Graph-Based Hybrid Recommendation Using Random Walk and Topic Modeling

Hai-Tao Zheng, Yang-Hui Yan, and Ying-Min Zhou

Tsinghua-Southampton Web Science Laboratory,
Graduate School at Shenzhen, Tsinghua University, Beijing, China
zheng.haitao@sz.tsinghua.edu.cn,
{yanyh13,zhou-ym14}@mails.tsinghua.edu.cn

Abstract. In this paper, we propose a graph-based method for hybrid recommendation. Unlike a simple linear combination of several factors, our method integrates user-based, item-based and content-based techniques more fully. The interaction among different factors are not done once, but by iterative updates. The graph model is composed of target user's similar-minded neighbors, candidate items, target user's historical items and the topics extracted from items' contents using topic modeling. By constructing the concise graph, we filter out irrelevant noise and only retain useful information which is highly related to the target user. Top-N recommendation list is finally generated by using personalized random walk. We conduct a series of experiments on two datasets: movielen and lastfm. Evaluation results show that our proposed approach achieves good quality and outperforms existing recommendation methods in terms of accuracy, coverage and novelty.

Keywords: hybrid recommendation, random walk, topic modeling, sparsity, novelty.

1 Introduction

With an exponential growth of information available on the Internet, it has become increasingly important to help people get personalized services. Recommendation systems, which are designed to solve the problem by analyzing users' preference, are studied extensively in prior research.

The most widely used recommendation techniques are collaborative filtering-based (CF) and content-based (CB) methods. In the user-based CF [2], target user is recommended new items that are rated by his similar-minded neighbors. While in the item-based CF [3], the items similar to target user's historical records will be recommended. User-based CF tends to recommend popular items, which, however, harms the overall diversity for all users. Item-based CF can boost long-tail items, but decrease individual diversity. CB usually calculates the similarity between constructed user profile and item contents, which is often used to help improve the performance of CF methods. By considering both the collaboration and the content, hybrid recommendation technology can achieve the benefits of different methods. But a simple linear combination of two factors in the user similarity calculation or the results adjustment can not perform effectively, which even decreases the precision for specific users.

Graph-based recommendation model [12, 14] is flexible, which can make a good integration of varieties of contextual information. A recommendation list is generated by using personalized random walk(RWR) or hitting time in the graph. However, in the existing graph-based methods, random walk is usually conducted in the whole graph. Even a subgraph constructed from depth-first or breadth-first search is still very large. This practice not only causes high computational complexity, but also introduces too much noise, affecting the performance.

In this paper, we propose a hierarchical graph model for hybrid recommendation, in order to combine different techniques together appropriately and maintain small computational complexity. When recommending for the target user, we only select relevant information from each factor to construct a concise graph. Thus the running time is greatly reduced and irrelevant noises are avoided. On the other hand, as the collaboration and content factors are combined more naturally by using topic modeling and iterative interaction in the well-defined structure, our accuracy is highly improved. Finally, we can alleviate the data sparsity as random walk-based ranking allows us to utilize indirect relationships between graph nodes. And we can boost long-tail items, as long as they contain the same topics target user prefers.

In conclusion, our main contributions are listed as follows:

- We propose a novel hierarchical graph model for hybrid recommendation, which integrates different factors iteratively. And we systematically study our well-defined data fusion graph structure and justify its rationality.
- Our method can alleviate the data sparsity and boost long tail recommendations that many existing methods suffer. Also, we can cover the majority of distinct items in data corpus.
- We conduct enough experiments on two datasets. Results show that our method performs better in three measures especially in sparse dataset.

The rest of this paper is organized as follows. Section 2 presents some related work on recommendation. We give a detailed description of our proposed method RWR-UIC in section 3. Experiments and analysis are included in section 4. Finally we conclude the paper with remarks of our work in section 5.

