International Journal of Database Theory and Application
Vol.7, No.2 (2014), pp.131-140
http://dx.doi.org/10.14257/ijdta.2014.7.2.13

An Expected Item Bias based Hybrid Approach for Recommendation System[1]

Kaikuo Xu[1], Changan Yuan[2*], Fan Li[3] and Xianbin Liu[4]
[1]*College of Computer Science, Chengdu University of Information Technology, ChengDu, 610225, China*
[2]*Guangxi Teachers Education University, Nanning 530001, Chinas*
[3]*Cloud Computing Open Laboratory, Chengdu University of Information Technology, ChengDu, 610225, China*
[4]*School of Computer Science, Sichuan University*
[1]*kaikuoxu@gmail.com*, [2]*yca@gxtc.edu.cn*

Abstract

To improve the accuracy of memory based recommendation while keeping the low time cost, an expected item bias (EIA) based similarity computation is proposed. And a hybrid approach (HA) integrating the global rating information and local rating information is also proposed. The features of two classical datasets MovieLens and Netflix for recommendation system benchmarking are anglicized. The experiments on MovieLens and Netflix show that both EIA and HA could improve the performance alone. A combinational use of them will lead even better results on the two benchmark datasets.

Keywords: *Recommendation System, Hybrid Approach, Personized Service, Collaborative Filtering*

1. Introduction

Recommendation system is widely used in various e-business systems such as Netflix [1], CDNow [2] and Amazon [3]. The goals of application of recommendation system are: (i) increase the chance of being visited for a specified commodity; (ii) increase the time the users surf the sites; (iii) help users to discover interesting commodities. Different recommendation techniques are used to lead users to their preferable products. They are basically classified into three categories:

(i) Recommendation according to the properties of commodities such as brand, price and available season. For example, recommend shampoo of another brand to users buy shampoo frequently; recommend commodities of a brand to users buy commodities of the same brand frequently. This is the basic approach: although it is commonly used, the accuracy still needs improvement.

(ii) Recommendation according to users' ratings and their shopping list. For example, commodities with good ratings for most users are really good and is worthy of being recommended; recommend commodities bought by a user to another user with same ratings

[1] This work was supported the Natural Science Key Foundation of Guangxi under Grant No.2011GXNSFD018025, the Development Foundation of CUIT under grant No. KYTZ201108, the Natural Science Foundation under Grant No. 61363037 and Haikou city key science and technology program (2012-027).
[*] Corresponding Author

on the same commodities. This requires users' ratings and cannot work everywhere. Douban [4] and Digg [5] are of this type.

(iii) Recommendation according to users' surfing history. For example, if many users visit commodities A, C and E, A, C and E should be a recommendation commodity for each other respectively; recommendation according to a user's current visiting action and its surfing history. This method could work anywhere and could improve users' experience.

MovieLens[6] and Netflix[1] are two classical datasets for the research of recommendation system. MovieLens is from project named GroupLens while Netflix is from the world's biggest DVD renter. GroupLens is one of the most famous groups in the field of recommendation system. It roughly adopts an ensemble approach combining both collaborative filtering and association rules.

In this paper, an expected items bias based approach is proposed to compute the similarity between user/item pair, no matter the pair has common items/users. Also a hybrid approach integrating the global and local rating information is proposed to improve the accuracy while keeping the time speed. The experiments show the availability of the proposed approaches.

The rest of the paper is organized as follows. Section 2 summarizes on some related work. Section 3 analyzes the sparsity of data and the strategies to conquer it. Section 4 proposes an expected item bias based similarity computation. Section 5 proposed hybrid approach for recommendation. Section 6 evaluates the proposed methods on MovieLens and Netflix. Section 7 concludes the paper.

2. Related Work

The research on recommendation system could be traced back to cognitive science [7] and approximation theory [8]. Until 1990s, recommendation system began to be an independent branch as the focus of research. In general, recommendation system picks one or several potential user preferable items from a set of candidate items and recommends them to users to make decisions. The detail definition of recommendation system is given in [9]. The description below is simplified.

