Determining Optimal Number of Neighbors in Item-based *k*NN Collaborative Filtering Algorithm for Learning Preferences of New Users

A.Mojdeh Bahadorpour[1], B.Behzad Soleimani Neysiani[2], C.Mohammad Nadimi Shahraki[1]
[1] *Department of Software Engineering, Faculty of Computer Engineering, Najafabad branch, Islamic Azad University, Najafabad, Isfahan, Iran.*
[2] *Department of Software Engineering , Faculty of Computer & Electrical Engineering, University of Kashan, Kashan, Isfahan, Iran*
bahadorpour@sco.iaun.ac.ir

***Abstract*— Although the collaborative filtering (CF) is one of the efficient techniques to develop recommender systems, it suffers from a well-known problem called cold start which is a challenge to know the new user preferences. Ask To Rate technique is a simple way to solve this problem. In this technique, some items are shown to the new user, and ask her/him to rate them. Usually, Ask To Rate technique selects the items using kNN algorithm. However, determining k or number of the new user's neighbors in this algorithm is critical, because it affects the accuracy of recommender system. In this paper, a CF based recommender system is improved by Ask To Rate technique to solve cold start problem. Consequently, k or number of the new user's neighbors is determined by an experimental evaluation. The experimental results on MovieLens dataset show that the highest accuracy of recommendations can be seen when the number of neighbors is set by a low value e.g. 10-15 neighbors.**

***Index Terms*— Cold Start Problem; Collaborative Filtering; kNN Algorithm; Recommender System.**

I. INTRODUCTION

Recommender systems [1, 2] are one of the oldest and most successful applications in the Web usage mining which help people make decisions in a compact information space. Depending on how recommendations are generated, there are different recommender system techniques [3]. The CF recommender systems use opinions (ratings) of other users to suggest items to the target user [4].

A common problem in CF recommender systems is the cold start problem [5-7]. It occurs when the new user is logged into the system. Due to lack of ratings of the new user in the CF, it is impossible to calculate the similarity between her/him and other users and thus the system cannot make accurate recommendations.

To mitigate the user's cold start problem, collecting data and learning his/her preferences should be done when he logs into the system. One of the most outspoken techniques for overcoming this problem is to ask the new user for explicit ratings on several specified items [5, 6, 8-11]. Then, the system begins to suggest items to the user with initial information about the new user preferences using the CF recommender algorithm.

In the neighbor-based CF algorithm (user-based/item-based algorithm), It is critical to determine the optimal number of neighbors for the new user because it has a direct impact on the accuracy of the recommendations made by the algorithm.

The rest of the paper is structured as follows. Section 2 introduces CF-based recommender systems and one of the algorithms of this technique. Section 3 introduces some approaches for mitigating the cold start problem. Section 4 presents the studied problem. Section 5 implements and evaluates the results of implementations. Finally, Section 6 is the conclusion.

II. COLLABORATIVE FILTERING SYSTEM

The CF recommender system is originated from a behavior used by human for centuries, i.e. sharing ideas and opinions with others. The main focus of these systems is based on the similarity between users instead of similarity based on item contents, and they try to consider the utility of items according to opinions of similar users (neighbors) for the target user who previously rated items [4, 12]. These systems compare the record of the target user's preferences with those of all other users in order to find users with similar interests. This set of users with similar interests is known as neighborhood of the target user. Mapping the records of a user to his/her neighbors can be done based on the similarity of item ratings, access to pages with similar content or buying the same items. Then the obtained neighbors are used for recommending items that are not still observed by the target user.

Breese et al. [13] found that two types of CF algorithms have been proposed. They presented two categories based on the usage of CF algorithms of the user-item matrix in arguments:

- Model-based approaches: they are made of a two-stage process for making recommendations. In the first stage, a model (such as data mining or machine learning algorithms) is calculated offline for rating users. In the second stage, a recommendation is made for the target user based on the learned model [13].
- Memory-based approaches: By these algorithms, all rates, items and users, i.e. memory-based data, are stored in memory in a structure called the user-item matrix, and all or part of this database is used to make a prediction or recommendation. The neighbor-based CF (user-based/item-based algorithms) and Item/user-based top-N recommendations [14] are the most common memory-based techniques.