2 Related Work

Recommendation systems are basically divided into two categories: CF-based and content-based(CB). CF [1] explores user-item rating matrix and can be further classified into user-based CF and item-based CF. User-based CF [2] assumes that similar users express similar interests in future items. Items rated by similar-minded neighbors are recommended to the target user. User similarities are calculated based on ratings. [3] proposed an item-based method. They recommend new items which are similar to the items target user has rated. Items similarities are also calculated using rating matrix. In CF, usually a small subset of users or items are used as neighbors. CF is simple in training phase and can be used in many domains such as news and multimedia. The most important feature of CF is content-independent [11]. But if ratings are sparse, Standard CF methods result in poor results. Content-based methods [6] try to construct user's profile using items' contents. To generate recommendations, Similarity between candidate item's contents and the constructed user profile is calculated. CB does not suffer from rating sparsity, but a big drawback is that it ignores the implicit associations between users, which leads to poor results purely using CB. Hybrid recommendation methods [10] can obtain the advantages of both methods by considering collaboration and content at the same time, but, usually a simple weighted combination of two factors cannot perform effectively.

Latent factor models have been successfully applied to recommendation. For example, [8] proposed a three-layer aspect model in which ratings and contents are combined in a probabilistic way, to address the cold-start problem. Matrix factorization was developed in [9,20]. It decomposes user-item rating matrix into low latent space. To predict the missing score, we just need to multiply the latent vectors of candidate item and target user. However the latent space doesn't have evident interpretation for human beings. Additionally, if training set is rather sparse, factorization may suffer from overfitting.

Graph-based methods are getting more and more attention recently. By setting a biased probability of jumping to the starting nodes, RWR is very useful for personalized recommendation [13]. [14] studied a click-through bipartite graph for a series of applications including recommendation task. [16] proposed a novel recommendation method which performs random walk on an items' graph, where the edges denote similarities between items. [7] adopted a multi-layer graph model for personalized query-oriented reference paper recommendation, but incorporating too many terms may make the graph extremely large. [12] proposed a random walk-based model which combines users, items, tags, social relationship into the whole graph. There is one thing in common in the above graph-based recommendation models: To generate recommendations, random walk is usually conducted in the whole graph. This practice not only leads to high computational complexity but also brings unnecessary noise. In this work, we just construct a sub-graph, which contains information mostly related to target user. collaboration and content more integrated more naturally and fully.

3 Proposed Method

The framework of our proposed method **RWR-UIC**(**R**andom **W**alk with **R**estart which combines **U**ser-based, **I**tem-based and **C**ontent-based factors) is shown in Fig.1. In the bottom part, concept mining on items' contents and similarity calculation between users and items are done offline to save online response time. In the online procedure, when a user request recommendations, we firstly construct a concise graph as illustrated in Fig.3 for him, then top-N list is generated by using personalized random walk.

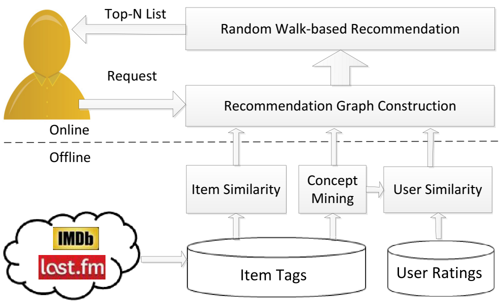


Fig. 1. System framework

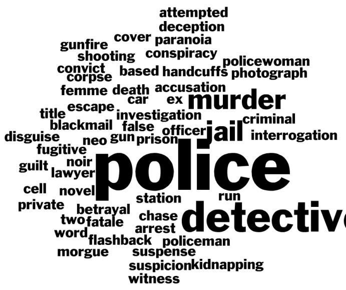

attempted deception gunfire cover paranoia shooting conspiracy policewoman convict corpse based handcuffs photograph femme death accusation car ex murder title escape investigation criminal blackmail false disguise neo gun prison officer jail interrogation fugitive guilt noir lawyer police cell novel station run private betrayal two fatale chase arrest detective word flashback policeman morgue suspense suspicion kidnapping witness

Fig. 2. Sample concept tag cloud

3.1 Concept Mining

A document may involve multiple topics. Using the words alone, we may fail to find conceptually related documents that use different wordings. Topic models like PLSI [4] and LDA [5] treat document as a probabilistic mixture of topics and estimate the doc-topic distributions from a collection of documents unsupervisedly. When the corpus is not particularly large, LDA can effectively prevent overfitting because it adds priors to the parameters. In this work, we adopt LDA topic model and use gibbs sampling [18] to infer the topic distributions.