Let *u* be an active user, *i* be an item (commodity), in order to recommend one item to *u*, a measure function *utility*(*u, i*) is needed to score the utility of *i* to *u*; Let *ISubSet* be a set of items (commodities), in order to recommend several items to *u*, a measure function *utility*(*u*, *ISubSet*) is needed to score the utility of *ISubSet* to *u*. To be formally, Adomavicius etc describes recommendation system recommending one item as a maximization problem: [10]

$$\forall u \in U, i' = \arg\max_{i \in I} (utility(u, i)) \qquad (1)$$

And the description on recommendation system recommending several items is quite the same:

$$\forall u \in U, ISubSet = \arg\max_{i \in I} (utility(u, ISubSet)) \qquad (2)$$

When the size of *ISubSet* is 1, formula 1 is equal to formula 2.

Recommendation system with measure function is classified into three categories according to how the measure function computes scores:

(i) Content based recommendation [11]. If user likes item *i*, recommend an item with same taste with *i*. For example, in movie recommendation, same taste refers to the theme, actor and director of the movies. Since there is no unified model to describe items of different types, no generalized content based methods exist right now.

(ii) Collaborative recommendation [12]. According to [13], collaborative recommendation is classified into two categories: memory based method and model based method. The focus of this paper is memory based method since model based method doesn't fit for online processing. Memory based method judges the degree a user likes an item based on the ratings the user gives to other items. Let $R_{u,i}$ be an unknown rating for *u* and *i*, it is computed according to the aggregation of the ratings of top *n* similar users with *u:*

$$R_{u,i} = \underset{u' \in Uset}{aggr} (r_{u',i}) \qquad (3)$$

Here *Uset* is the set of top *n* similar users with *u* having rated *i*. The simplest way of aggregation is computing the average rating. The most significant flaw for collaborative filtering is that the accuracy is affected by the sparsity of the data seriously, which will be discussed in section 3.

(iii) Hybrid approach. It combines both content based recommendation and collaborative filtering. It is classified into four categories according to the way of the combination. (a) Combine the results of the two methods; (b) Integrate the former to the latter; (c) Integrate the latter into the former; (d) merge the two into a unified one. The study in [14-16] shows that hybrid approach could improve the accuracy dramatically.

3. Analysis on the Sparsity of Data

Definition 1 (**Sparsity**). Let *n* be the cardinality of users, *m* be the cardinality of items, *p* be the cardinality of ratings, the sparsity of the rating matrix is 1-(*p*/(*n***m*)).

Example 1. Let us take MovieLens as an example. The values for *n, m* and *p* are 3900, 6040 and 1000209 respectively. Thus the sparsity of MovieLens is 1- 1000209/(3900*6040)=1-0.0042 =0.9958. It indicates that the ratings for MovieLens is really sparse.

As a matter of fact, the ratings for most real datasets are sparse. To produce accurate results from highly sparse data is difficult. Two strategies are used to conquer this problem. The first strategy is filling the missing values. The work in [17] filled each missing value with permanent score and the result shows that the accuracy of recommendation is improved. The work in [18, 19] filled each missing value with computed score and it leads better results than those in [17]. The problem is that even the filling result is affected by the sparsity of the data. And the similarity computation between item pair is time cost. The work in [19] further adopted the iteration scheme and clustering algorithm and gets even better improvement. The problem is that iteration scheme and clustering are both time cost. Also clustering algorithm is of accuracy problem. The second strategy is recursive prediction [12]. It improves the accuracy more or less. However, the time cost increases sharply with the recursive depth.

To improve the accuracy of recommendation and tackle the problem in [18, 19], the expected item bias is used to filling the missing values, which will be described in the next section in detail. Before further description, the symbols are described in Table 1.

