A High Precision Recommendation Algorithm Based on Combination Features

Xinhui Hu[1,2], Qizhi Liu[1(✉)], Lun Li[1], and Peizhang Liu[2]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
lqz@nju.edu.cn
[2] ZTE Corporation, Shenzhen/Nanjing, China

Abstract. Conventional recommendation systems often use binary relations matrices, which could not represent raw data sets efficiently and uniformly. The graph model can represent multiple relationships and form a unified standard of the feature space to recommend the candidate items. Existing graph-based work is generally based on the path to establish the feature space of the data, only concerned about the impact of an item on the description of the user. Utilizing combination features to construct user profiles, this paper concentrates on the contribution maked by combination of items and designs a user-based collaborative filtering algorithm (CFC), and validates the validity of the algorithm in the prototype of the proposed system. The experimental results show that the recommendation algorithms can significantly improve accuracy of the recommendation.

Keywords: Recommendation system · Graph model · Combination features Collaborative filtering

1 Introduction

The recommendation system can build user profiles based on user behavior habits, such as past click, audition, read, collect, and purchase, as well as infer the user's possible future behavior to help users filter a large amount of irrelevant or low-relevant information. The development of e-commerce, social networking, Internet advertising, search engines, and other areas [19] has promoted the application of recommendation systems and has exposed the problems of traditional recommendation technologies.

With the further popularization of the Internet, users create a large number data while benefiting from information. The growth of data volume is very rapid and highly inline. It is a big challenge to how to make use of numerous highly inline data to perform complex queries. The dataset associated with the recommender system is usually composed of triples like (users, purchase, items) or (users, have, attributes) or (items, have, attributes), which are similar to the representation of RDF, so it is appropriate to describe these semi-structured data in a graph model. The graph model is not forced to rely on the intermediate index structure, has a more flexible data model, can easily improve the recommended system engine when data types and data sources change, and is easier to retrieve association data, suitable for a large number of highly inline data

C. Liu et al. (Eds.): DASFAA 2018, LNCS 10829, pp. 74–86, 2018.
https://doi.org/10.1007/978-3-319-91455-8_7

management. According to existing cases, the graph database is 1000 times more efficient in some applications than a relational database with the same information representation capability, and the code amount is reduced by 10–100 times [38].

The existing recommendation method often uses a binary relationship matrix to represent the data model. However, the binary relationship matrix does not effectively and uniformly express the raw data set because elements in the natural world have multiple relationships. Furthermore, in the graph model, the attributes of node elements can also be integrated into the graph model as nodes, so the graph model greatly enriches the expressiveness of the data set model. For example, users and items can constitute a bipartite model, and users, items, and tags can constitute a three-part graph model. By building a heterogeneous graph model, some different types of nodes and edges are linked to the graph model, that can express multiple relationships.

In fact, the attributes of node elements can also be integrated into the graph model as nodes. The rich expressive ability of the graph model can unite multiple recommendation methods (such as collaborative filtering, content-based filtering) into a complete model, and form a feature space with uniform standards, from which to obtain the required multiple elements for recommendation. For example, there are 3 users {*Alice, John, Tom*} and 3 movies {*AngelsAndDaemons*, *TheDaVinciCode*, *TheShining*}, 7 edge types {*literaryGenre, subject, author, noblework, influencedBy, subsequentWork, broader*}. From the example, semantic path features can be extracted for each user for the recommendation algorithm. In addition, the graph model itself can also provide potential features to enrich the node's feature properties. Therefore, the ability to express the graph model should be fully demonstrated.

Related graph-based research usually uses two types of collections (users, items) or three types of collections (users, items, tags) as models, and the elements of a single collection are independent of each other. This model does not fit well with the complex models of the real world for ignoring many valuable information. Using graph data models can create very flexible data patterns. In the recommendation method based on the graph [12, 17], user profiles are usually constructed only by a single item, which may generate noise and drop the recommendation accuracy. The contributions of this paper:

- We propose combination features based on graph data model.
- We build heterogeneous graphs and feature space using combination features to compute user profiles, and make full use of the advantages of graph data model in terms of strong expressiveness.
- We design the related recommendation algorithm based on multiple relation matrix and the combined features.
- We use HetRec MovieLens dataset in the experiments to evaluate the performance of our method. The results show superior performance over other graph-based recommendation algorithms and benchmark algorithms in terms of precision and recall.

