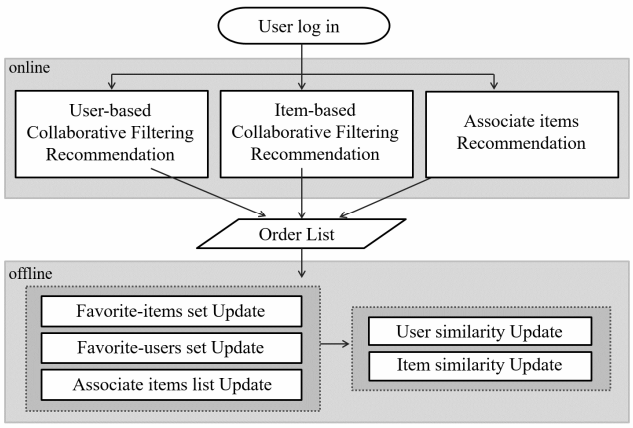

Figure 1. Proposed recommendation system

Each time a user logs-in, our system recommends items in real time using three recommendation algorithms. This process is described in the *online* box.

In this section, we first describe how to implicitly score a rating on items that are purchased repeatedly. We then describe the offline process and online process.

3.1 Rating Scores for Repetitively PurChasing Items

In our recommendation, we consider only items that were purchased more than C times in the past by users (C=5, for example). For each item i, we calculate the *score(u i)* for each user u:

$$score(u,i) = \frac{repetitivePurchaseCounts(u,i)}{duration(u,i)} \quad \textbf{(2)}$$

where *repetitivePurchaseCounts(u,i)* is the number of purchases of item i by user u, and *duration(u,i)* is the time interval between the first and the last purchasing time of item i by user u. In our experiments, we set the basic unit of duration as three months, so that *score(u,i)* is the number of repetitive purchases of item i during three months, averagely, by user u.

We normalize the scores so that they range from 1 to 5, and assign the *normalized score*, *scoreN*, as the rating for item i by user u. These are the implicitly assigned ratings mentioned in section 1, which will be used in collaborative filtering algorithms later. To convert the scores to the normalized scores between 1 and 5, we need to consider the distribution of *score(u, i)* values in the dataset and the overall maximum and minimum values of the *score(u, i)*.

In our dataset, the maximum of the scores is 22.7 and the minimum is 1. Figure 2 depicts the distribution of the *score* values for all users and items in our dataset, and we can observe the percentage of scores greater than 10 is very low (less than 3 %). Therefore, we set the *score* values of 10 or larger as 10. So, we are to convert the scores between 1 and 10 to the normalized scores between 1 and 5. Considering the distribution of the scores shown in Figure 2, we choose a quadratic function instead of a linear function for the normalization:

$$scoreN(u,i) = a \times score(u,i)^2 + b \times score(u,i) + c \quad \textbf{(3)}$$

where *a, b and c* are constants.

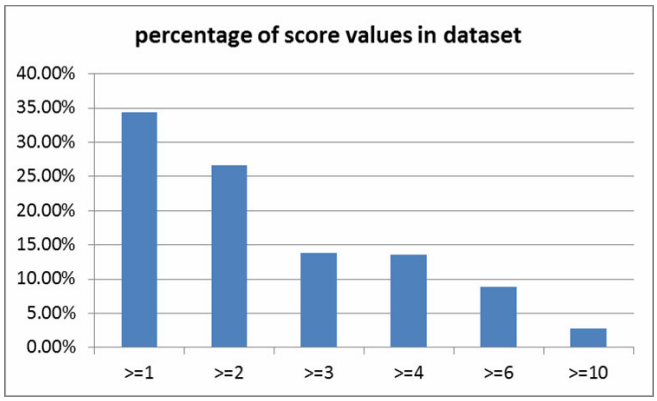

Figure 2. Percentage of *score* values for users and items overall in our dataset

To select the three constants by regression analysis to properly assign values while considering the proportion of the original score values, we determine to convert the score of 1 to the normalized score of 1, the score of 2 to the normalized score of 2, and the score of 10 to the normalized score of 5. After fixing three points at (1, 1), (2, 2), and (10, 5), we can calculate the three constants to obtain a = -5/72, b = 87/72, and c = -10/72 of the quadratic function of Eq. 3. Our normalized *scoreN* function is as followings:

$$scoreN(u,i) = \frac{-5 \times score(u,i)^2 + 87 \times score(u,i) - 10}{72} \quad \textbf{(4)}$$

3.2 Computing the Similarity

A critical step in a user-based (or item-based) collaborative filtering algorithm is to compute the similarity between users (or items) and then to select the most similar users (or similar items).