In our rating corpus, items contents are not available. However, annotation tags in the social media websites, such as lastfm and imdb, are good descriptions of artists' musical style or films' storyline. By crawling these tags, we can construct items' contents. However, the uncontrolled tagging behavior in websites results in tag redundancy and ambiguity. So LDA is used to help us capture the co-occurrence between related tags and extract the items' topic distributions p(t_k|i). The results of this step are used for both user profiling and recommendation graph construction. A sample tag cloud is generated by using top 30 terms of an topic. We can see that words are related to "crime", as illustrated in Fig.2.

3.2 User Profiling

Formally, suppose that there exist a set of users U=$\{u_1, u_2, ..., u_M\}$ and a set of items I=$\{i_1, i_2, ..., i_N\}$. For each user u that belongs to U, a list of items are available with the corresponding rating $R_{u,i}$. To generate recommendations, we start with constructing user's preference profile as a two-attribute tuple $\overrightarrow{u} = \{T, L\}$. T represents u's interest distributions in content topics, in the format of vector $\{p(t_1|u), p(t_2|u), ..., p(t_K|u)\}$. L denotes a list of rated items $\{i_1, i_8, ...\}$ that u has shown interest in. We update T by the following procedure. T is firstly initialized to $\overrightarrow{0}$. Then $\forall$ item i which has been rated by u, we multiply $R_{u,i}$ with i's topic distribution $\{p(t_1|i), p(t_2|i), ..., p(t_K|i)\}$, and add the multiplied vector into T. Finally, $p(t_k|u)$is normalized by $\sum_{k=1}^{K} p(t_k|u)$. Based on the constructed $\overrightarrow{u}$, we use Eq.(1) to calculate user similarity. $\delta \in [0, 1]$.

$$S(u_1, u_2) = \delta \cdot sim(T_{\overrightarrow{u_1}}, T_{\overrightarrow{u_2}}) + (1 - \delta) \cdot sim(L_{\overrightarrow{u_1}}, L_{\overrightarrow{u_2}}) \quad (1)$$

Similarity is usually calculated by cosine Eq.(2) or Pearson correlation Eq.(3). $\overrightarrow{r_a}$ and $\overrightarrow{r_b}$ are two vectors. $\bar{r_a}$ and $\bar{r_b}$ are the mean values of respective vectors. $sim_{a,b}$ is the similarity between $\overrightarrow{r_a}$ and $\overrightarrow{r_b}$.

$$sim_{a,b} = \frac{\sum_{i=1}^{N} r_{a,i} \cdot r_{b,i}}{\sqrt{\sum_{i=1}^{N} r_{a,i}^2 \cdot \sum_{i=1}^{N} r_{b,i}^2}} \tag{2}$$

$$sim_{a,b} = \frac{\sum_{i=1}^{N}(r_{a,i} - \bar{r_a}) \cdot (r_{b,i} - \bar{r_b})}{\sqrt{\sum_{i=1}^{N}(r_{a,i} - \bar{r_a})^2 \cdot \sum_{i=1}^{N}(r_{b,i} - \bar{r_b})^2}} \tag{3}$$

To calculate Pearson, we need to isolate co-occured items of two vectors in advance. Additionally, if the number of co-occured items between two vectors is small, penalizing the correlation score obtained from very few evidence can improve prediction accuracy. More specifically, set a threshold ϵ. If the number of co-rated items τ is less than ϵ, we multiply the score by $\frac{\tau}{\epsilon}$ [12]. In our experiments, $sim(T_{\overrightarrow{u1}}, T_{\overrightarrow{u2}})$ is calculated using Eq.(2), $sim(L_{\overrightarrow{u1}}, L_{\overrightarrow{u2}})$ is calculated by Eq.(2) in movielen and Eq.(3) in lastfm for getting better performance. It is also common to use only a subset of users for both performance and accuracy when making predictions. In our experiments, A constant number is set.

