Learning to Rank for Hybrid Recommendation

Jiankai Sun[1], Shuaiqiang Wang[2], Byron J. Gao[3], Jun Ma[1]
[1]Shandong University, Jinan, 250101 China
[2]Shandong University of Finance and Economics, Jinan, 250014 China
[3]Texas State University-San Marcos, San Marcos, TX 78666, USA
sjk2412@gmail.com, swang@sdufe.edu.cn,
bgao@txstate.edu, majun@sdu.edu.cn

ABSTRACT

Most existing recommender systems can be classified into two categories: collaborative filtering and content-based filtering. *Hybrid recommender systems* combine the advantages of the two for improved recommendation performance. Traditional recommender systems are rating-based. However, predicting ratings is an intermediate step towards their ultimate goal of generating rankings or recommendation lists. *Learning to rank* is an established means of predicting rankings and has recently demonstrated high promise in improving quality of recommendations. In this paper, we propose LRHR, the first attempt that adapts learning to rank to hybrid recommender systems. LRHR first defines novel representations for both users and items so that they can be content-comparable. Then, LRHR identifies a set of novel meta-level features for learning purposes. Finally, LRHR adopts RankSVM, a pairwise learning to rank algorithm, to generate recommendation lists of items for users. Extensive experiments on benchmarks in comparison with the state-of-the-art algorithms demonstrate the performance gain of our approach.

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

General Terms: Algorithms, Performance, Experimentation.

Keywords: Recommender systems, Collaborative filtering, Learning to rank, Features

1. INTRODUCTION

Most existing recommender systems can be classified into two categories: *collaborative filtering* (CF) and *content-based filtering*. CF is based on the assumption that if users X and Y rate n items similarly or have similar behaviors, they will rate or act on other items similarly [11]. CF only uses the user-item rating matrix to make predictions and recommendations. Content-based filtering makes recommendations by finding regularities in the textual content information of users and items, such as user profiles and product descriptions, where users and items are represented by a set of explicit features. To combine the advantages of the two and improve recommendation performance, various hybrid recommender systems [7] have been proposed in recent years, which have been found outperforming CF and content-based methods [13].

Learning to rank for recommendation. Traditional recommendation systems are rating-based, where the recommendation problem is reduced to the task of rating prediction. However, the ultimate goal of recommendation systems is to generate rankings or recommendation lists. Rating prediction is just an intermediate step towards ranking prediction and considered as an inferior objective [2].

EXAMPLE 1. *The objectives of rating and ranking prediction are not necessarily consistent with each other. Suppose the ground truth ratings are 5 and 4 for items A and B. While "rating(A) = 4, rating(B)=5" is a better result than "rating(A)=2, rating(B)=1" w.r.t. rating prediction, it is worse w.r.t. ranking prediction.*

Recent efforts on ranking-based recommender systems include [5] and [10] that particularly made use of learning to rank techniques. *Learning to rank* [6] utilizes supervised or semi-supervised machine learning techniques to automatically construct a ranking model based on a given training data. It has been extensively and effectively applied to rank documents in information retrieval and Web search [4].

Same as CF, existing works on learning to rank for recommendation [10, 12] only use the user-item rating matrix, where they adopt probabilistic matrix factorization [9] to represent users and items with *latent features*. These methods have demonstrated comparative advantages over their traditional CF counterparts.

Learning to rank for hybrid recommendation. Existing hybrid recommender systems are not ranking-based. Existing learning to rank for recommendation methods do not use valuable content information. In this study, we seek combined advantages of the two and propose LRHR, the first attempt that uses learning to rank for hybrid recommendation.

Nonetheless, it is not straightforward to adapt learning to rank to hybrid recommender systems. Unlike queries and documents in information retrieval, users and items are not directly *content-comparable*. In information retrieval, content similarity for query-document pairs indicates their relevance and can be used to rank documents. However, it does not make sense to compute the content similarity between a user profile and an item description because they are semantically unrelated.

To adapt learning to rank to hybrid recommender systems, LRHR first defines novel representations for both users and items so that they can be content-comparable. Then, LRHR identifies a set of novel meta-level features for learning purposes. Finally, LRHR adopts RankSVM, a pairwise learning to rank algorithm, to generate recommendation lists of items for users. Extensive experi-

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

ments on benchmarks in comparison with the state-of-the-art algorithms show that LRHR gains in rank-based performance measure of $NDCG@n$.