Of the many different ways to compute similarity, in our experiment, we use *Jaccard similarity* measure between two users or between two items, and we compare its result with the baseline algorithm using the *cosine similarity*.

Jaccard similarity between two sets *s1* and *s2* is defined by the number of elements in the intersection of the two sets divided by the number of elements in the union of them.

$$sim(s1,s2) = \frac{|s1 \cap s2|}{|s1 \cup s2|} \quad \textbf{(5)}$$

Favorite-items set and favorite-users set. We define *favorite-items set*, *Fi(u)*, as the set of frequently purchased items for each user *u*, which is the set of repetitively purchased items, purchased more than *C* times by user *u* (*C*=5 or 10, for example)[12]. We also define *favorite-users set*, *Fu(i)*, as the set of users who purchased item *i* many times, i.e., more than *C* times.

After computing the favorite-items set from the dataset, we can compute the favorite-users sets in the following way:

```
for each user u
    for each item i in Fi(u)
        add u into Fu(i)
```

We limit the number of elements in *Fi(u)* to *K* (*K*=30, in our experiment) in the order of *scoreN(u,i)* values, in other words, *Fi(u)* contains top-*K* items in terms of *scoreN(u,i)*. Since we treat repetitively purchasing items only, it is reasonable to maintain and use favorite items only at the recommendation process.

User similarity and item similarity. We calculate user similarity *UserSim(u1, u2)* for all pairs of users *u1* and *u2* by Eq. 6, with *Jaccard similarity* measure:

$$UserSim(u1,u2) = \frac{|Fi(u1) \cap Fi(u2)|}{|Fi(u1) \cup Fi(u2)|} \quad \textbf{(6)}$$

where *Fi(u)* is favorite-items set for user *u*.

Jaccard similarity considers primarily the number of common items in *Fi(u1)* and *Fi(u2)*, regardless of the scores of those items. Since we limit the number of elements to be contained in *Fi(u)* to *K*, and since we consider only the items that were purchased in the past more than *C* times repetitively, this similarity has a significant meaning. In the aspect of running time, it takes shorter time than computing cosine similarity or Pearson's similarity.

We calculate item similarity *ItemSim(i1, i2)* for all pairs of items *i1* and *i2*, by Eq. 7, with *Jaccard similarity* measure:

$$ItemSim(i1,i2) = \frac{|Fu(i1) \cap Fu(i2)|}{|Fu(i1) \cup Fu(i2)|} \quad \textbf{(7)}$$

where *Fu(i)* is favorite-users set for item *i*.

3.3 Computing Associate Items

Associate items of an item *i* are the items which are frequently bought at the same time with item *i* by users, and an *associate items list* for each item *i* contains top-*l* most frequent associate items (*l* to be explained later). To find associate items we employ *Apriori* algorithm, which is one of the association rule mining algorithms [10].

The Apriori algorithm progresses in multiple stages until all of those concurrently purchased *frequent item-sets* are obtained. Considering the execution time, however, we only carry out two stages and find the *frequent item-pairs*, that is, *associate item-pairs*.

In the first stage, every shopping basket is examined and the *purchase count* is increased by one for each of the items in the basket (see Figure 3). A one-dimensional array is used to store the purchase counts. In the second stage, the shopping baskets are re-examined, and the items having the purchase counts greater than *s*, a *support threshold* value, are found out. For each pair of those items obtained, we increment *freqCount* of the pair by one in the item-item matrix *freqCount*. The value of *freqCount* is the number of times the item pair appeared in a same basket in the past.

```
first stage
  for each basket
    for each item in the basket
      add 1 to its purchase_count (into an array to store counts)

second stage
  for each basket
    find frequent items, such that purchase_count > s
    for each frequent item-pairs
      add 1 to its freqCount (into a matrix to store counts)
```

Figure 3. Two-pass *Apriori* algorithm

The value of *s* should be set relatively large in order to reduce the execution time, and in general, *s* is set to 1% of the number of baskets [10].