3.3 Recommendation Graph Construction

Algorithm 1 describes the process of graph construction by using selected information which is highly related to target user. The graph has four layers which combine user-based, item-based and content-based factors together as illustrated in Fig.3. Neighbor layer contains the target user labeled with blue and his similar-minded neighbors. In candidate layer: On one hand, we add items rated by those similar-minded users(user-based); On the other hand, we also add items labeled with blue that are similar to target user's historical records from steps 8 to 14(item-based). Namely, we treat target user himself as a fake neighbor with candidate items whose number is equal to the average rated items of all real neighbors. Items similarities are calculated using Eq.(2) based on content tags. Tags' importance are weighted by classical $TF * IDF$. It is easy to see that the items from two factors may be overlapped. In candidate layer, we remove the items target user has rated and set them as history layer labeled with yellow(content-based). Every connection in graph G is bi-directional.

In the graph, the left part acts as collaboration factor; while the right part represents content factor. Collaboration factor influences candidate items by direct rating link. Since simple term matching is not precise for linking items because of existing synonyms and polysemants. We use intermediate topic layer to help content factor propagate its influence to the candidate items. Top-N recommendation list is finally generated from the candidate items in the second layer by using personalized random walk.

Algorithm 1. Recommendation Graph Construction

```
   Input  : Target user $\mu$, Item topics $T[N][K]$, User-Item rates $R[M][N]$
   Output: $G(V, E, W)$ : a set of triples:$\langle node_a, node_b, weight\rangle$
 1 begin
 2     Init : $TotalCount \leftarrow 0$, $Candidate\_Set \leftarrow \emptyset$, $Item\_Map \leftarrow \emptyset$, $G \leftarrow \emptyset$
 3     for $u \in \mu.Neighbors$ do
 4         for $i \in R[u] \& i \notin R[\mu]$ do
 5             $G \leftarrow G \cup \langle u, i, R[u][i]\rangle$
 6             $Candidate\_Set \leftarrow i \cup Candidate\_Set$
 7         $TotalCount \leftarrow TotalCount + Size(R[u])$
 8     for $i \in R[\mu]$ do
 9         for $\nu \in i.Neighbors$ & $\nu \notin R[\mu]$ do
               // Merge:put $\nu$ into $Item\_Map$ and sum up values of same $\nu$
10             $Merge(<\nu, simi_{\nu,i} * R[\mu][i]>, Item\_Map)$
11     $Num \leftarrow TotalCount/Size(\mu.Neighbors)$
12     for $\nu \leftarrow Sort(Item\_Map.Values, descending)[1 : Num]$ do
13         $G \leftarrow G \cup \langle \mu, \nu, Item\_Map.get(\nu)/\sum_{i \in R[\mu]} simi_{\nu,i}\rangle$
14         $Candidate\_Set \leftarrow \nu \cup Candidate\_Set$
15     for $Topic\ k = 1 : K$ do
16         for $i \in R[\mu] \cup Candidate\_Set$ do
17             $G \leftarrow G \cup \langle i, k, T[i][k]\rangle$
```

3.4 Random Walk-Based Recommendation

To generate the top-N recommended items in the proposed graph, we use personalized random walk with restart(RWR). By setting a biased probability of jumping to the starting nodes that are highly related to target user, RWR allows us to calculate the relatedness between candidate items and target user's preference. In our graph, neighbor nodes and history nodes are highly related to the target user, thus they are regarded as the starting nodes.

RWR works as follows: From the starting nodes, RWR is performed by randomly jumping to another linked node at each step, the jumping probability is proportional to the weight of outside links. Additionally, in each step there also exist probability α to force random walk restart at the starting nodes. Formally it is defined by Eq.(4).

$$r^{t+1} = (1 - \alpha)Mr^t + \alpha q \tag{4}$$

M is the transition matrix. $M_{i,j}$ denotes the probability of j being the next state given that the current state is i. q is the initial query vector in which the elements corresponding to the starting nodes are set to 1, others are set to zero. r^t records the visiting probability of each node at step t. Update the values of vector r iteratively until convergence at step l. Finally r_i^l represents the relatedness between node i and the starting nodes.

