Scalable Collaborative Filtering Using Incremental Update and Local Link Prediction

Xiao Yang
School of Computer Science and Technology, Harbin Institute of Technology
Harbin, China
shawcsn@gmail.com

Zhaoxin Zhang
School of Computer Science and Technology, Harbin Institute of Technology
Harbin, China
heart@hit.edu.cn

Ke Wang
Department of Mathematics, College of Science, Harbin Institute of Technology
Harbin, China
w_k@hit.edu.cn

ABSTRACT
The traditional collaborative filtering approaches have been shown to suffer from two fundamental problems: data sparsity and difficulty in scalability. To address these problems, we present a novel scalable item-based collaborative filtering method by using incremental update and local link prediction. By subdividing the computations and analyzing the factors in different cases of item-to-item similarity, we design the incremental update strategies in item-based CF, which can make the recommender system more efficient and scalable. Based on the transitive structure of item similarity graph, we use the local link prediction method to find implicit candidates to alleviate the lack of neighbors in predictions and recommendations caused by the sparsity of data. The experiment results validate that our algorithm can improve the performance of traditional CF, and can increase the efficiency in recommendations.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information filtering.

General Terms
Algorithms, Performance, Design, Experimentation.

Keywords
Collaborative Filtering, Incremental Update, Similarity Graph, Link Prediction, Scalability

1. INTRODUCTION
The ongoing rapid expansion of the Internet greatly increases the necessity of effective recommender systems for filtering the abundant information. Recommender system can automatically provide personalized recommendations based on the historical record of users' activities. As one of the most successful approaches to building recommender systems, collaborative filtering (CF) uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users. There are many different collaborative filtering methods, which can be classified into User-based CF, Item-based CF and model-based CF [1,2] in general. Although model-based approaches are known to provide accurate recommendation performances, a number of models have difficulty explaining the predictions. Some other hybrid CF algorithms[2] (that combine CF with other recommendation techniques) has been proposed to improve the performance of recommendation results, but the tunable parameters in most hybrid methods are difficult to determine under a universal form in different scenarios for giving a reasonable recommendation list.

However, both the above collaborative filtering approaches suffer from two fundamental problems in practice: data sparsity [3] and inability to scale up [4]. Data sparsity refers to the difficulty that most users rate only a small number of items and hence a very sparse user-item matrix is available. As for computational scalability, algorithms based on standard CF approaches often cannot cope well with the large numbers of users and items. Moreover, another drawback in current CF approaches is that they are more appropriate for static settings and off-line compute, incorporating new data to the system may be a non-trivial task. In real world problems, there are always new users, items and ratings that should be incorporated into the recommendations in an online manner. So we need to design the incremental collaborative algorithms to give real-time responses to the change of data in an effective and efficient way.

Recently, many researchers have proposed some significant methods to improve the applying of collaborative filtering algorithms in real online systems. Takács et al.[3] proposed various scalable solutions that are validated against the Netflix Prize data set. Koren [4]introduced a new neighborhood model with an improved accuracy on par with recent latent factor models, which is more scalable than previous methods without compromising its accuracy or other desired properties. Khoshneshin et al.[5] proposed an evolutionary co-clustering method that improves predictive performance while maintaining the scalability of co-clustering in the online phase. Papagelis et al.[6] proposed a method for addressing the scalability problem based on incremental updates of user-to-user similarities for online application. Davidson et al.[7] discussed the video recommendation system in use at YouTube.

In this paper, by analyzing the components of traditional collaborative filtering methods, we investigate the specific computational complexity and some weaknesses in the selection of neighbors. Based on principal compute factor analysis and graph-based selection of similar neighbor items, we propose a novel scalable item-based collaborative filtering method by using incremental update and local link prediction. Traditional CF recommender systems need to adopt the traditionally static algorithmic framework, which periodically recomputed the

CIKM'12, October 29-November 2, 2012, Maui, HI, USA.

similarity on the whole data. To address this scalability problem, we present an incremental CF method, based on incremental updates of item-to-item similarities that is able to recommend items orders of magnitude faster than classic CF, while maintaining the recommendation quality. Meanwhile, we utilize the similarity graph of items to generate recommendation results rapidly. Due to the sparsity of datasets, the process of rating prediction and item recommendation may lack similar neighbors to get reasonable recommendations. We try to alleviate this problem by using local link prediction in item similarity graph to find implicit neighbor candidates. Empirical studies on two datasets (MovieLens and Netflix) show that our new proposed approach can improve the performance of CF while maintaining the scalability of recommendation.