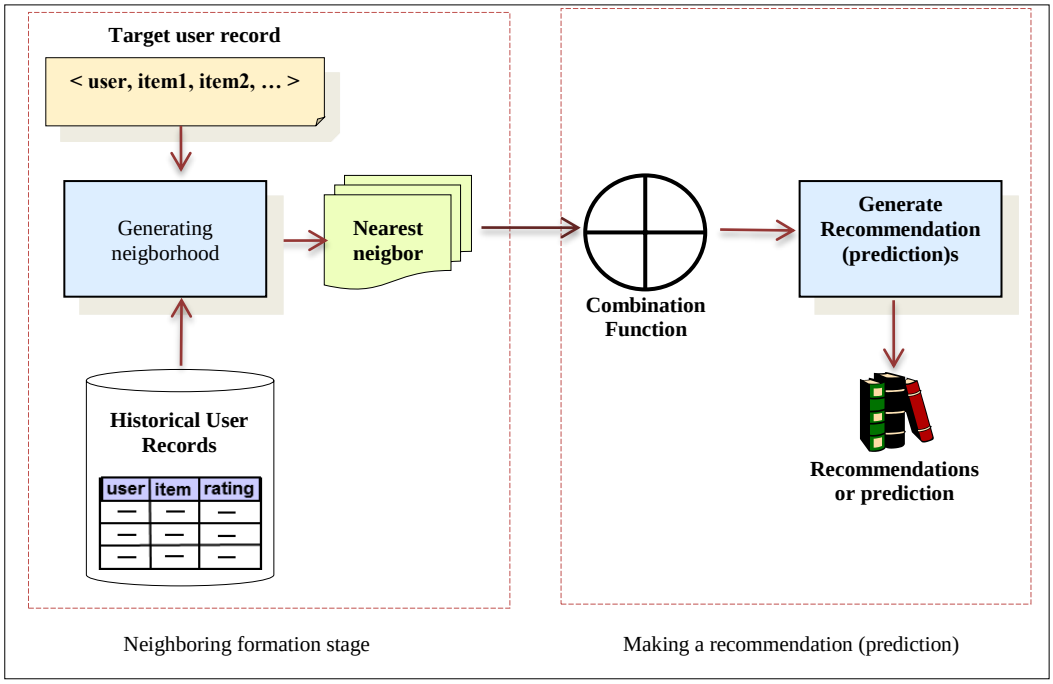

Figure 1 Neighbor-based CF algorithm

A. *Item-based k Nearst Neighbor CF algorithm*

The item-based *k*NN CF algorithm [15] is a neighbor-based approach which is of memory-based type. Instead of calculating the similarity between users, the algorithm calculates the similarity between items that the target user has rated (purchased or observed) with the target item. In this algorithm, the predicted value of items is produced based on two stages, as shown in Figure 1 [16].

In the first stage of the algorithm, i.e. the neighboring formation stage, the neighbors of the target user are identified according to the target user profile u_t and other users with similar interests; i.e. all rated items of the target user are sorted by their similarity with the target item a_t.

There are several ways to calculate the similarity between items such as Pearson correlation coefficient, cosine distance, and adjusted cosine similarity [15]. This study uses the adjusted cosine similarity to calculate the similarity between items, as shown in Equation 1.

$$sim(a_t, a_i) = \frac{\sum_{m=1}^{h} (r_{u_{m,},a_t} - \overline{r_{u_m}}).(r_{u_m,a_i} - \overline{r_{u_m}})}{\sqrt{\sum_{m=1}^{h} (r_{u_m,a_t} - \overline{r_{u_m}})^2} \cdot \sqrt{\sum_{m=1}^{h} (r_{u_m,a_i} - \overline{r_{u_m}})^2}} \quad (1)$$

Where $\overline{r_{u_m}}$ is the average ratings for the *m*th user for all items that have been rated by her/him and *h* is the number of the set of all users who have rated both items at and a_i ($h \in u(a_t) \cap u(a_i)$).