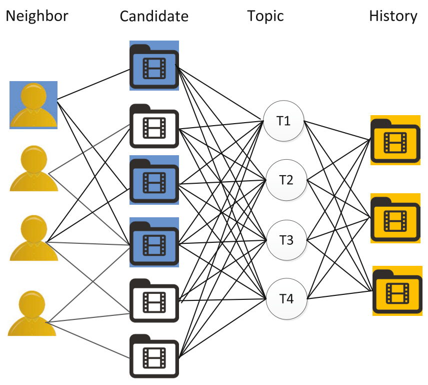


Fig. 3. Recommendation Graph

$$M=\begin{array}{c|cccc} & \text{Neighbor} & \text{Candidate} & \text{Topic} & \text{History} \\ \hline \text{Neighbor} & 0 & NC & 0 & 0 \\ \text{Candidate} & CN & 0 & CT & 0 \\ \text{Topic} & 0 & TC & 0 & TH \\ \text{History} & 0 & 0 & HT & 0 \end{array}$$

Fig. 4. Transition matrix of left graph. NC, CN, CT etc. are M's sub-matrices

In our setting, $q = (q_N, q_C, q_T, q_H)$, and the elements represent the four layers. Based on the starting nodes, we set $q = (\beta q_N, 0, 0, (1-\beta)q_H)$. β controls the tradeoff between collaboration and content. q_N^u denotes the similarity between neighbor u and target user. Considering that target user is also put into the neighbor layer, we set his similarity equal to 1. q_H^j denotes the importance of item j in the content factors. Here, we define $q_H^j = R_{u,j} * IDF(j)$ where $IDF(j) = Log(\frac{|I|}{ItemFreq(j)})$. $|I|$ denotes the number of total items. q_N and q_H are then normalized to 1 separately. $r^{(0)}$ is the initial visiting probability which is set equal to q. To get transition probability of each node, we need to normalize each row of M to 1, as illustrated in Fig. 4. In order to stop CN(ratings) from suppressing CT(topic distributions), we consider to give equal contributions from candidate layer to its two sides. Namely we firstly normalize CN and CT to 1 for each row respectively. Then CT and CN are normalized to 1 together for each row. After getting the stable visiting probability of each node, we select the top-N items from candidate layer by sorting their values in descending order.

3.5 Complexity Analysis

The main computation of random walk-based methods is updating the values of each node in the graph. Assuming that the number of average rated items per user is P, then there are N users, (N+1)P items, and K topics nodes in our recommendation graph. For updating user nodes, the computational complexity is O(NP). The computational complexities for candidate, topic and history nodes are O(NP(N+K)), O(K(NP+P)) and O(KP) respectively. Therefore, the total complexity in one iteration is O(P(N^2+2NK+2K+N)). Usually K and N are small fixed values, which indicates that the running time of our method is proportional to the number of average rated items per users. In a long period, average ratings of the whole users will not change so much, so the complexity of our proposed method possesses good stability.

4 Experiment

4.1 Experiment Settings

Datasets. For experimentation, we use two different datasets[1]: Lastfm and Movielen. In order to get enough tags to construct items' contents, we crawl the tags of the corresponding artists from last.fm[2] website, and the plot keywords of movies from imdb[3] website respectively. After that, we filter out items with less than 20 tags and items whose tags are not available. In fact the number of filtered items in this step is very small. The statistics of two datasets are listed in Table 1. Movielen is relatively dense whose average ratings per user is almost 400, while Lastfm is rather sparse. The density of user-item rating matrix denotes the percentage of non-zero values.

Compared Methods. To better understand our proposed method, We compared with five representative algorithms. (1)UCF: User-based collaborative filtering [15]; (2)ICF: Item-based collaborative filtering [15]; (3)UICF: Hybrid collaborative filtering, which combines the predicted score from both UCF and ICF to generate the final rankings. To consider the content-based factors, users similarities are calculated by using Eq.(1) described in section 3.2. $\delta = 0.5$; (4)PureSVD: An algorithm based on Matrix Factorization. [19] conducted extensive experiments to suggest that PureSVD outperforms all other powerful models such as AsySVD , SVD++ [20]; (5)RWR-FULL: Personalized random walk in graph which contains all the users and items [12]. They use target user and his already rated items as the starting nodes.

Further Study. To further explore the rationality of the way in which we fuse different factors. We compared a set of variants of our proposed method: RWR-U only considers the user-based factor; RWR-UI considers user-based and item-based factors; RWR-UC considers user-based and content-based factors. Their respective parameters are set in the same way as RWR-UIC.