In the rest of this paper, we introduce the research background, main contents and objectives, as well as give the definition and research basis of related problems of the recommendation system in Sect. 2. In Sect. 3, we elaborate the combination features and their formal representations, and point out the problems that may arise from

extracting features from a single item. We detail the feature model of the user-item pair based on the combination feature and the feature index structure which can reduce the duplicate calculation. In Sect. 4, we describe the user-based collaborative filtering recommendation algorithm (CFC) which is based on the combined features. We present the result of our experiment and the process of our algorithm implementation in Sect. 5. Finally, we discuss our conclusions.

2 Research Foundation and Problem Definition

For a given user set U and item set I, $|U| = m$, $|I| = n$, forming a user-item pair of $|U \times I| = m * n$, assuming that p ($p < m^* n$) user-item pairs have already been scored, then the recommendation system is a system that can predict the score of the remaining $m^* n - p$ unknown-rated user-item pairs. The ultimate goal is to give each user a list in which all the items are sorted by the score and to minimize the system losses.

Definition 1. Top-k recommendation: For a given user u, the top-k recommendation refers to selecting the top k items by the system as the recommendation list which have not been scored by u.

If $k = 1$, i.e. top-1 recommendation, the hit rate will be very low. Therefore, $k = 5$, $k = 10$ or $k = 15$ is usually used to evaluate the algorithm. If the test set's sample appears in the top-k recommended list, it is counted as a hit.

In the applications of television programs, movies, music, news, books, labels, etc., there are two main types of recommendation modes, collaborative filtering (CF) and content-based filtering (CBF). CF uses a user's past behavior (collection or evaluation) to build a model to determine which users have similar behaviors or interests with the user, so as to predict the user's possible interest points. CBF uses a series of similar discretized features of an item to recommend other items. These two methods are often mixed, that is, mixed recommendation. However, these two modes are very vague, leaving a lot of space for different algorithms to use [12]. In addition, there are many other recommended methods, such as that based on inherent rules, based on mutual relations, based on graphs, based on global relevance etc. The graph -based recommendation method is a recent research hotspot.

Definition 2. Graph data model: The graph data model of the recommendation system refers to a heterogeneous directed graph $G = (V, E, TV, TE, \varphi V, \varphi E, L, \varphi EW)$, where V is a set of finite nodes, $E \in V \times V$ is a set of finite edges, TV is a set of finite node types, TE is a set of finite edge types, φV: $V \rightarrow TV$ is a mapping function from node to node type, φE: $E \rightarrow TE$ is a mapping function of edge-to-edge type. Each node vi has a corresponding attribute table $li \in L$, li refers to a set of pairs (attribute, value). For each attribute, a li contains one corresponding attribute value pair at most. φE: $E \rightarrow w$ means that each edge can correspond to a weight value w, such as φEW $(ei) = wei$.

The recommendation system can represent various types of data sets (e.g., relation, XML, RDF triples) through heterogeneous graph-based models. A graph data model is built from a data set and integrates multiple different data sets to form a larger, richer

data set by linking the same entities [12]. Typical approaches include random walks [15], matrix decomposition [16], and sort-based learning [17].

3 Combination Features

3.1 Feature Space Construction

The graph-based recommendation method is often focused on the path-based graph element. That is, starting from a single item of a user to extract features for the user, so that only the impact of a single item constructs the profile, which can easily cause noise problems in collaborative filtering. For example, in Fig. 1, we predict the score of the dashed line based on the score of the solid line, or recommend *A* for *J*. In that case, only the score data of *B* is helpful for the recommendation prediction (without content recommendation, ignoring the connection between other items and *J*) because only *B* also has a score of *J*. If *A*–*K* and *A*–*L* are used as the basis for the recommendation, since *C* and *D* have score records for *K* and *L*, respectively, *C*–*K* and *D*–*L* are also calculated in this single item model. However, for the desired results (*A*–*J*), it is actually better that these two records are not considered in the calculation. In the worst case, because the score of *C* and *D* are taken into consideration, *M* is recommended to *A*.