2. PROBLEM STATEMENT

Let U be a collection of user profiles, each depicting user's attributes such as age and gender. Let I be a collection of item descriptions, each depicting item's attributes such as title and release date. Each user $\boldsymbol{u} \in U$ rates a set of items from I. Let $R_{|U|,|I|}$ be the rating matrix, where each element $r_{p,q}$ is a natural number indicating the rating score of item $\boldsymbol{i}_q$ from user $\boldsymbol{u}_p$.

Let f be a ranking function. For a set of items $I_{\boldsymbol{u}}$ from a single user $\boldsymbol{u}$, $f(I_{\boldsymbol{u}})$ outputs a ranking of $I_{\boldsymbol{u}}$. Let f^* be the optimal ranking function. A given loss function $s(f, f^*)$ can be used to estimate the goodness of f based on the distance between f and f^* with respect to U.

Ranking-based hybrid recommendation problem. Given a set of users U, a set of items I, a rating matrix R with respect to U and I, and a loss function s, the problem of ranking-based hybrid recommendation is to generate a ranking function f from U, I and R such that $s(f, f^*)$ is minimized, where f^* is the optimal ranking function.

Technical challenges. Ranking items for users are not as straightforward as ranking documents for queries. In the document ranking problem, queries are treated as documents. They can be represented using terms (i.e., pre-processed words) from the vocabulary of documents. Then, a set of meta-level features [1], such as term frequency (TF) and inverse document frequency (IDF), can be defined and used to measure the similarity or relevance of each query-document pair. Documents can be ranked according to their relevance to the query.

In the hybrid recommendation problem, user profiles can be considered as queries and item descriptions can be considered as documents. However, user profiles and item descriptions are not content-comparable. The vocabulary for user profiles and the vocabulary for item descriptions do not overlap and any content-based similarity between users and items would be zero and useless. Thus it is difficult to define meta-level features and adapt learning to rank to hybrid recommendation.

3. FEATURES IN LRHR

In this section, we represent user profiles and item descriptions to make them content-comparable. Then, we define two categories of features: meta-level features and rating-based features.

3.1 Representations for Users and Items

LRHR combines the vocabularies for user profiles and item descriptions to form a single vocabulary T. Then it adopts the *bag of words* model to represent users and items as vectors. In particular, each user $\boldsymbol{u}$ is represented by a bag of user terms and a bag of item terms. The former contains the content information (profile) of $\boldsymbol{u}$ and the latter contains the content information (descriptions) of all items rated by $\boldsymbol{u}$. Each item $\boldsymbol{i}$ is also represented by a bag of item terms and a bag of user terms. The former contents the content information (description) of $\boldsymbol{i}$ and the latter contains the content information (profiles) of all users who have rated $\boldsymbol{i}$. In each representation, the frequency for each term will also be recorded.

In forming T, all the continuous values are first discretized. For example, the age attribute of users can take values of $(0, 18]$, $(18, 24]$, $(25, 34]$, etc.

EXAMPLE 2. *Let $\boldsymbol{u}$ be a user who is 24, male, and a technician. Let $I_{\boldsymbol{u}}$ be a set of movies that are rated by $\boldsymbol{u}$, containing 6 action movies, 4 adventure movies and 1 western movie. Then $\boldsymbol{u}$ can be represented by 6 terms: (18, 24](1), male(1), technician(1), action(6), adventure(4), and western(1).*

EXAMPLE 3. *Let $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ be two users who have rated movie $\boldsymbol{i}$. Let $\boldsymbol{u}_1$ be a 24 year-old male technician, and $\boldsymbol{u}_2$ be a 19 year-old male student. Let $\boldsymbol{i}$ be an action and western movie. Then $\boldsymbol{i}$ can be represented by 6 terms: (18, 24](2), male(2), technician(1), student(1), action(1), and western(1).*

3.2 Meta-level Features

Now we define a set of explicit meta-level features, inspired by term frequency TF, inverse document frequency IDF, and their combination TF-IDF.

TF-IDF is widely used for document ranking in information retrieval. The term frequency $TF_{t,\boldsymbol{d}}$ of term t in document $\boldsymbol{d}$ is generally defined as the number of times that t occurs in $\boldsymbol{d}$. It indicates the relevance of $\boldsymbol{d}$ to t.

The inverse document frequency IDF_t of term t measures the general importance of t. It is obtained by dividing N by DF_t and then taking the logarithm of that quotient, where N is the total number of documents and DF_t is the document frequency of t, i.e., the number of documents containing t. Formally,

$$IDF_t = \log_{10}\left(\frac{N}{DF_t}\right)$$

The TF-IDF value of a term is commonly defined as the product of its TF and IDF values.

$$TF\text{-}IDF_{t,\boldsymbol{d}} = TF_{t,\boldsymbol{d}} \times IDF_t$$

Inspired by TF, IDF and TF-IDF that are used in learning to rank for information retrieval, we define a set of explicit meta-level features for LRHR. There are two categories of terms: user terms and item terms. Thus there are also two categories of meta-level features: user-oriented features and item-oriented features.