In the second stage, i.e. the process of making a recommendation (prediction), the prediction is made according to Equation 2 by examining the target user's ratings and finding the k most similar items to the item a_t.

$$prediction\ (u_t, a_t) = \frac{\sum_{k\_similar\_items,d} (sim(a_t, d) * r_{u_t,d})}{\sum_{k\_similar\_items,d} (|sim(a_t, d)|)} \quad (2)$$

One of the common problems in CF recommender systems is the cold start problem. It occurs when the new user is logged into the system. Due to lack of new user ratings in CF, it is impossible to calculate the similarity between her/him and other users and thus the system cannot make detailed recommendations. So the system is faced with the challenge of how to generate predictions or recommendations for the new user who has logged in recently.

Also the item cold start problem occurs when some items are recently inserted in system and are not yet rated by users, therefore they are not recommended by system. To cope with the cold start problem, different methods have been proposed, solving the problem in recent years N.Houlsby et al. used a new matrix factorization model to rate data and a dynamic learning strategy. In suggested method a framework based on a Bayesian active learning strategy is applied [17].

With the arrival of the new user, recommender systems should try to gather information and learn about his interests before s/he can fully use the system, because it is critical to consider the cold start problem. The users criticize the utility of recommender systems based on their first experience. There are several approaches to reduce these limitations [6, 9, 10, 18, 19].

III. Overcoming The Cold Start Problem

One of the newest methods to cope with the user cold start problem is proposed by B.Lika et al [20]. Initially, the method has presented a model which has been design based on Demographic data and Similarity techniques. User's neighbors found, suppose that the new user and the neighbors with the same features and foreground will have similar preferences. Then each new user is classified into a group and according a rating prediction mechanism, some items are selected for new user to rate them.

One of the techniques to overcome the cold start problem is to ask the new user to explicitly rate several specified items and quickly build up a profile for her/him [5, 6, 9, 10] or by using questionnaire trees (tree structures) to build

adaptive questionnaires [10, 21]

In ask to rate approaches, the selected items are not recommendations. In these approaches, some items - which are focused on learning user preferences - will be displayed to the new user. As shown in Figure 2, after displaying some items to the new user and receiving a certain number of rates, the ask to rate process is completed. In the user-item matrix, while the row for the new user is non-empty, the new user enters the normal phase of the recommender system. The CF system will use these rates to calculate the similarity between the new user with other users (neighbors), and recommended items or predicted rates are created for the user, and the system starts to check user activities for forming a feedback loop to update the profile.

Little research has been done to guide the cold start problem based on ask to rate technique. The introduced approaches can be divided into two categories: non-adaptive and adaptive.

Figure 2 New user signup process and initial interview

A. *Non-adaptive Approaches*

In these approaches such as entropy and variance [6]; random, popularity, pure entropy and balanced strategy [9]; entropy0 and Harmonic mean of Entropy and Logarithm of Frequency(HELF) [10]; Greedy method, Other People's Greedy and Variations [5], regardless of knowledge changes to any user who is being interviewed, the same items will be shown to all users. Next, we will introduce three introduced approaches.

1) *Popularity*

In statistical terms, popularity of an item can be considered as items with higher frequency. In this approach, after considering the user-item matrix, items that have received more ratings are sorted in descending order. Then a few items with the highest level of popularity are selected as the most popular items and displayed to the new user.