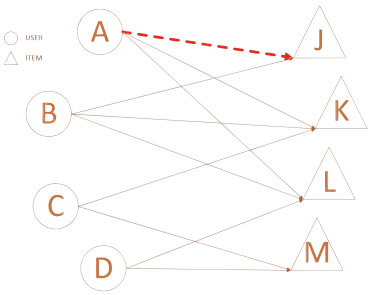


Fig. 1. Noise problems with single item features

This paper proposes a solution to the combination of items and the concept of combined features for the recommendation system. As shown in Fig. 2, *K* and *L* are combined together (coarse line in the figure). That is, only when the user has *K* and *L* at the same time, the two can work together to portray users, thus strengthening the similarity between *A* and *B*, to increase the likelihood of accurately recommending *J* to *A*.

Definition 3. Combination features: Given the user-item pair set *S* of the recommendation system, suppose that the number of related items from the *S* for the user *u* is *n*, and we take the *m* items ($m < n$) as the basis combination features of *u*, namely the combination features: $CombFeature(u) = Combination(f, u, m, n) = \{f(\{item1, item2, \ldots, itemm\}), f(\{item1, item2, \ldots, itemm-1, itemm+1\}), \ldots, f(\{itemn-m+1, itemn-m+2, \ldots, itemn\})\}$, where *f* refers to a map of the *m* items to the final feature.

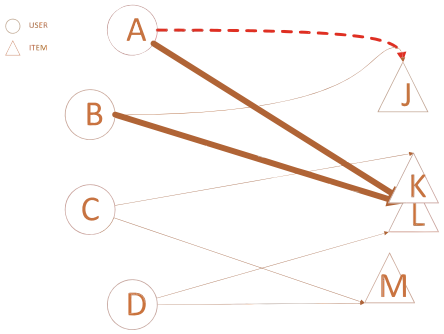


Fig. 2. Solution for combination of items

Different from focusing on the contribution of a single item, the combination features use multiple items to build a profile of the user and build the user's feature space.

Definition 4. Feature space: Given the set S of user-item pairs in a recommendation system, all elements (users or items) of S or features that can be extracted constitutes a feature space. Eg., the recommendation system S contains $user1\text{-}item1 \Rightarrow \{feature1, feature2\}$, $user2\text{-}item2 \Rightarrow \{feature2, feature3\}$ and $user3\text{-}item1 \Rightarrow \{feature3, feature4\}$, the feature space of S is $featureSpace = \{feature1, feature2, feature3, feature4\}$.

The original number of combination feature is the combination number. For ranking learning algorithms, effectively controlling the size of feature space is very important. So we give a mapping function to reduce the size of the feature space.

3.2 User-Item Feature Construction

Combination features generally represents the user's taste and preferences. However, the intermediate process of the algorithm based on the ranking learning is to predict the value of the user-item, so we need to rebuild the feature space for the user-item.

In order to predict the value of the user-item, each user-item pair needs to be given a feature space about the item. If the user's characteristics are (likes romantic comedy, likes American movies) and the characteristics of the items are (romantic comedy, American movies), then the characteristics of such a pair of user-items are (like romantic comedy, like American movies, romantic comedies, American movies), then such user-items get relatively high scores. On the contrary, if the characteristics of the item are (war, Chinese film) and the resulting user-item characteristics are (like to waste a comedy, like American movies, war, Chinese movies), the predicted score is relatively low.

One of the basic assumptions of this paper is that a user has a certain preference for similar types of movies with the movie which is rated by the user. Of course, similar types of movies may resulted in scoring errors for the quality of the movie is not the same. Therefore, we introduce popularity, which refers to how much an item is rated by a user. That is, if an item is connected with more users, the item with the higher degree of welcome. Here we give the definition of the popularity of the item collection.

Definition 5. Popularity of the collection of items: Given collection of items set_{items}, the popularity of set_{items} is the centrality of set_{items}, represent the number of users who are also associated with set_{items}, referred to as $Count(set_{items})$.