Table 1. List of Symbols

Symbol	Definition
u, R_u	A single user in U, the item subset rated by u
U	The user set
i, j	Two single items in I
I	The item set
$r_{u,i}$	The rating score of u on i
$\bar{r}_i, \bar{r}_j$	The average rating score for all users on i, j respectively.
$PR_{u,i}$	The predicted rating score of u on i
μ	The global average rating score
sim(u_1, u_2)	The similarity between u_1 and u_2
sim(i_1, i_2)	The similarity between i_1 and i_2
$S_{i,j}$	The user subset rating both i and j, i.e. $S_{i,j} = \{u \in U \mid r_{u,i} \neq \Phi \; \& \; r_{u,j} \neq \Phi\}$

4. Similarity Computation

Similarity computation is an essential step for memory based method. The most commonly used similarity computation formulas are Pearson coefficient, cosine similarity and adjust cosine similarity. Here Pearson coefficient is adopted for improvement. Our method also works for other similarly computation formulas.

$$sim(i, j) = \frac{\sum_{u \in S_{ij}} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in S_{i,j}} (r_{u,i} - \bar{r}_i)^2 \sum_{i \in S_{i,j}} (r_{u,j} - \bar{r}_j)^2}} \quad (4)$$

The Pearson coefficient for computing the similarity between items is shown in formula 4. Our improved computation process is shown in algorithm 1. Algorithm 1 calls algorithm 2 and algorithm 2 calls algorithm 3. They will be explained one by one in reverse order.

In algorithm 3, *dev* is used to record the expected item bias between items and the number of users having rated both items. The rows of trainRatingMatrix represent items while the columns represent users. trainRatingMatrix[*j*.itemID][k] != 0 indicates that user *k* has rated item *j*.

Proposition 1. Let *n* be the number of users, the time complexity of algorithm 3 is O(*n*).

In algorithm 2, MaxRatingNumberOfItem indicates the number of items. urm records the ratings for each user.

Proposition 2. Let *m* be the number of items *u* has rated, the time complexity of algorithm 2 is O(*m*).

Observation 1. In most datasets, $|R_u \cap R_v| / |R_u U R_v| < 1/10$.

Proposition 3. Let *n* be $|R_u \cap R_v|$, *m* be $\max(|R_u|, |R_v|)$, the time complexity of algorithm 1 is O(*nm*).

Proof: According to observation 1, algorithm 1 needs to predicts the rating score for 1/10 of $|R_uUR_v|$. Since the time complexity of algorithm 2 is O(m), therefore the time complexity of

Algorithm 1 ∶ UserCorrelation

```
Input：  u, v
Output： correlation
1.  float sum = 0;
2.  float sumSquareX = 0;
3.  float sumSquareY = 0;
4.  int count = 0;
5.  for(item i in ($R_u$ U $R_v$))
6.  {
7.      float meanU1 = getMeanOfUser(u);
8.      float meanU2 = getMeanOfUser(v);
9.      if(user u has rated item i but user v has not )
10.     {
11.         ratingMatrix[v][i]  = predict(v, item);
12.     }
13.     else if(u has not rated i but v  has rated i)
14.     {
15.         ratingMatrix[u][i] = predict(u, item);
16.     }
17.    float x = (ratingMatrix[u.userID][i] - meanU1);
18.     float y = (ratingMatrix[v.userID][i] - meanU2);
19.     sum += x*y;
20.     sumSquareX += x*x;
21.     sumSquareY += y*y;
22. }
23. correlation =  sum/ (sumSquareX*sumSquareY);
24. return correlation;
```

Algorithm 2 ∶ Predict

```
Input：  u, i
Output： ratingscore
1.  float rating = 0;
2.  float sumDev = 0;
3.  for(int k = 0; k <MaxRatingNumberOfItem;k++)
4.  {
5.      if(urm[u.userID][9] != null)
6.      {
7.          int id= urm[u.userID][9].item.itemID;
8.          if(i.itemID !=  id)
9.          {
10.             rating += (dev[i.itemID][id].dev +
          urm[u.userID][9].rating*dev[i.itemID][id].count);
11.             sumDev += dev[i.itemID][id].count;
12.           }
13.      }
14.      else
15.      {
```

```
16.             break;
17.         }
18.     }
19.     ratingscore = rating/sumDev;
20.     return ratingscore;
```

algorithm1 is $n/10+9n/10*O(m)$. For simplicity, it is denoted as $O(nm)$.