For each item *i*, its *associate items list* is obtained by finding top-*l* frequently bought associate items of *i*, that is, top-*l* items in order of *freqCount* value of item-pair.

3.4 Off-line Preparation Process

Once a new purchase occurs, since the score of corresponding user-item is changed, *favorite-items set* for each user and *favorite-users set* and *associate items list* for each item are all updated *off-line* at a regular basis (see the lower bottom in Figure 1). Then, we recalculate *user similarity* for all pairs of users, and

recalculate *item similarity* for all pairs of items. Also, for each user u, we update *l-nearest neighboring users set, NeighborU(u)*, which contains l similar users in order of user similarity value with u, *UserSim(u, v)*, $v \in U$. Similarly, we update *l-nearest neighboring items set, NeighborI(i)* for each item i, which contains l similar items with i in order of item similarity value, *ItemSim(i, j)*, $j \in I$. U is the set of all users, and I is the set of all items.

3.5 On-line Recommendation Process

As shown in the upper side of Figure 1, we implement recommendation algorithms in three ways: user-based collaborative filtering recommendation, item-based collaborative filtering recommendation, and associate items recommendation.

User-based collaborative filtering recommendation. Whenever a user u logs in, a specific number (= t) of items are recommended in order of prediction value, *predict(u, i)*, $i \in I$, by the collaborative filtering algorithm. For each user *u2* contained in the *l*-nearest neighboring users set, *NeighborU(u)* (= set *N)*, we compute *predict(u, i)* for each item i contained in the favorite-items set of *u2*, *Fi(u2)*, as in Eq. 8. Then, out of the items that user u have never purchased, we recommend the top-t items in the order of prediction values.

$$predict(u,i) = \frac{\sum_{u2 \in N} (UserSim(u,u2) \times scoreN(u2,i))}{\sum_{u2 \in N} UserSim(u,u2)} \quad \textbf{(8)}$$

Since we calculate prediction values for the items only in the favorite-items set of *l*-neighboring users, our algorithm runs faster than general collaborative filtering methods.

Item-based collaborative filtering recommendation. To seek another top-t recommending items for each user u, we examine items in *l*-nearest neighboring items set, *NeighborI(j),* for each item j which is contained in the favorite-items set of u, *Fi(u)*. For each i of these items, we compute *predict(u, i)* as in Eq. 9. Then, out of the items that user u have never purchased, we recommend the top-t items in the order of prediction values.

$$predict(u,i) = \frac{\sum_{j \in Fi(u)} (ItemSim(i,j) \times scoreN(u,j))}{\sum_{j \in Fi(u)} ItemSim(i,j)} \quad \textbf{(9)}$$

Associate items recommendation. For each item i, we maintain top-l items in the *associate items list* (e.g. l = 10), as explained in section 3.3. For each item in the favorite-items set of u, *Fi(u)*, we examine all of their top-l associate items. In a similar way to item-based collaborative filtering, for each item i of all these, we compute *predict(u, i)* as in Eq. 10, where we use *freqCount(i, j)* value, instead of *ItemSim(i, j)*. Out of the items that user u have never purchased, we recommend the top-t items in the order of prediction values.

$$predict(u,i) = \frac{\sum_{j \in Fi(u)} (freqCount(i,j) \times scoreN(u,j))}{\sum_{j \in Fi(u)} freqCount(i,j)} \quad \textbf{(10)}$$

4 Experimental Evaluation

4.1 Data sets and Evaluation Metrics

As experiment data, actual sales data for 15 months by *Happycoop Co*., one of Korean consumers' cooperative companies (http://shop.happycoop.or.kr) is employed. The number of items is about 4,000 and the number of users is about 5,000, while the number of orders is about 988,000. We use the data after grouping by users and items, with the condition that the number of purchasings is greater than or equal to 5 (C=5).

To evaluate the quality of the top-t recommendation algorithms, we split the dataset into a training set and a test set, by randomly selecting one of the non-zero entries of each row (i.e., of each user) to be part of the test set, and use the other entries for the training set. In order to raise accuracy, for each of the experiments we perform five different runs, each time using a different random partitioning into training and test set. The results reported in the next section are the averages over these five trials. In all of experiments we set l = 10 as the number of neighboring users, neighboring items, or the size of associate items list, and set t = 30 as the number of items be recommended by the top-t recommendation algorithms.