2. METHODOLOGY

Essentially, a recommender system consists of users and item, and each user has selected or rated some items. The object in the recommender system can be denoted as a triple from consisting of users, items and ratings: $S = \{U, I, R\}$. We denote the set of users as $U = \{u_1, u_2, ..., u_m\}$, the set of items as $I = \{i_1, i_2, ..., i_n\}$ and the set of ratings as $R = \{r_1, r_2, ..., r_k\}$. The main purpose of recommender system is predicting your personal preference to those objects you have not yet selected, based on the historical record of users' activities. And the output of the system is recommendation list sorted in rank of predicting scores. Since real users are usually concerned only with the top part of the recommendation list, a more practical approach is to consider the number of a user's relevant objects ranked in the top-N places.

2.1 Item-based Collaborative Filtering

We know that the choice between a user-based and an item-based approach partly depends on the frequency and amount of change in the users and items of the system. If the list of available items is fairly static in comparison to the users of the system, an item-based method may be preferable since the item similarity weights could then be computed at infrequent time intervals while still being able to recommend items to new users. As the increase speed of users is much faster than items in most online media systems such as *Amazon*, *YouTube*, *Last.fm* and so on, the item-based [7] recommendation algorithm become a more practical technology in commercial or online applications. We can compute and cache similarities that converge of items, which can give item based recommenders a performance advantage. On the user experience side, item-based recommendation is more explainable and succinct. And Deshpande et al.[9] validated that item-based algorithms are up to two orders of magnitude faster than the traditional user-neighborhood based recommender systems and provide recommendations with comparable or better quality. So the item-based CF is more suitable for effective and scalable recommender system.

Item-based CF first analyzes the user-item matrix to identify relationships between different items, and then uses these relationships to indirectly compute recommendations for users. It contains two main components: item similarity computation and prediction computation. The similarity between two items is dependent upon the ratings given to the items by users who have rated both of them. There are a number of different mathematical formulations that can be used to calculate the similarity between two items. Here we use the adjusted cosine similarity [10] as the measure of item-to-item similarity, which take into the fact that different users have different ratings schemes. Formally, the similarity between items *i* and *j* using this scheme is given by:

$$sim(i,j) = \frac{\sum_{u \in U} (r_{ui} - \overline{r}_u)(r_{uj} - \overline{r}_u)}{\sqrt{\sum_{u \in U} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_{u \in U} (r_{uj} - \overline{r}_u)^2}} \quad (1)$$

the set of users who both rated *i* and *j* is denoted by *U*, r_{ui} denotes the rating of user *u* on item *i*, and $\overline{r}_u$ is the average of the *u*-th user's ratings. The prediction of the target item for the user is given by the sum of the average rating of the target item and the weighted average of its neighbors [10]:

$$p_{ui} = \overline{r}_i + \frac{\sum_{j \in N(i)} sim(i,j)(r_{u,j} - \overline{r}_j)}{\sum_{j \in N(i)} |sim(i,j)|} \quad (2)$$

where $\overline{r}_i$ and $\overline{r}_j$ denote the average rating of the item *i* and *j* over all users, and the $N(i)$ denotes the set of selected neighbors of target item.

2.2 Incremental Similarity Update

A scalable recommender system should handle the constantly changing data source in online dynamic environment, where new users keep appearing, new items are introduced and preferences changing. In the process of collaborative filtering, the main update of computation is the similarity of user-to-user or item-to-item caused by the updating of ratings. Some information recommender systems still adopt the traditionally static periodical update of similarity and recommendation in an off-line way. So the incremental update strategies should be considered in the design and implementation of CF algorithms. An incremental user-based CF method for continuous explicit ratings has been proposed by Papagelis et al. [7]. This method is based on incremental updates of user-to-user similarities. What we propose here is an item-based incremental algorithm for updating of item-to-item similarities.

Inspired by the incremental update mechanism of user-to-user similarities in [7], we present a method to deal with the incremental update in item-based CF based on rigorous mathematics derivation. When a user *u* submits a new rating or updates the value of an already submitted rating, the item-to-item similarity value may need to be recomputed. We try to express the new similarity value between two items in relation to their old similarity value. This describes an incremental update of their associated similarity. We adopt the following notation for the adjusted cosine similarity measure of equation 1:

$$A = sim(i,j), B = \sum_{u \in U} (r_{ui} - \overline{r}_u)(r_{uj} - \overline{r}_u) \quad (3)$$

$$C = \sqrt{\sum_{u \in U} (r_{ui} - \overline{r}_u)^2}, D = \sqrt{\sum_{u \in U} (r_{uj} - \overline{r}_u)^2} \quad (4)$$

$$A = \frac{B}{\sqrt{C}\sqrt{D}} \quad (5)$$

In this way, we split the similarity measure into three factors *B*, *C*, *D*, independently calculate the new values of each factor *B*', *C*', *D*' and then combine these values so as to obtain the value of the new similarity *A*' as shown below:

$$A' = \frac{B'}{\sqrt{C'}\sqrt{D'}} \Rightarrow A' = \frac{B+x}{\sqrt{C+y}\sqrt{D+z}} \quad (6)$$

where *x*, *y*, *z* are increments that need to be computed after either the submission of a new or the update of an existing rating. We consider the different computations needed for the two special cases of response to the actions of user *u*:

(1) Submission of a new rating to item *i*: in this case, if the user *u* have not rated *j*, we can find that the increments are not needed to be changed. And if the user *u* has rated *j*, the increments to be updated can be computed as equations 7-9[1].

$$x = \left(r_{uj} - \frac{r_{ui}' + \overline{r}_u \cdot |r_u|}{|r_u|+1} \right) \cdot \frac{r_{ui}' + \overline{r}_u \cdot |r_u|}{|r_u|+1} \quad (7)$$

$$y = \left(\frac{(r_{ui}' - \overline{r}_u) \cdot |r_u|}{|r_u|+1} \right)^2 \quad (8)$$

$$z = \left(r_{uj} - \frac{r_{ui}' + \overline{r}_u \cdot |r_u|}{|r_u|+1} \right)^2 \quad (9)$$

(2) Update of an existing rating to item *i*: in this case, if the user *u* have not rated *j*, we can find that the increments are not needed to be changed. And if the user *u* has rated *j*, the increments to be updated can be computed as equations 10-12[1].

$$x = r_{ui}' \cdot r_{uj} - \left(r_{ui} - \overline{r}_u\right)\left(r_{uj} - \overline{r}_u\right) - \left(r_{ui}' + r_{uj} - \frac{r_{ui}' - r_{ui}}{|r_u|} \right) \cdot \left(\frac{r_{ui}' - r_{ui}}{|r_u|} + \overline{r}_u \right) \quad (10)$$

$$y = \left(r_{ui}' - \frac{r_{ui}' - r_{ui}}{|r_u|} - \overline{r}_u \right)^2 - \left(r_{ui} - \overline{r}_u\right)^2 \quad (11)$$

$$z = \left(r_{uj} - \frac{r_{ui}' - r_{ui}}{|r_u|} - \overline{r}_u \right)^2 - \left(r_{uj} - \overline{r}_u\right)^2 \quad (12)$$

We use the values of *B*, *C* and *D* and the respective increments *x*, *y*, *z* to update *B*', *C*' and *D*'. To support the incremental computation of the new similarities we need to define a caching scheme in a database, which stores the values of *B*, *C* and *D* for items. Furthermore, we cache the average rating and the number of items that each user has rated. Cached information needs to be updated after the submission of a new or the update of an existing rating.

[1] The proof of equations is omitted here due to the limit of pages.

2.3 Similarity Graph and Link Prediction

After the item similarity computation of item-based CF, we can use the similar neighbors of target item to prediction computation or generating recommendation lists. To get the prediction or recommendation more efficient and stable, the neighborhood-based methods are frequently used in collaborative filtering. As shown in equation 2, we can predict the target item for the user by computing the weighted sum of item's neighbors. We often choose the k most similar items to constitute the set of neighbors, and the number of neighbors can influence the accuracy and efficiency of recommender systems. To eliminate the time consumption of the Nearest Neighbor algorithm for finding the neighbors of target item, we use the graph-based storage model where the items are represented as the nodes in a graph and the edges between the nodes indicate the degree of similarity between the items. The recommendations and prediction are computed by traversing nearby nodes in this graph. The graph representation of the model allows it to capture transitive relations of items which cannot be captured by nearest neighbor algorithms. High-performance graph database such as the *Neo4j* (http://neo4j.org/) can support fast traversal of weighted graph. We cache and update the top-80 nearest neighbors of items in another memory-based database to give real-time responses to the need of recommendation.

The sparsity problem occurs when available data are insufficient for identifying similar items (neighbors) and it is a major issue that limits the quality of recommendations and the applicability of CF in general. Base on the item similarity network, we propose an effective approach that provides reasonable prediction and recommendations even when sufficient data are unavailable. When making predictions or generating recommendations, the sparse datasets may result in lack of enough neighbors of target item, which lead to inaccurate and unreasonable results. By utilizing the unique network structure of item similarity graph, we use the link prediction to supplement some implicit neighbor candidates. Link prediction [8] aims at estimating the likelihood of the existence of a link between two nodes, based on the observed links and the attributes of the nodes. There are many different algorithms to make link prediction in graph, and the linkage proximity metric is an efficient way to find possible nodes to link. We use the Adamic-Adar Coefficient [8] extended for weighted networks as the proximity measure to find the implicit neighbors for a target item in similarity graph, and this measure is given by:

$$S(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{w(x,z) + w(y,z)}{\log(1 + \sum_{c \in \Gamma(z)} w(z,c))} \quad (13)$$

where the $\Gamma(x)$ denotes the set of neighbors of node *x* in the network, and the $w(x,y)$ is the link weight between nodes *x* and *y*. In this way, we can replenish some useful items to alleviate the lack of neighbors caused by the high sparsity in data.