Evaluation Metrics. To measure the performance, we adopt the following metrics to cover various aspects of our consideration. Additionally, we randomly split each user' ratings into five parts equally. 20% of the items in each user's profile are put aside as T_u for testing. As default, we focus on the performance of top 200 recommendations.

(a). **Accuracy**. We use precision, recall and MAP curves to measure accuracy. $Precision = \frac{|recs| \cap |tests|}{|recs|}$ and $Recall = \frac{|recs| \cap |tests|}{|tests|}$ have been used widely in recommendation [17]. To address the bias of rank positions in the list, we also utilize MAP. If correct answers are ranked higher, MAP is higher. MAP=$\frac{1}{|U|} \sum_{u \in U} \{ \frac{1}{|T_u|} \sum_{k=1}^{n} P(k) h(k) \}$. $h(k) = 1$ if the kth recommended item belongs to T_u, 0 otherwise. P(k) denotes the precision ranked up to k.

[1] http://grouplens.org/datasets/hetrec-2011/
[2] www.lastfm.com
[3] www.imdb.com

(b). **Coverage**. Obviously, accuracy is not enough for evaluation. Recommendations should cover distinct items stored in database as many as possible to boost presentation and sales. Coverage=$\frac{1}{|I|}|\bigcup_{u\in U}R(u)|$. R(u) denotes the recommended items for user u. $|I|$ is the total number of unique items.

(c). **Novelty**. It is trivial to recommend popular items which are so apparent to bring few surprise to users. Thus good recommendation lists should be better in Novelty=$\frac{1}{|U|}\sum_{u\in U}\{\frac{1}{|R(u)|}\sum_{i\in R(u)} log(item\_pop(i)+1)\}$. The lower, the better. Here, $item\_pop(i)$ denotes the rating popularity of item i.

Parameters Setting. The threshold of similar users or similar items in our proposed method, is set equal to UCF and ICF, so that we can compare the performance between basic CF and our method fairly. In experiments, we set them as 30 in both datasets for default. β controls the balance between collaboration and content, and we tested its sensitivity on a set of limited values $\beta \in \{0, 0.01, 0.05, 0.1, 0.5, 1\}$. In the random walk-based methods, α controls the probability of jumping to the starting nodes. We set α=0.8 as proposed in RWR-FULL [12], because we also find that when α increases, MAP, precision and recall all increase. This can be explained as a more personalized model as increasing α makes the model go back to the initial query vector q more frequently. The number of topics, K is set 30 in movielen and 40 in lastfm by cross validation. Iteration of all the random walk-based methods is set 100. The reduced dimension of PureSVD is set 50 as analyzed in [19] for getting better performance.

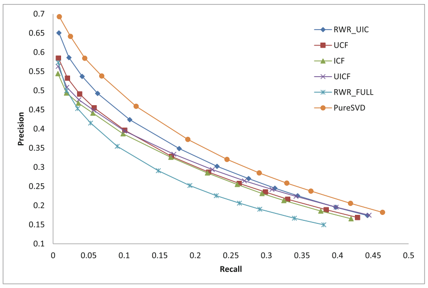
4.2 Results and Analysis

β controls the tradeoff between collaboration and content in query vector, so we firstly conducted a series of experiments to study its influence on final performance. If $\beta = 1$, we just regard neighbors as the starting nodes; while if $\beta = 0$, starting nodes are only history items. Because of a limitation of space, we only give the findings. We find that spreading out from only one side can not perform well. The optimal β is 0.01 in movielen and 0.05 in lastfm, which are very small. Because in the right graph part, nodes are fully connected by dense edges, while the left is rather sparse. As the transmitted values to other nodes need to be divided by outdegrees, to make the influence from history layer enough significant, we must give higher weight to q_H , which means a lower β for q_N.