Firstly, we measure the quality of our recommendations by looking at the number of *hits*, i.e., the number of items in the test set that also present in the top-t recommended items returned for each user [17]. If m is the total number of users, we compute the *recall* of the algorithm as:

$$recall = \frac{Number\ of\ hits}{m}$$

A recall value of 1 indicates that the recommendation algorithm is able to always recommend the hidden item, whereas a recall value of 0 indicates that the algorithm is not able to recommend any of the hidden items.

Secondly, we compute the *mean absolute error (MAE)* between the ratings (= scores) and the predictions [24],

$$MAE = \frac{\sum_{i=1}^{m} |r_i - p_i|}{m}$$

where m is the number of predictions made, r_i is the real rating and p_i is the prediction value. The smaller the *MAE* is, the more accurately the recommendation algorithm predicts the users ratings.

Finally, to measure the percentage of users to whom at least one item is recommended, we compute the recommendation *coverage*. Even if the recommendation algorithm has good *MAE* performance, if the coverage is low, the recommendation is meaningless. We compute the coverage as the ratio of the number of items in the test set provided non-zero prediction, over the total number of users.

$$Coverage = \frac{Number\ of\ items\ (p_i > 0)}{m}$$

4.2 Experimental Results

To evaluate our three recommendation algorithms for repetitively purchasing items, firstly we measure the *recall* value, which represents the ratio of recommendation hits. We label the three algorithms *user-CF*, *item-CF* and *assoc-item*. Since existing recommendation methods do not consider periodic and repetitive purchasing environments, it is difficult to directly compare other existing results. Therefore, in order to compare with existing similar results, we use the commonly used cosine similarity measure to implement the user-CF algorithm and label this algorithm as *baseline*.

Figure 4 shows that although the performance recall of all three algorithms are not very high, there are some improvements compared with the baseline algorithm.

Figure 5 and Figure 6 depict prediction performance MAE and recommendation coverage, as we define in the previous section. For item-based CF algorithm, both MAE and coverage have good performance of all, and associate-items recommendation algorithm has the next good performance.