2) *Pure Entropy*

One of the shortcomings of the popularity method is that it ignores the potential information that may be included in the item rates; i.e. one can imagine that specific items lead to obtaining further information on user interests than others. In general, an item that some people like and others dislike contains more information than an item that all people like. Entropy was proposed by [6] as a method with low complexity for calculating the amount of information embedded in the ratings of an item and was presented again in [9]. Entropy of items represents the distribution of user opinions about items. For each item, at is calculated using the pseudo-code shown in Figure 3.

```
function Entropy (at)
entropy (at) = 0
foreach item at in dataset
   for i as each of the possible rating values
                if at's rating = i
                   valu[i] += 1
   end for
   proportioni  = valu[i] / total number of users who rate at
   entropy (at) += proportioni * Math.log (proportioni , 2)
end foreach
entropy (at) = – entropy (at)
End
```

Figure 3 Pseudo code of entropy approach

3) *Balanced Strategies*

To exploit the advantages of both of entropy and popularity approaches, their combination is proposed. This is why items with a higher entropy rate contain more information, although users may find relatively few items for ratings; and inversely, items with a higher popularity contain higher ratings from users, although each rating may contain little information for the recommender system. In this approach, using Bayes theory, it is assumed that the popularity and entropy approaches are independent of each other. The first balanced approach is *poularity*entropy*. But in the studies and experiments of [9], it was found that the popularity value is often dominant in this product. Thus, the (log *poularity)*entropy* balanced approach was proposed. By applying log, the popularity criterion becomes linear and will have a better consistency with entropy. Then, based on the value obtained by a balanced strategy, items are sorted in descending order, and a certain number of items with highest criterion value are shown to the new user.

B. *Adaptive Approaches*

In this type of approaches, the new user's past ratings and her/his profile changes during the initial interview are considered. New users will rate items that are sorted in a personalized way and will observe a more effective interview process than non-adaptive approaches others, such as item-item personalized approach [9]; IGCN (Information Gain through Clustered Neighbors) approach [10]; Naïve Bayes and Perturbed Other People's Greedy approaches [5]; Clustering method [18]; and questionnaire trees [21]. Next, a case of adaptive approaches is discussed.

1) *Item-item personalized approach*

In this approach, first by a non-adaptive approach (in the study by [9], was chosen by the research group), 200 films (items) are selected. Among them, a film is shown randomly to the new user until he rates at least one film. Then, using a recommender system engine such SUGGEST [22], the similarity between the films is calculated and similar films (which might have been seen by the user) are selected based on the values already given by the user as rating. When the user rates more films, the similar film list is updated. However, the films already seen by the user will not be redisplayed. This approach does not consider the user's favorite films, and it only focuses on the films seen by the user.

IV. Determining The Number Of Neighbors For The New User

After user login to the recommender system, the number of user's neighbors in the user- or item-based *K*NN algorithm (*k* value) should be carefully selected because the number of nearest neighbors to the new user directly affects the accuracy of recommendations.

In this paper, to determine the number of neighbors of the new user, the new user signup process offline simulation framework proposed by [10] is implemented. Under this framework, in the initial interview stage and making a fast profile for the new user, a certain number of items are selected and displayed by one of the approaches to solve the cold start problem based on the ask for rating. Then the new user will log into the recommender system, and the recommendation operation will begin. Considering different values for *k* and examining the mean absolute error (MAE) for predictions, one can investigate the optimum value of *k* and detect its determinants.

V. Experiments And Results

To determine and extract the number of neighbors of the new user in the item-based KNN algorithm, the dataset extracted from the MovieLens movie recommender service is used which was collected by the GroupLens research group in the University of Minnesota [23]. The characteristics of this dataset with 95.73 percent of sparsity are shown in Table 1.