Meta-level features for recommendation. Although conceptually we treat users as queries, items as documents, and transform the ranking-based recommendation problem into a document ranking problem in information retrieval, there are significant differences between the two.

(1) In information retrieval, each document uses the same weighting scheme for all terms in the document. In our case, each representation has two categories of terms with different frequencies. In particular, as shown in Example 2 and Example 3, user terms in user representations and item terms in item representations have low frequencies (generally 1), which we call *low-frequency terms*. User terms in item representations and item terms in user representations have very high frequencies, which we call *high-frequency terms*. Thus we need to use two weighting schemes.

In LRHR, we use the original TF for low-frequency terms, and use WF, a logarithm variant of TF, for high-frequency terms. Specifically,

$$WF_{t,\boldsymbol{d}} = \begin{cases} 1 + \log_n TF_{t,\boldsymbol{d}}, & \text{if} \quad TF_{t,\boldsymbol{d}} > 0 \\ 0, & \text{otherwise} \end{cases}$$

(2) While the low-frequency terms in user and item representations usually have similar frequencies (generally 1), the high-frequency terms in the two kinds of representations may vary significantly. For example, a user may not have rated many movies, but a movie may have been rated by much more users (This is not reflected in Example 3. To construct a more realistic example, we would have to add many users for movie i).

In general, the actual frequencies for the two kinds of high-frequency terms depends on the application. In LRHR, for the WF features, we use a larger logarithm base (n) for high-frequency terms with relatively higher frequencies, and a smaller logarithm base for high-frequency terms with relatively lower frequencies.

(3) In information retrieval, queries are always shorter than documents. For efficiency, the common practice for weighting queries and documents is to compute IDF for query terms and not to compute IDF for documents. In our case, we have two categories of terms: user terms and item terms. In LRHR, we compute IDF for user terms in user representations and item terms in item representations.

EXAMPLE 4. *Let **u** be a user represented by 6 terms: (18, 24](1), male(1), technician(1), action(64), adventure(16), and western(8). Let **i** be a movie represented by 6 terms: (18, 24](10,000), male(100,000), technician(1,000), student(1,000), action(1), and western(1). For the first 3 user terms in **u** and the last 2 item terms in **i**, their TF values are 1. In this application, obviously the frequencies for high-frequency terms (user terms) in item representations are much higher than the high-frequency terms (item terms) in user representations. Thus we use a larger logarithm base (10) for the high-frequency terms in **i** and a smaller logarithm base (2) for the ones in **u** in computing WF. Specifically, the TF values for the last 3 item terms in **u** are 7, 5 and 4 respectively, and the TF values for the first 4 user terms in **i** are 5, 6, 4, 4 respectively.*

*Suppose the IDF values for the user terms (18, 24], male, technician and student are 4, 2, 3 and 3, and the IDF values for the item terms action, adventure, and western are 2, 2 and 2. Then the TF-IDF values for the terms in **u** are 4, 2, 3, 14, 10 and 8. The TF-IDF values for the terms in **i** are 20, 12, 12, 12, 2, 2.*

3.3 Rating-based Features

For each user term t in a representation for item ***i***, we define a rating-based feature, the average rating score $avgScore_i^t$. Let U_i be the set of users who have rated the item ***i***. Let $U_i^t = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_l\} \subseteq U_i$ be the set of users who have rated ***i*** and whose representations contain the term t. Let $r_{k,\boldsymbol{i}}$ be the rating score assigned for ***i*** by user $\boldsymbol{u}_k \in U_i^t$. Then,

$$avgScore_i^t = \frac{\sum_{k=1}^{l} r_{k,\boldsymbol{i}}}{l}.$$

For each item ***i***, we also define a rating-based feature, the average score of the item $avgScore_i$, which takes the average of $avgScore_i^t$ values for all user terms in a representation for ***i***.

Similarly, we also define 2 rating-based features for each user profile ***u***: the average score $avgScore_u^t$ for the item term t assigned by ***u***, and the average score $avgScore_u$ which takes the average of $avgScore_u^t$ values for all item terms in a representation for ***u***.

In addition, for each item ***i***, we also use N_i, the number of users who have rated ***i*** as a feature.