3. EXPERIMENTS

3.1 Datasets and Evaluation

To test the performance of our recommendation algorithms, we choose two classical datasets for the experiment. The first dataset is the MovieLens which is obtained from the shared datasets of GroupLens Research (http://www.grouplens.org/node/12). The

second dataset is a randomly-selected subset of the huge data set of the Netflix Prize (http://www.netflixprize.com/). The detailed properties of each dataset are shown in Table 1. The sparsity of the datasets is defined as one minus the quotient of ratings divided by the total number of user-object pairs.

Table 1. Properties of datasets

Dataset	Users	Items	Ratings	Sparsity
MovieLens	5883	3618	961216	0.9548
Netflix	11726	7491	1976562	0.9775

To evaluate the performance of our scalable item-based collaborative filtering (SICF), we use the Mean Absolute Error (MAE) [1] and the Root Mean Squared Error (RMSE) [1] as the measures of accuracy. And the Mean Response Time (MRT) of generating a recommendation for randomly-selected 100 users is chosen as the evaluation metric of efficiency. Experiments have been carried out on a 3.0 GHz, 8G RAM server.

3.2 Experimental Result

In this paper, we choose the classical user-based KNN CF and item-based KNN CF as our baseline in the comparison of experiment. We divided our datasets (by random sampling)into 85% training-15% testing sets and report the average results of a 5-fold cross-validation. The experimental results of datasets are shown in Table 2.

Table 2. Algorithmic performance on each dataset.

Dataset	Method	MAE	RMSE	MRT(s)
MovieLens	ItemKNN	0.7018	0.8873	2.198
	UserKNN	0.7147	0.902	3.776
	SICF	**0.6974**	**0.8671**	**0.119**
Netflix	ItemKNN	0.7449	0.9573	4.426
	UserKNN	0.7486	0.9621	7.154
	SICF	**0.7235**	**0.9265**	**0.234**

From the experimental result we can find that our scalable item-based collaborative filtering can provide higher accuracy based on the supplement methods for alleviating the sparsity of datasets, and can improve the scalability and efficiency of recommendations. The SICF is a more practical method for online dynamic environment.

4. CONCLUSION AND FUTURE WORK

High dimensionality and sparsity seem to be the main limit of quality for most of the CF-based recommendation systems. For dealing with these tough problems, we proposed a scalable item-based collaborative filtering method. We propose the incremental update strategies in item-based CF to make the recommender system more scalable and adaptive to the constantly changing of data. Based on the graph-based representation of similarity of items, we can solve the insufficient of neighbors caused by the sparsity in data by using the local link prediction. Experimental results show that our proposed method can significantly improve the accuracy of predication as well as solve the scalability problem. For future work, we will investigate the other more efficient incremental update methods which can support distributed computation, and try to develop more precise algorithms of link prediction in the item similarity graph.

5. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No.61100189, 61003261).

6. REFERENCES

[1] Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T. 2004. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS). 22, 1 (2004), 5–53.

[2] Adomavicius, G. and Tuzhilin, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on. 17, 6 (2005), 734–749.

[3] Takács, G., Pil'aszy, I., Németh, B. and Tikk, D. 2009. Scalable collaborative filtering approaches for large recommender systems. The Journal of Machine Learning Research. 10, (2009), 623–656.

[4] Koren, Y. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transactions on Knowledge Discovery from Data (TKDD). 4, 1 (2010), 1.

[5] Khoshneshin, M. and Street, W.N. 2010. Incremental collaborative filtering via evolutionary co-clustering. Proceedings of the fourth ACM conference on Recommender systems (2010), 325–328.

[6] Papagelis, M., Rousidis, I., Plexousakis, D. and Theoharopoulos, E. 2005. Incremental collaborative filtering for highly-scalable recommendation algorithms. Foundations of Intelligent Systems. (2005), 7–17.

[7] Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. and others 2010. The YouTube video recommendation system. Proceedings of the fourth ACM conference on Recommender systems (2010), 293–296.

[8] Liben-Nowell, D. and Kleinberg, J. 2007. The link-prediction problem for social networks. Journal of the American society for information science and technology. 58, 7 (2007), 1019–1031.

[9] Deshpande, M. and Karypis, G. 2004. Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS). 22, 1 (2004), 143–177.

[10] Park, S.T. and Pennock, D.M. 2007. Applying collaborative filtering techniques to movie search for better ranking and browsing. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (2007), 550–559.