Algorithm 3 : calculateDev

```
Input:  i, j
Output: dev
1.  float devTemp = 0;
2.  for(int k = 1; k < NumberOfUser + 1;k++)
3.  {
4.     if(trainRatingMatrix[i.itemID][k] != 0 &&
           trainRatingMatrix[j.itemID][k] != 0)
5.     {
6.        devTemp = (trainRatingMatrix[i.itemID][k]-
7.                trainRatingMatrix[j.itemID][k]);
8.        dev[i.itemID][j.itemID].addDev(devTemp);
9.        dev[j.itemID][i.itemID].addDev(-devTemp);
10.    }
11. }
12. return dev;
```

Since for some user pair, there are no common rated items, the similarity between them cannot be computed with Pearson coefficient or cosine similarity. In most cases, this is not true. In reality, even people don't know each other many have some relationships. The theory of Six Degrees of Separation explains this phenomenon [20]. And algorithm 1 could compute the similarity between this type of user pair through similarity propagation. Through similarity computation, the neighbors of all users could be generated.

5. The Hybrid Approach for Recommendation

The naïve approach is to recommend according to the neighbor's preference. Formula 5 predicts the degree u likes i based on the neighbors of u. The similarity computation could use any formula in Section 4.

$$PR_{ui} = \frac{\sum_{u' \in U} sim(u, u') \times r_{u'i}}{\sum_{u' \in U} |sim(u', u)|} \quad (4)$$

In general, the average rating of a user reflects whether he/she is picky on the items. When the average rating of a user is higher than the global average rating, he/she is optimistic; while the average rating of a user is lower than the global average rating, he/she is pessimistic. Therefore, we integrate average rating score of the users in our method. And this also holds for items. When the average rating score of an item is higher than the global average rating score, it is enjoyed by most users; while the average rating score of an item is lower than the global average rating score, it is not liked by most users. Also we integrate average rating score of the items in our method. Our method is a linear combination approach, which is described in formula 5.

$$PR_{ui} = \alpha(\mu + \overline{r_u} - \mu + \overline{r_i} - \mu) + (1-\alpha)\frac{\sum_{u' \in U} sim(u,u') \times r_{u',i}}{\sum_{u' \in U} |sim(u',u)|} \tag{5}$$

After simplification, formula 6 is generated.

$$PR_{ui} = \alpha(\overline{r_u} + \overline{r_i} - \mu) + (1-\alpha)\frac{\sum_{u' \in U} sim(u,u') \times r_{u',i}}{\sum_{u' \in U} |sim(u',u)|} \tag{6}$$

Formula 6 will not increase the time cost comparing with formula 4.

There are also some other hybrid approaches like the method in [21]. However, they are more time cost. The method in [21] uses least square method to learn the parameters, which is very slow when the dataset is large.

6. Experiments

The experiments are run on an INTEL core 2DuoProcessorE2160 with 2G memory with Windows XP. The algorithms are implemented with Eclipse Europa and the program is run on J2SE 5.0.

6.1. Datasets

Both MovieLens and Netflix are used in the experiment. 80% of the data are used as the training dataset while the left 20% are used as the testing dataset.

Figure 1 and Figure 2 show some characteristics of Netflix [22]. According to Figure 1, most values fall in [0, 5000]. According to Figure 2, most average rating scores fall around 3.8.

Figure 3 and Figure 4 show some characteristics of MovieLens [6]. In Figure 3, most values fall in [3.8, 4.1]. And according to Figure 4, most user pair share zero common item and most common item number is smaller than 100.