4. THE LRHR ALGORITHM

LRHR treats each user as a query, each item rated by the user as a document associated with the query, and the corresponding rating score as the relevance label.

LRHR formulates the generic ranking-based hybrid recommendation problem as a pairwise learning to rank problem, where item (documents) pairs are constructed based on their relevance scores. For the actual learning to rank algorithm that returns the ranking function f, LRHR uses RankSVM [4], a classic pairwise learning to rank algoirthm. RankSVM takes document pairs as instances in learning, and formulates learning to rank as classification.

In more details, let $\mathbf{x}$ be a document. Let $f_{\mathbf{w}} = \mathbf{w} \cdot \mathbf{x}$ be a linear function where $\mathbf{w}$ denotes a vector of weights and $\cdot$ denotes inner product. Let $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_i^{(2)}$ be two documents associated with query q_i. The preference relationship $\mathbf{x}_i^{(1)} \succ \mathbf{x}_i^{(2)}$ means that $\mathbf{x}_i^{(1)}$ is more relevant than $\mathbf{x}_i^{(2)}$ w.r.t. q_i, which can be expressed by $\mathbf{w} \cdot \left(\mathbf{x}_i^{(1)} - \mathbf{x}_i^{(2)}\right) > 0$. Otherwise, $\mathbf{x}_i^{(1)} \prec \mathbf{x}_i^{(2)}$ and $\mathbf{w} \cdot \left(\mathbf{x}_i^{(1)} - \mathbf{x}_i^{(2)}\right) < 0$. This way, the document sets with relevance labels can be transformed into a set of preference instances with labels $y_i = +1$ $\left(\mathbf{x}_i^{(1)} \succ \mathbf{x}_i^{(2)}\right)$ and $y_i = -1$ $\left(\mathbf{x}_i^{(2)} \succ \mathbf{x}_i^{(1)}\right)$. These preference instances are the training data for RankingSVM. The loss function of RankingSVM can be represented as follows.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i \left(\mathbf{w} \cdot \left(\mathbf{x}_i^{(1)} - \mathbf{x}_i^{(2)}\right)\right) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \end{aligned}$$

The LRHR algorithm. We have explained the main procedures of LRHR, we now summarize them and present the pseudocode in Algorithm 1.

Algorithm 1: The LRHR Algorithm

Input : A collection of user profiles U, a collection of item descriptions I, and a rating matrix R
Output: Ranking function f

1 $T \leftarrow$`ExtractTerms` (U, I)
2 $F \leftarrow$`ExtractFeatures` (U, I, R, T)
3 $(U, I \times I) \leftarrow$`Formulate` (U, I, R)
4 $f \leftarrow$`LearningToRank` $(U, I \times I, F)$

Line 1 extracts the vocabulary T for representing users and items. Line 2 extracts a set of features F, including meta-level features and rating-based features. Lines 3 formulates the ranking-based hybrid recommendation problem as a pairwise ranking problem, where pairs of items $I \times I$ associated with each user are generated based on their ratings. Line 4 learns a ranking function using a learning to rank algorithm such as RankSVM.

Discussion. The main technical challenges in adapting learning to rank for hybrid recommendation are to represent users and items with the same vocabulary and make them content-comparable, and to define meta-level features and rating-based features for learning purposes. For the actual learning to rank algorithm, many other algorithms can be considered besides RankSVM. For example, RankBoost, AdaRank and ListNet. While our experiments demonstrate initial success with RankSVM, the impact on recommendation quality from the various options of ranking algorithms needs to be further evaluated.

5. EXPERIMENTS

5.1 Methodology

We used MovieLens [1] dataset containing 1M movie ratings. In our experiments, the dataset was partitioned into 5 parts for 5-fold cross validation, where 4 parts were used for training and 1 part for testing, and the averaged performance was reported.

We used 5 benchmark recommendation algorithms as comparison partners. They included a hybrid recommender systems CBCF

[1](http://www.grouplens.org/node/73

[7] (content-boosted collaborative filtering), 2 ranking-based collaborative filtering algorithms including EigenRank [5] and CoFiRank [12], and 2 content-based filtering algorithms including NBCBF [8] (a conventional approach that is based on naive Bayesian text classification and adopted in CBCF) and LRCBF (our proposed learning to rank-based approach with only those content features).

Since our study focuses on improving item rankings instead of rating prediction, we employ the Normalized Discounted Cumulative Gain (NDCG) [3] metric for evaluation of the recommendation performance. This metric is popular in information retrieval for evaluating ranked results, where documents are assigned graded rather than binary relevance judgements.