Table 1
Characteristics of MovieLens 1M dataset

Dataset Characteristics	Users	Films	Ratings
Total	6040	3883	1000209
Minimum rating number	20	1	1
Maximum rating number	2314	3428	5
Average ratings	165.6	269.9	3.58

For performing accurate experiments and evaluations, this article uses the 20:80 split for the original dataset into two training and test sets so that if the dataset is considered as a user-item matrix, this segmentation is done vertically and for each user, 80% of ratings will randomly fall into the training set and the remaining 20% will fall into the test set. The training dataset is used for calculation of the used approach (item-item personalized) and running offline simulation. The evaluations are reviewed using the test dataset.

To assess the accuracy of recommendations, the mean absolute error (MAE) criterion is used [10, 13]. The accuracy of recommendation is measured by Equation 3 based on the error level in the prediction of item's ratings.

$$MAE = \frac{\sum_{i=1}^{n} \left| prediction_{u,a_i} - real_{u,a_i} \right|}{|N|} \qquad (3)$$

To calculate the MAE for user *u*, *N* is the number of items that user *u* has rated $predication_{u,a}$, is the predicted rate that user *u* will give to item *a*, and $real_{u,a}$ is the real rate of user *u* for item *a*. The *MAE* is negatively oriented. It means that a lower value represents a better prediction.

In the experiments, the item-item personalized approach is used to select and display initial items to the new user; 15, 30, 45, 60 items are selected and displayed. Then, when the new user logs into the recommender system and the recommendation operations of the item-based *k*NN CF algorithm begins, the number of neighbors of the target user is selected between 5 to 50 with a distance of 5 and the number of 500 (for more exact examination). Then, given the average of accuracy values for predictions (the MAE criterion), the optimal value for *k* variables is chosen. According to the standard dataset used in the new user signup process (Figure 4), optimal values for the number of neighbors are 5, 10 or 15 .

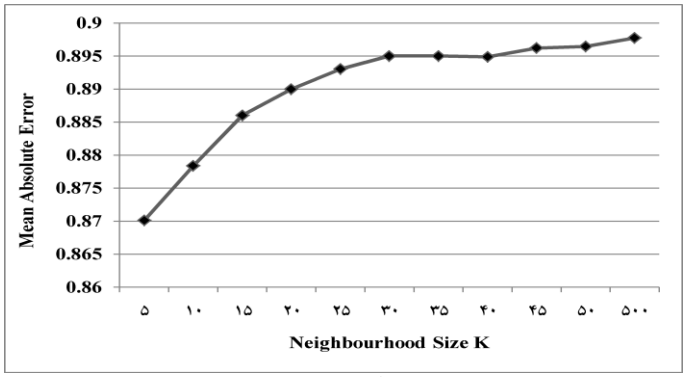

Figure 4 Determining Optimal Number of Neighbors (K) in Item-based kNN Collaborative Filtering algorithm

As shown in Table 2, considering the distribution of positive and negative values of similarity between a target item and *k* neighbors, one can justify the increased error level of resulting predictions due to the increased number of neighbors. In this table, the average number of neighbors with positive and negative similarities was determined using the adjusted cosine similarity relationship used in the used item-based *k*NN algorithm.

For example, if 5 neighbors are considered for calculating the predicted rate of an item, about 44% of similarities become positive and 56% negative; i.e. if the item has exactly 5 neighboring items, about 2 items has positive similarities with that item and about 3 items has negative similarities. As the number of neighbors which have positive similarities with the item decreases and the number of neighbors with negative similarities increases, the accuracy of recommendations will decrease because as the number of neighbors with negative similarities increases, the item-based *k*NN CF algorithm will not be able to make accurate predictions.

Table 2
Distribution of neighbors with positive or negative similarity

number of neighbors (*k* value)	Positive similarity (%)	Negative similarity (%)
5	43.90	56.10
10	36.03	63.98
15	32.96	67.05
20	31.48	68.53
25	30.58	69.42
30	30.01	69.99
35	29.59	70.42
40	29.30	70.70
45	29.00	71.00
50	28.78	71.22
500	28.62	71.38

VI. CONCLUSION

The recommender systems are one of the best tools to deal with the problem of overload information which will help users to find optimal interested items. The CF algorithm is one of the most common recommender system algorithms. The designers of these systems are trying to affect new users, those who have the highest judgment about the recommender system, and make recommendations with high accuracy for them. The most straightforward solution is a short interview with the new user for evaluating several specific products or items. After making an initial profile for the new user to log into the recommender system and estimating the rate value for items not seen by the user, the number of neighbors of the new user which is used in the neighbor-based CF algorithm is an influencing variable in the accuracy of recommendations.