Definition 6. User-item combination feature: Given the user-item pair set S of the recommendation system, suppose there are p users and n items for the user u. For the item i, we first take the feature vector a of u. then compute the feature vector b of i. Assume that the user's feature space is {*feature*1, *feature*2,…, *featurep*}, and the feature space of the item is {*featurep*+1, *featurep*+2,…, *featuren*}. The user-item feature space is a combination of feature space {*feature*1, *feature*2,…, feature*n*}. And the feature vector of *u*-*i* is [*a*,]. which is a concatenation of *a* and *b*. In order to take into account the popularity, the value of each dimension feature is changed according to the corresponding popularity.

Suppose there is a user and that the item sets have three feature modes: *P*1, *P*2, *P*3 after they have been mapped, which constitute the user's feature space (*P*1, *P*2, *P*3). The feature space of the items is (*USA*, *Comedy*, *Donald Petrie*). Then we get the feature table as shown in Table 1. Since the users are the same, the features of the four user-items are all three-dimension (1, 1, 1), and the items features are different.

We also constructed a feature index structure for the dataset to organize the feature space of all users and user-items, speeding up the predictor and avoiding redundancy calculation.

Table 1. Example of feature representation

Users - Items	*P*1	*P*2	*P*3	USA	Comedy	Donald Petrie
User-A	1	1	1	1	1	1
User-B	1	1	1	0	0	0
User-C	1	1	1	1	0	0
User-D	1	1	1	0	1	0

4 CFC Algorithm

In the collaborative filtering algorithm based on the combination feature (CFC), to predict user u's score on item i, supposing u's feature set is $Sf = \{F1, F2, F3, \ldots, Fp\}$, we first traverse the feature set in the training data, and find all users with each feature, and determine for each user with the same feature whether or not i has a score. In this way, while looking for similar users, determine the contribution of each feature to i and finally get a prediction score for u on i. This method does not directly calculate the user's similarity, but indirectly looks for similar users through the features of each dimension, and adds up the predicted values of user-items pairs. Due to a coarse-grained feature model, the features of each dimension have been able to express enough similarities, the approach is feasible.

The input of CFC is a combination feature index T, the user u of the recommendation service, the parameter k of the top-k, the combination feature parameter (n, m). The output is the recommended list of u. Algorithm pseudo code:

Algorithm 1：CFC

CFC(Feature Table T , User u , k , n ,m)

Input: user set U , item set I , Users u need to provide recommendation service , combination feature table T , parameter k of top-k, combination feature parameter n , m

Output: top-k recommendation list l of ;

1: Initialization: X , X refers to u 's scoring array;

2: Find the corresponding feature of u in T to get the feature array F ;

3: for Feature : F do

4: for User : U do

5: if $isFeature(u, f)$ then

6: for Item : I do

7: if $isConnecte(u, i)$ then

8: $updat(X)$;

9: end if

10: end for

11: end if

12: end for

13: end for

14: Sort X in descending order;

15: $l \leftarrow$ top-k elements of X ;

16: return l

In Algorithm 1, the way of updating (line number 8) is free to design according to statistical results. This paper uses formula 1 to update the scoring array:

$$X(i) = Xu(i) + countfui/countfu \tag{1}$$

Where *countfui* refers to the number of users associated with an item in a user with feature *f*, *countfu* refers to the number of users.

The effect of parameters (*n*, *m*) in the algorithm on dataset MovieLens is shown in Figs. 3 and 4.