6.2. Evaluation

Both MAE (Mean Absoulte Error) and RMSE (Root Mean Squared Error) are chosen as the evaluation criteria. The methods in [6], [18] and [19] are denoted as T_CF, Fill, Iterator respectively. The method use the similarity computation in Section 4 is denoted as EIB. The

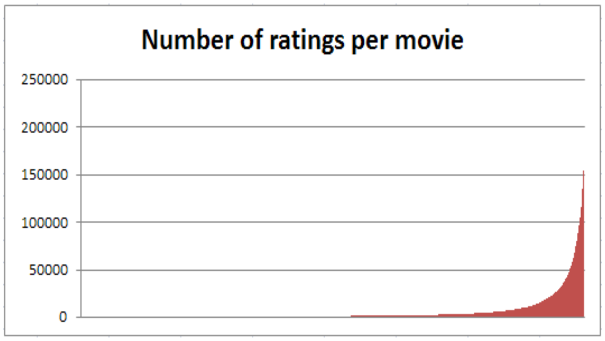


Figure 1. Number of Ratings per Movie in Netflix

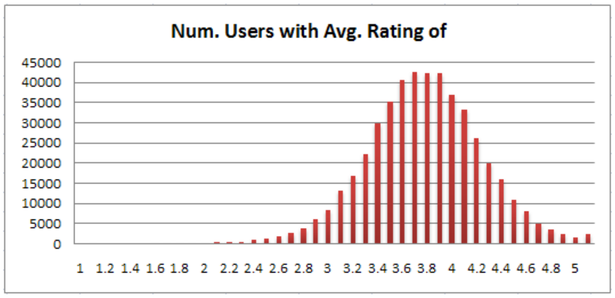
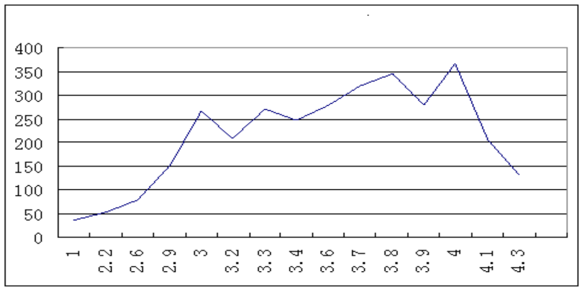


Figure 2. Num. Users with Avg. Rating of in Netflix



Figuew 3. Num. Users with Avg. Rating of in MovieLens

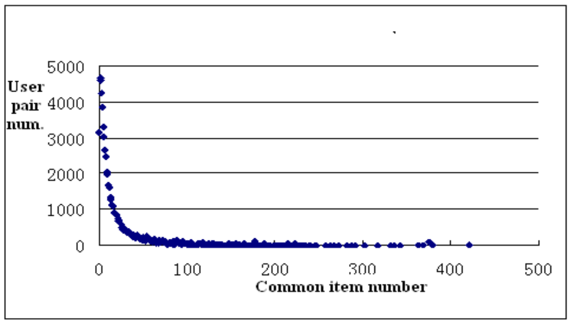


Figure 4. Num. user pair with Common item number in MovieLens

method using the hybrid approach in Section 5 is denoted as HA. The method using both of them is denoted as EIB&HA.

6.3. Results

Table 2- Table 5 show the results for all the combinations of MovieLens&Netflix and MAE&RMSE. According to the four table, both EIB and HA could improve the accuracy of the algorithm The performance of EIB is better than HA. And the combination of EIB and HA leads to better results more or less. However, the

Table 2. MovieLens&MAE

Method	k=5	k=10	k = 15	k=20
T_CF	0.925	0.918	0.916	0.912
Fill	0.9	0.86	0.85	0.86
Iterator	0.827	0.806	0.794	-
HA	0.871	0.864	0.866	0.866
EIB	0.775	0.775	0.772	0.773
HA&EIB	0.770	0.771	0.760	0.770

Table 3. MovieLens&RMSE

Method	k=5	k=10	k = 15	k=20
T_CF	1.211	1.203	1.203	1.201
HA	1.134	1.136	1.130	1.131
EIB	1.059	1.062	1.062	1.060
HA&EIB	1.054	1.055	1.051	1.052