In the context of collaborative filtering, item ratings assigned by users can naturally serve as graded relevance judgements. Specifically, the NDCG metric is evaluated over some number n of the top items on the ranked item list. Let U be the set of users and $r_{u,p}$ be the rating score assigned by user u to the item at the pth position of the ranked list from u. The NDCG at the nth position with respect to the given user u is defined as follows.

$$NDCG_u@n = Z_u \sum_{p=1}^{k} \frac{2^{r_{u,p}} - 1}{\log(1+p)}$$

For the set of users U, the average NDCG at the nth position is:

$$NDCG_{avg}@n = \frac{1}{|U|} \sum_{u \in U} NDCG_u@n$$

The value of NDCG ranges from 0 to 1. A higher value indicates better ranking effectiveness. NDCG is very sensitive to the ratings of the highest ranked items. This is modeled by the discounting factor $\log(1+p)$ that increases with the position in the ranking. This is a highly desirable characteristic for evaluating ranking quality in recommender systems. This is because, just as in Web search, most users only examine the first few items from the recommended list. The relevance of top-ranked items are far more important than other items [5].

5.2 Results

Figures 1 shows the performance comparison under the NDGG@1-5 measures. From the figure we can see that:

(1) LRHR significantly outperformed the comparison partners on the MovieLens dataset, gaining performance improvement from 3.58% to 7.78% under NDGG@1-5. For example, LRHR achieved 0.7524 and 0.7542 on NDCG@1 and 2 while these numbers of CoFiRank are 0.6981 and 0.7091, gaining 7.78% and 6.36% respectively.

(2) Hybrid algorithms outperformed collaborative filtering and content-based filtering. First of all, LRHR outperformed any other comparison partner including LRCBF, our proposed learning to rank-based approach with only those content-based features. Besides, CBCF also outperformed most of sophisticated content-based filtering and collaborative filtering algorithms, though it only adopted the most straightforward recommendation techniques such as naive Bayesian text classification for content-based prediction and Pearson correlation coefficient for rating-based similarity measures.

(3) Collaborative filtering algorithms EigenRank and CoFiRank significantly outperformed the conventional content-based filtering algorithm CBCBF . EigenRank gained performance improvement from 4.75% to 7.18% under NDGG@1-5, while CoFiRank gained from 2.66% to 7.67%.

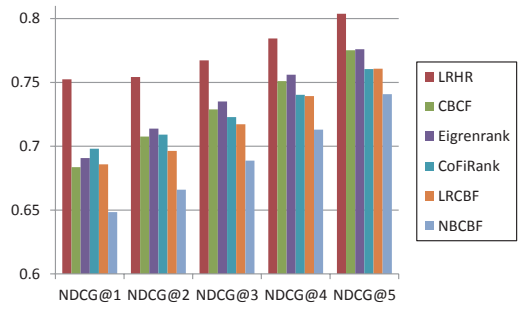

Figure 1: Performance under NDCG@n

6. CONCLUSION

In this paper we have described LRHR, the first attempt that effectively adapts learning to rank to hybrid recommender systems, gaining combined advantages from ranking-based and hybrid recommender systems. Experiments on benchmark datasets have demonstrated the promise of the approach. For future work, we plan to perform a systematic study on LRHR, investigating the various factors that may affect its performance. For example, the definition of meta-level features and the selection of learning to rank algorithms. In addition, while we target recommendation quality in this study, it is important to consider efficiency as in the case of learning to rank for information retrieval.

7. ACKNOWLEDGEMENT

This research was supported in part by the Natural Science Foundation of China (60970047, 61103151, 61173068, 71171122), the Foundation of Ministry of Education of China (20110131110028, 12YJC630211), the Natural Science Foundation of Shandong Province of China (ZR2012FM037, BS2012DX012), and the National Science Foundation (OCI-1062439, CNS-1058724).

8. REFERENCES

[1] S. Gopal and Y. Yang. Multilabel classification with meta-level features. In *SIGIR*, 2010.

[2] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.*, 10, 2009.

[3] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[4] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, 2002.

[5] N. N. Liu and Q. Yang. Eigenrank: A ranking-oriented approach to collaborative filtering. In *SIGIR*, 2008.

[6] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009.

[7] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI*, 2002.

[8] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *DL*, 2000.

[9] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[10] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *RecSys*, 2010.

[11] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artif. Intell.*, 2009:Article ID 421425.

[12] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Cofirank - maximum margin matrix factorization for collaborative ranking. In *NIPS*, 2011.

[13] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Trans. Knowl. Data Eng.*, 16(1):56–69, 2004.