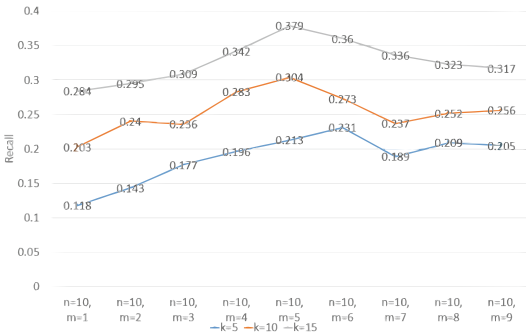


Fig. 3. Recall rate affected by m value

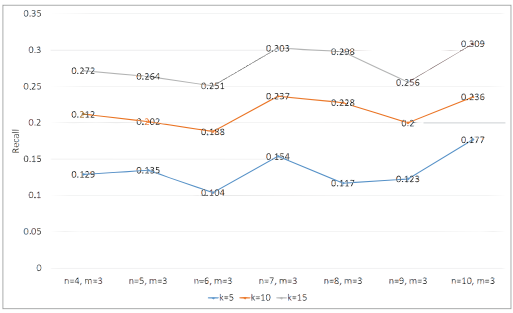


Fig. 4. Recall rate affected by n value

5 Experimental Results Analysis

The recommendation system can be roughly divided into three modules: user profiles modeling, item profiles modeling and recommendation algorithm implementation. This paper mainly focuses on the user profiles modeling and the recommendation implementation modules. In the item modeling, the basic item features are extracted according to their attributes, which is used as an auxiliary process for user-item feature construction.

5.1 Construction of Recommended Systems

In a recommendation system, at first, there should have a user list, an item list and a binary relationship between them, i.e. the original bipartite model, which is the data core of the system. Usually, a corresponding user profiles library and an item profiles library are also needed. After that, a recommendation algorithm framework is built on top of the two profiles libraries. Then the user can get the recommendation service. The user can score the recommended items and the system can feedback and update the data core to re-adjust the user and item library. Figure 5 shows the operational flow of the recommendation system.

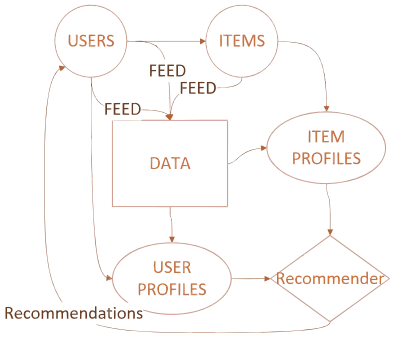


Fig. 5. Recommended system model

5.2 Baseline

- Popularity-based recommendations

Popularity-based recommendation (POP) refers to sorting the list based on the score of the item. It is not a personalized recommendation method. This paper uses a more reasonable Popularity-based recommendation method (POPN) as a baseline. POPN depends on the number of high-scoring items to determine its popularity. Its effect is much better than that of POP.

$$\text{POP}: \ Scoreitem = Coun\{user|user.itemList.contains(item)\} \tag{2}$$

$$\text{POPN}: \ Scoreitem = Coun\{user|user.itemList.contains(item) \ \&\& \ (user, item) > ratingt\} \tag{3}$$

Where *ratingt* is the threshold.

- Non- normalized Neighborhood collaborative filtering

Non-normalized Neighborhood CF (NNCF) is an improvement over traditional k-nearest neighbor collaborative filtering (KNNCF), which usually better than KNNCF in top-k recommendation. The top-k recommendation system aims to sort the users according to their attractiveness, so it is not necessary to normalize the score [8].

- PureSVD

PureSVD (PSVD) is a matrix decomposition method based on collaborative filtering. It reduces the rank of the scoring matrix by single value decomposition [8].

- Personalized PageRank

Personalized PageRank (PRANK) is a random walk method with reset.

- Path Guide

Path Guide (PG) [9] refers to the method of PRANK with semantic path guidance. It is a typical graph-based recommendation method. Through the process of random walk and iterate, PG can get a stable probability value.

5.3 Experiment Settings and Result Analysis

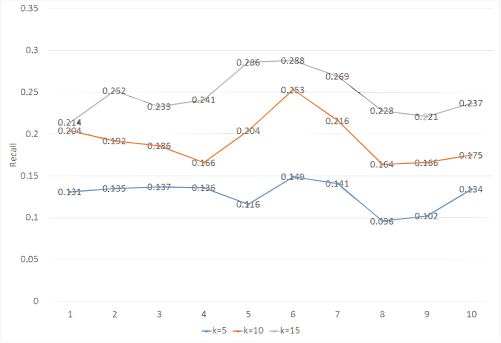
The test methods used in this paper are similar to those in [6, 15]. First, the user-item scoring list is sorted according to the time stamps, and divided into two subsets according to the scoring time. The early 90% is used as the training set. T high score records in the remaining 10% used as a test set, the items contained in which are the most relevant items to the user. Then, we randomly select R items that have not been scored for each user and rank them. Finally, evaluate the prediction according to i's position p in the *Rank*. If $p <= k$, the prediction is considered as a hit, otherwise, it is considered as a fail. The overall recall rate of the recommended system is calculated according to the following formula:

$$recal(k) = \#hits/T \tag{5.3}$$

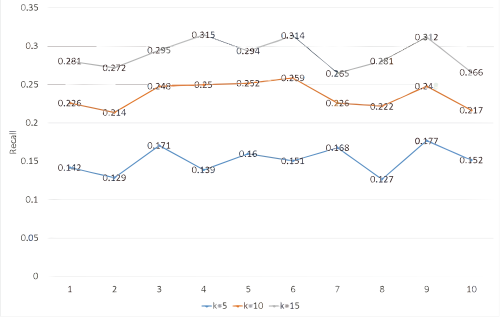
Where *#hits* refers to the number of hits. The overall recall rate is the proportion of hits in the T.

This paper uses the movie data set HetRec MovieLens as experimental data set. The entire dataset contains a total of 2217 users and 10197 items (movies). Each item has 5 attributes: director, category, actor, country, and label. The total data set contains 855,598 scoring records, of which the first 770031 records, sorted by time stamp, are used as training data, and 85567 records is taken out as a test case. The records are aggregated according to the user in training, i.e., one user corresponds to one query. During the test, we randomly select one record and 1000 items that were not scored by the user in the record to form a test query and count the hits.

This paper constructe the user's preferences from favorite items, not consider the items that the user does not like. For each user, a sorting list based on popularity recommended. The experimental results show that the POPN is better than the POP. Compared with POP, the average recall rate in POPN increased by 0.025(k = 5), 0.043(k = 10), 0.043(k = 15), respectively (Fig. 6).



(a) POP 10 Recommended Results Recall



(b) POPN 10 Recommended Results Recall

Fig. 6. Experimental results based on popularity

Table 2. Recommended precision

Recall(k)	CFC	POP	POPN	NNCF	PSVD	PRANK	PG
K = 5	0.213	0.127	0.152	0.181	0.182	0.152	0.213
K = 10	0.304	0.193	0.236	0.262	0.270	0.238	0.305
K = 15	0.379	0.247	0.290	0.322	0.339	0.300	0.371
Precision(k)	CFC	POP	POPN	NNCF	PSVD	PRANK	PG
K = 5	0.282	0.119	0.151	0.179	0.180	0.150	0.281
K = 10	0.335	0.185	0.224	0.254	0.262	0.237	0.335
K = 15	0.397	0.239	0.288	0.320	0.331	0.308	0.392

In order to compare the proposed method with those baseline methods in Sect. 5.2 intuitively, Table 2 shows the best results of those methods in the optimal parameters.

From Table 2, we can see that the precision of CFC (n = 10, m = 5) algorithm is far superior to all benchmark methods except PG and close to PG.

In addition, CFC does not use any other information on users or items except ID. That shows CF method based on the combination features is effective.

The running time of the CFC recommendation algorithm increases with the number of combination features and the number of users corresponding to each feature. When taking 9 combinations of 10 items, the average number of users corresponding to the average item combination is the smallest, which is also consistent with statistics law. While taking 1 combination of 10 items, the number of users corresponding to the average item combination reaches 214.2, so the running time is increasing.

In the case of $k = 5$, we can see that the CFC ($n = 10$, $m = 9$) algorithm is much better than all other baseline methods. The performance is also better than other baseline methods in the case of $k = 10$ and $k = 15$. The CFC algorithm of this paper calculates the similarity between users based on the combination of features, and predicts the user's relevance to the items, which is used to rank and produce recommendation results. Due to the relatively large feature space, the time consumed by the sorting process is proportional to the number of user's features and the average number of users with the same features. Fortunately, when $n = 10$ and $m = 9$, the result is the best (Fig. 7).