Table 4. Netflix&MAE

Method	k=5	k=10	k = 15	k=20
T_CF	0.828	0.805	0.794	0.797
HA	0.795	0.794	0.789	0.789
EIB	0.744	0.737	0.737	0.736
HA&EIB	0.736	0.732	0.731	0.729

Table 5. Netflix&RMSE

Method	k=5	k=10	k = 15	k=20
T_CF	1.13	1.102	1.098	1.092
HA	1.084	1.081	1.079	1.077
EIB	1.019	1.017	1.018	1.016
HA&EIB	1.016	1.014	1.014	1.010

increasing of k doesn't bring much impact on the results. Sometimes it even makes the accuracy worse. than HA. And the combination of EIB and HA leads to better results more or less. However, the increasing of k doesn't bring much impact on the results. Sometimes it even makes the accuracy worse.

7. Conclusions

In this paper, an expected item bias based method is proposed to compute the similarity between users. It could fill the missing values of datasets and could be applied to users having no common items. Furthermore, a hybrid approach integrating the global rating information is also proposed. Both methods could improve the accuracy of memory based recommendation. And a combining use will lead to even better results.

References

[1] https://signup.netflix.com/global.
[2] http://www.cdnow.com.
[3] http://www.amazon.com.
[4] http://www.douban.com.
[5] http://www.digg.com/.
[6] http://www.grouplens.org/node/73.
[7] E. Rich, "User Modeling via Stereotypes", Cognitive Science, vol. 3, no. 4, **(1979)**.
[8] M. J. D. Powell, "Approximation Theory and Methods", Cambridge Univ. Press, **(1981)**.
[9] H. Wang, "The design and implementation of commericial recommendation system", Doctor thesis, University of Science and Technology of China, **(2007)**.
[10] G. Adomavicius and A. Gupta, "Towards Comprehensive Real-Time Bidder Support in Iterative Combinatorial Auctions", Information Systems Research, vol. 16, no. 2, **(2005)**.
[11] P. Melville, R. J. Mooney and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations", Proc. 18th Nat'l Conf. Artificial Intelligence, **(2002)**.
[12] J. Zhang and P. Pu, "A recursive prediction algorithm for collaborative filtering recommender systems", Proceedings of the 2007 ACM conference on Recommender systems, Minneapolis, MN, USA, **(2007)** October 19-20.
[13] J. S. Breese, D. Heckerman and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", Proc. 14th Conf. Uncertainty in Artificial Intelligence, **(1998)**.
[14] P. Melville, R. J. Mooney and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations", Proc. 18th Nat'l Conf. Artificial Intelligence, **(2002)**.
[15] M. Pazzani, "A Framework for Collaborative, Content-Based, and Demographic Filtering", Artificial Intelligence Rev., vol. 13, **(1999)**, pp. 5-6.
[16] I. Soboroff and C. Nicholas, "Combining Content and Collaboration in Text Filtering", Proc. Int'l Joint Conf. Artificial Intelligence Workshop: Machine Learning for Information Filtering, Stockholm, Sweden, **(1999)** July 31-August 6.
[17] J. Breese, D. Hecherman and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering", Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98), **(1998)**.
[18] A. L. Deng, Y. Y. Zhu and B. L. Shi, "A collaborative filtering recommendation algorithm based on item rating prediction", Journal of Software, vol. 14, no. 9, **(2003)**.
[19] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-Based collaborative filtering recommendation algorithms", Proceedings of the 10th International World Wide Web Conference, **(2001)**.
[20] M. Newman, A. Barabási and J. W Duncan, "The Structure and Dynamics of Networks", Princeton, NJ: Princeton University Press, **(2006)**.
[21] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model", Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08), Las Vegas, Nev, USA, **(2008)** August.
[22] http://www.igvita.com/2006/10/29/dissecting-the-netflix-dataset.