This paper determines the optimal number of neighbors in the item-based CF *k*NN algorithm after login of the new user to the recommender system. After implementing the new user signup process framework, the results indicate that optimal number of neighbors for the new user is 5 to 15 in accordance with standard dataset used. If the number of neighbors is considered greater than 15, more neighbors with negative similarity will be involved in calculating the item rate prediction for the new user, reducing the accuracy of recommendations. If the number of neighbors is considered less than 5, no neighbors may be found for the user. In these conditions, it is proposed that the k value is not constant for everyone, and only positive neighbors for each user are considered. It can be a subject for future studies.

REFERENCES

[1] P. Resnick, and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56-58, 1997.

[2] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109-132, 2013.

[3] R. Burke, "Hybrid recommender systems: survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331-370, 2002.

[4] X. Su, and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1-19, 2009.

[5] M. Crane, "The new user problem in collaborative filtering," Department of Computer Science, University of Otago, Dunedin, New Zealand., 2011.

[6] A. Kohrs, and B. Merialdo, "Improving collaborative filtering for new users by smart object selection," 2001.

[7] X. Zhao, "Cold-Start Collaborative Filtering," UCL (University College London), 2016.

[8] N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems." pp. 1805-1808, 2010.

[9] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: learning new user preferences in recommender systems." pp. 127-134, 2002.

[10] A. M. Rashid, G. Karypis, and J. Riedl, "Learning preferences of new users in recommender systems: an information theoretic approach," *ACM SIGKDD Explorations Newsletter*, vol. 10, no. 2, pp. 90-100, 2008.

[11] T. N. Lillegraven, and A. C. Wolden, "Design of a Bayesian recommender system for tourists presenting a solution to the cold-start user problem," Department of Computer and Information Science, Norwegian University of Science and Technology, 2010.

[12] G. Adomavicius, and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, 2005.

[13] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering." pp. 43-52, 1998.

[14] G. Karypis, "Evaluation of item-based top-n recommendation algorithms," in Proceedings of the tenth international conference on Information and knowledge management, Atlanta, Georgia, USA, 2001, pp. 247-254.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms." pp. 285-295, 2001.

[16] B. Mobasher, "Personalizing the Web: Building effective recommender systems," 2005.

[17] N. Houlsby, J. M. Hernández-Lobato, and Z. Ghahramani, "Cold-start active learning with robust ordinal matrix factorization." pp. 766-774, 2014.

[18] C. Bouras, and V. Tsogkas, "Clustering to deal with the new user problem." pp. 58-65, 2012.

[19] M.-H. Nadimi-Shahraki, and M. Bahadorpour, "Cold-start Problem in Collaborative Recommender Systems: Efficient Methods Based on Ask-to-rate Technique," *CIT. Journal of Computing and Information Technology*, vol. 22, no. 2, pp. 105-113, 2014.

[20] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 4, pp. 2065-2073, 2014.

[21] R. Karimi, A. Nanopoulos, and L. Schmidt-Thieme, "A supervised active learning framework for recommender systems based on decision trees," *User Modeling and User-Adapted Interaction*, vol. 25, no. 1, pp. 39-64, 2015.

[22] G. Karypis, "SUGGEST* Top-N Recommendation Engine.," Karypis Lab, 2000.

[23] GroupLensResearch, "MovieLens Data Sets," *The available rating data sets from the MovieLens web site*, M. w. site, ed., 2013.