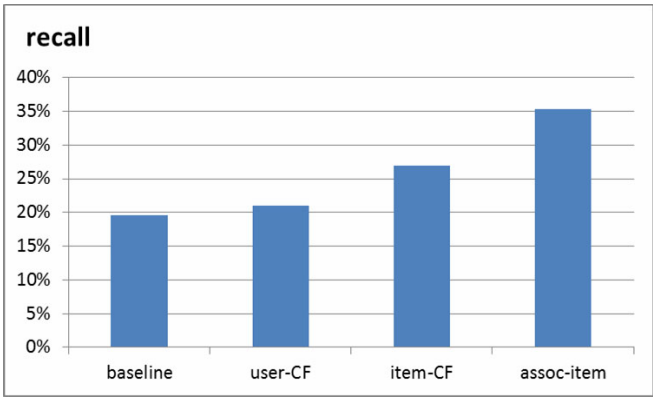

Figure 4. Recommendation hit ratio: recall

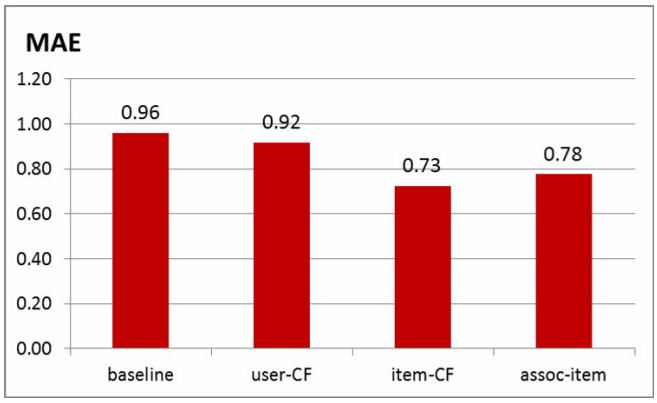

Figure 5. Prediction performance: MAE

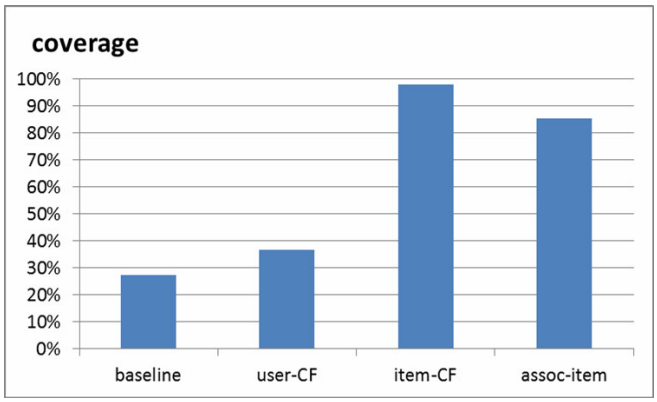

Figure 6. Recommendation coverage

Figure 7 shows the computation time for computing each similarity or finding associate-items. This is the most time-consuming job for each recommendation algorithm.

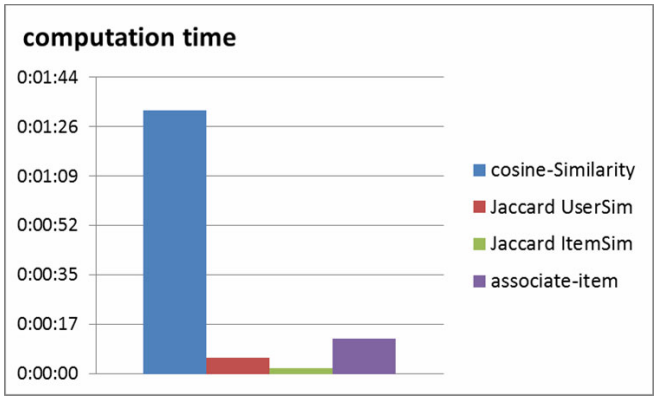

Figure 7. Computation time for computing similarity or finding associate-items

Our experimental results suggest that we will use an *item-CF* algorithm based on item similarity rather than a *user-CF* algorithm based on user similarity when using our system in e-commerce.

Also, when calculating user similarity or item similarity, these results show that it is more efficient to use Jaccard similarity than the commonly used cosine similarity because of the nature of our system of dealing with items that are purchased repetitively.

The *assoc-item* algorithm, unlike the item-CF

algorithm, analyzes all the baskets in the purchase history and calculates the associate items, so it is possible to recommend more diverse items to the user. Therefore, it is also suggested to use the assoc-item algorithm, although the computation time is longer and the performance is slightly lower than the item-CF algorithm.

And it is also suggested to combine the item-CF algorithm and the assoc-item algorithm to recommend the top-*N* items.

5 Conclusion

For regular customers purchasing items repetitively in large shopping malls, our recommendation system can be used to recommend other items showing a periodical and repetitive purchase pattern. By analyzing the items' sales pattern exclusively, we can improve the performance of the recommendation for repetitively purchasing items. We implicitly assign ratings for these items by using the *normalized repetitive purchase counts*, considering overall purchasing intervals, and use these rating values as the scores for the recommendation prediction. We implement user-based and item-based collaborative filtering algorithms for recommending these items. According to the fact that a set of items are purchased together at the same shopping basket for these items, we also implement an algorithm to recommend top-*t* associate items using the association rule mining algorithm. In our experiments, we compare these three algorithms in view of the recommendation hit ratio, prediction performance, and recommendation coverage, along with computation time for these algorithms.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01059937).

References

[1] G. Adomavicius, A. Tuzhilin, Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 6, pp. 734-749, June, 2005.

[2] F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, *Recommendation Systems Handbook*, Springer, 2011.

[3] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of Recommendation Algorithms for E-Commerce, *Proc. of the 2nd ACM Conference on Electronic Commerce (EC '00)*, Minneapolis, MN, 2000, pp. 158-167.

[4] J. Wang, B. Sarwar, N. Sundaresan, Utilizing Related Products for Post-purchase Recommendation in E-commerce, *Proc. of the fifth ACM Conference on Recommender Systems (RecSys '11)*, Chicago, IL, 2011, pp. 329-332.

[5] Y. Yu, C. Wang, Item Attribute-Aware Probabilistic Matrix Factorization for Item Recommendation, *Journal of Internet Technology*, Vol. 15, No. 6 , pp. 975-984, November, 2014.