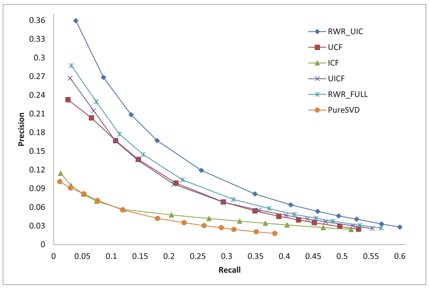
Then we compared our method with existing algorithms using two datasets. From Fig.5, we see that UCF Performs better than ICF especially in lastfm. Because the number of users is much smaller than that of items and we can get more information for each user than each item. However, UICF does not

Table 1. Datasets Statistics

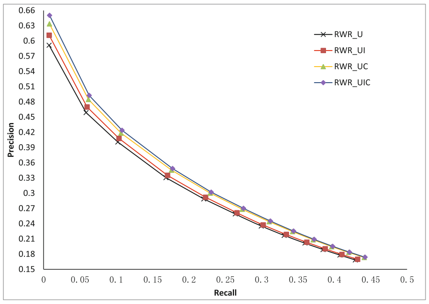
	# users	# items	# average rates	# average tags	# total rates	density
Movielen	2113	8631	394	101	834116	4.57%
Lastfm	1890	14998	47	56	90013	0.31%



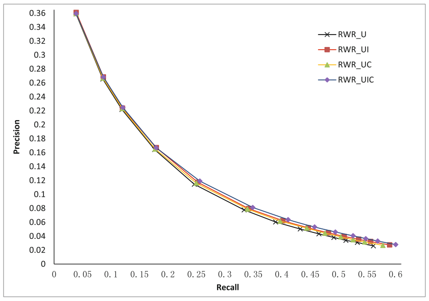
(a) movielen



(b) lastfm



(c) movielen



(d) lastfm

Fig. 5. Interpolated Precision-Recall curves for top-200 items. Fig(a),(b) compare our methods with other algorithms. Fig(c),(b) explore different variants of our approach.

gain much better performance than UCF, though it considers both content and collaboration. This phenomenon verifies our discussion that a simple weighted combination of several factors in the user similarity calculation and ranking adjustment can not perform effectively. But, our method RWR-UIC always outperforms other algorithms. The reason is that: On one hand, the random walk of our method matches people's decision-making process. Normally, people start with considering items: (1) similar to their history records (2) their similar-minded neighbors also like (3) contain the topics they prefer. The above three aspects are considered in our method at the same time in deciding an item's relatedness. And the interaction between different factors are not done once, but by iterative updates, so different factors are integrated fully. On the other hand, graph-based structure allows us to calculate the similarity between different items, even they have no direct connections. Namely, some extra latent relationships are explored.

Although both RWR-UIC and RWR-FULL use the graph-based ranking to generate top-N lists, RWR-UIC performs much better than RWR-FULL in both datasets. We argue that incorporating too much information brings some irrelevant noise which reduces the opportunity of transmitting to really related nodes, as RWR-FULL do. In contrast, We just retain the most relevant users and items in the graph and consider useful content factors apart from rating links. Another finding is that graph-based methods perform better in sparse dataset. In Lastfm

Table 2. Summary of MAP, Coverage and Novelty for all algorithms

	UCF	ICF	UICF	PureSVD	RWR-FULL	RWR-U	RWR-UI	RWR-UC	RWR-UIC
Movielen									
MAP	0.1822	0.1707	0.1849	**0.2265**	0.1501	0.1858	0.1903	0.1983	0.2010
Coverage	0.22	0.4831	0.2649	0.3599	0.0936	0.3007	0.5796	0.4518	**0.6001**
Novelty	6.2908	6.0313	6.2611	6.0130	6.4308	6.2511	6.1828	6.1187	**5.9950**
Lastfm									
MAP	0.1145	0.0598	0.1161	0.0517	0.1271	0.1464	0.1497	0.1487	**0.1517**
Coverage	0.4582	0.2709	0.3998	0.1937	0.4713	0.4983	0.6216	0.6504	**0.7148**
Novelty	3.3846	4.0411	3.6733	3.5219	3.8098	3.2975	3.2449	3.1428	**3.0934**

whose density is 0.39%, RWR-UIC and RWR-FULL both perform much better than all the other methods. While in the dense Movielen, RWR-FULL drops a lot in precision and recall. The advantage of our method RWR-UIC is not as significant as that in Lastfm. We explain it as follows: On one hand, graph-based methods can utilize indirect relationships to calculate similarity, which is rather useful in alleviating the sparsity of ratings; On the other hand, when density increases a lot from Lastfm to Movielen, Precision and Recall increase correspondingly. But introduced noise also weakens the growth rate.