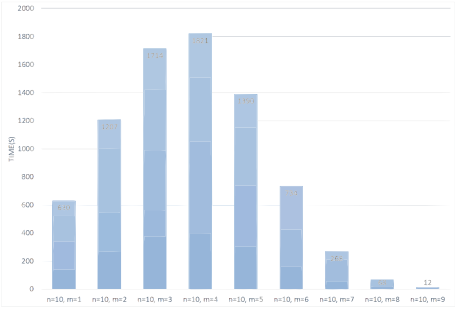


Fig. 7. CFC runtime

6 Discussion and Conclusion

Based on the graph model, this paper proposes the combination features to construct user profiles. Based on the user's features, the user-based Collaborative Filtering Recommendation Algorithm (CFC) is designed. In general, collaborative filtering based on items requires the condition that one item corresponds to multiple users, while the combination features model just requires multiple items correspond to one user, regardless of the individual item of the user. The CFC first looks for similar users based on the coarse-grained combination of features and scores the relevance of the items based on a similar user list to form a recommendation list. Finally, this paper uses the off-line evaluation method and conducts experiments in the recommendation system. The experimental results show that the effectiveness of the recommendation based on the combination of features. In the MovieLens data set, the best case is better than all comparison methods.

In this paper, the commonly used top-k recommended performance evaluation is used to determine the accuracy. Further work will be based on more evaluation indicators to conduct experiments. The experiments will be conducted on more types of data sets to study sorting based learning. Due to the reasons of data sets, the degree of heterogeneity of the graph data model in this paper is limited, and it is necessary to further study the recommendation method based on the high degree of heterogeneity graph model.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Grant No. 61572247).

References

1. Nguyen, H., Dinh, T.: A modified regularized non-negative matrix factorization for movielens. In: 2012 IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), pp. 1–5. IEEE (2012)
2. Li, Q., Kim, B.: Constructing user profiles for collaborative recommender system. In: Advanced Web Technologies and Applications, pp. 100–110 (2004)
3. Huang, Z., Chung, W., Ong, T.H., et al.: A graph-based recommender system for digital library. In: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 65–73. ACM (2002)
4. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
5. Takács, G., Pilászy, I., Németh, B., et al.: Scalable collaborative filtering approaches for large recommender systems. J. Mach. Learn. Res. **10**, 623–656 (2009)
6. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 39–46. ACM (2010)
7. Neo Technology. Powering Recommendations with a Graph Database. https://neo4j.com/

8. Vicknair, C., Macias, M., Zhao, Z., et al.: A comparison of a graph database and a relational database: a data provenance perspective. In: Proceedings of the 48th Annual Southeast Regional Conference, p. 42. ACM (2010)
9. Linden, G.D., Jacobi, J.A., Benson, E.A.: Collaborative recommendations using item-to-item similarity mappings: U.S. Patent 6,266,649[P], 24 July 2001
10. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
11. Wikipedia. https://en.wikipedia.org/wiki/Recommender_system
12. Beel, J., Gipp, B., Langer, S., et al.: Research-paper recommender systems: a literature survey. Int. J. Digit. Libr. **17**(4), 305–338 (2016)
13. Schafer, J., Frankowski, D., Herlocker, J., et al.: Collaborative Filtering Recommender Systems. In: The Adaptive Web, pp. 291–324 (2007)
14. Wang, J., De Vries, A.P., Reinders, M.J.T.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 501–508. ACM (2006)
15. Lee, S., Park, S., Kahng, M., et al.: Pathrank: a novel node ranking measure on a heterogeneous graph for recommender systems. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 1637–1641. ACM (2012)
16. Yu, X., Ren, X., Sun, Y., et al.: Personalized entity recommendation: a heterogeneous information network approach. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, pp. 283–292. ACM (2014)
17. Islam, M.S., Liu, C., Li, J.: Efficient answering of why-not questions in similar graph matching. IEEE Trans. Knowl. Data Eng. **27**(10), 2672–2686 (2015)
18. Ma, S., Li, J., Hu, C., et al.: Big graph search: challenges and techniques. Front. Comput. Sci. **10**(3), 387–398 (2016)
19. Noia, T.D., Ostuni, V.C., Tomeo, P., et al.: Sprank: semantic path-based ranking for top-n recommendations using linked open data. ACM Trans. Intell. Syst. Technol. (TIST) **8**(1), 9 (2016)