[6] J. L. Herlocker, J. A. Konstan, A. Borchers, J. Riedl, An Algorithmic Framework for Performing Collaborative Filtering, *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999, pp. 230-237.

[7] P. Resnick, H. R. Varian, Recommender Systems, *Communications of the ACM*, Vol. 40, No. 3, pp. 56-58, March, 1997.

[8] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, *Recommender Systems: An Introduction*, Cambridge University Press, 2010.

[9] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules in Large Databases, *Proc. of the 20th International Conference on Very Large Data Bases*, Santiago de Chile, Chile, 1994, pp. 487-499.

[10] A. Rajaraman, J. Leskovec, J. D. Ullman, *Frequent Itemsets in: Mining of Massive Datasets*, http://infolab.stanford.edu/~ullman/mmds/book.pdf

[11] Y. K. Choi, S. K. Kim, Recommendation Algorithms for Online Shopping Malls with Periodically Purchasing Users, *Journal of KIISE: Software and Applications*, Vol. 40, No. 8, pp. 453-462, August, 2013.

[12] Y. K. Choi, S. K. Kim, An Auxiliary Recommendation System for Repetitively Purchasing Items in E-Commerce, *International Conference On Big Data and Smart Computing*, Bangkok, Thailand, 2014, pp. 96-98.

[13] Happycoop Co., *One of Korean Consumer Cooperative Companies*, http://shop.happycoop.or.kr

[14] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, J. Riedl, GroupLens: Applying Collaborative Filtering to Usenet News, *Communications of the ACM*, Vol. 40, No. 3, pp. 77-87, March, 1997.

[15] H. Marmanis, D. Babenko, *Algorithms of the Intelligent Web*, Manning, 2009.

[16] M. Deshpande, G. Karypis, Item-based Top-N Recommendation Algorithms, *ACM Transactions on Information Systems*, Vol. 22, No. 1, pp. 143-177, January, 2004.

[17] G. Karypis, Evaluation of Item-based Top-N Recommendation Algorithms, *Proc. of the Tenth International Conference on Information and Knowledge Management (CIKM '01)*, Atlanta, GA, 2001, pp. 247-254.

[18] G. Linden, B. Smith, J. York, Amazon.com Recommendations: Item-to-Item Collaborative Filtering, *IEEE Internet Computing*, Vol. 7, No. 1, pp. 76-80, January/February, 2003.

[19] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based Collaborative Filtering Recommendation Algorithms, *Proc. of the 10th International Conference on World Wide Web (WWW '01)*, Hong Kong, China, 2001, pp. 285-295.

[20] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, J. D. Ullman, Computing Iceberg Queries Efficiently, *the 24rd*

International Conference on Very Large Databases, New York, NY, 1998, pp. 299-310.

[21] J. S. Park, M.-S. Chen, P. S. Yu, An Effective Hash-based Algorithm for Mining Association Rules, *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, CA, 1995, pp. 175-186.

[22] A. Savasere, E. Omiecinski, S. B. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, *the 21th International Conference on Very Large Databases*, Zurich, Switzerland, 1995, pp. 432-444.

[23] H. Toivonen, Sampling Large Databases for Association Rules, *the 22th International Conference on Very Large Databases*, Mumbai (Bombay), India, 1996, pp. 134-145.

[24] S.-C. Kim, K.-J. Sung, C.-S. Park, S. K. Kim, Improvement of Collaborative Filtering Using Rating Normalization, *Multimedia Tools and Applications*, Vol. 75, No. 9, pp. 4957-4968, May, 2016.

Biographies

Yoon Kyoung Choi received her bachelor's degree from Seoul National University, Korea, her master's degree from Korea Advanced Institute of Science and Technology (KAIST), and her Ph.D. degree from Chung-Ang University, Seoul, Korea. She is currently a professor at Baewha Women's University, Korea. Her areas of research interest are algorithms and recommendation system.

Sung Kwon Kim received his bachelor's degree from Seoul National University, Seoul, Korea, his master's degree from Korea Advanced Institute of Science and Technology (KAIST), Korea, and his Ph.D. degree from University of Washington, Seattle, U.S.A. He is currently a professor at Division of Computer Science and Engineering, Chung-Ang University, Seoul, Korea.