PureSVD performs better than our method in the dense movie dataset, which suggests the good performance of utilizing latent rating patterns based on matrix factorization. However, in the sparse music dataset, PureSVD performs the worst of all. It is evident that PureSVD suffers from overfitting when the rating matrix is very sparse. In general, our method has stable performance in both datasets.

From Table 2, we see that our method performs better in novelty. Because we also consider items generated by item-based technique, as long as they are similar to user's history records. More importantly, content-based factor further boosts similar items, if they contain the topics target user prefers. If we just consider rating links, more popular items will be recommended as RWR-FULL. So we can recommend not so "apparent" items.

From Fig.5(c) 5(d) and Table.2. We find that considering more factors such as item-based or content-based techniques always improves the performance of RWR-U. But RWR-UIC performs the best. It is verified again that the intuition of our random walk reflects people's decision-making process. People can judge an item's relatedness more accurately based on all the three aspects. Meanwhile the graph structure is suitable and effective to fuse different factors.

As random walk is conducted in a concise sub-graph which only contains carefully selected relevant information. Our method has a fast speed of convergence. We tested our method(written by JAVA) in a server with 32 GB RAM. In movielen, our running time is 220 ms, while RWR-FULL needs 1.94s; In lastfm, we just need 40 ms. RWR-FULL needs 174ms. The complexity of our method is greatly reduced compared to existing graph-based methods. PureSVD can be run fast online, but SVD decomposition is rather time-consuming offline.

5 Conclusion

In this paper, we propose a graph-based method for hybrid recommendation, which can combine different factors iteratively. Our method can alleviate the data sparsity and boost long tail items. Experimental results show that our method performs well especially in sparse dataset. In the future, we plan to explore more relationship in the same layer to further improve the performance.

Acknowledgement. This research is supported by the 863 project of China (2013AA013300), National Natural Science Foundation of China (Grant No. 61375054 and 61402045) and Tsinghua University Initiative Scientific Research Program (20131089256), and Cross fund of Graduate School at Shenzhen, Tsinghua University (JC20140001).

References

1. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence 2009, 4 (2009)
2. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: SIGIR 1999, pp. 230–237 (1999)
3. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW 2001, pp. 285–295 (2001)
4. Hofmann, T.: Probabilistic latent semantic indexing. In: SIGIR 1999, vol. 24, pp. 50–57 (1999)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
6. Lops, P., De Gemmis, M., Semeraro, G.: Content-based recommender systems: State of the art and trends. In: Recommender Systems Handbook, pp. 73–105 (2011)
7. Meng, F., Gao, D., Li, W., Sun, X., Hou, Y.: A unified graph model for personalized query-oriented reference paper recommendation. In: CIKM 2013, pp. 1509–1512 (2013)
8. Popescul, A., et al.: Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In: UAI 2001, pp. 437–444 (2001)
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
10. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Communications of the ACM 40(3), 66–72 (1997)
11. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW 2007, pp. 271–280 (2007)
12. Konstas, I., Stathopoulos, V., Jose, J.M.: On social networks and collaborative recommendation. In: SIGIR 2009, pp. 195–202 (2009)
13. Haveliwala, T.H.: Topic-sensitive pagerank. In: WWW 2002, pp. 517–526 (2002)
14. Craswell, N., Szummer, M.: Random walks on the click graph. In: SIGIR 2007, pp. 239–246 (2007)
15. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: CIKM 2001, pp. 247–254 (2001)

16. Yildirim, H., Krishnamoorthy, M.S.: A random walk method for alleviating the sparsity problem in collaborative filtering. In: RecSys 2008, pp. 131–138 (2008)
17. Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Recommender Systems Handbook, pp. 257–297 (2011)
18. Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., Welling, M.: Fast collapsed gibbs sampling for latent dirichlet allocation. In: KDD 2008, pp. 569–577 (2008)
19. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: RecSys 2010, pp. 39–46 (2010)
20. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD 2008, pp. 426–434 (2008)