Content-Based Top-N Recommendation using Heterogeneous Relations

Yifan Chen
National University of Defense Technology
15 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Xiang Zhao
National University of Defense Technology
56 PUBLICATIONS **227** CITATIONS
SEE PROFILE

Junkai Ren
University of Alberta
7 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Content-Based Top-N Recommendation using Heterogeneous Relations

Yifan Chen[1], Xiang Zhao[1], Junjiao Gan[2], Junkai Ren[1], and Yang Fang[1]

[1] National University of Defense Technology, China
[2] Massachusetts Institute of Technology, United States

Abstract. Top-N recommender systems have been extensively studied. However, the sparsity of user-item activities has not been well resolved. While many hybrid systems were proposed to address the *cold-start* problem, the profile information has not been sufficiently leveraged. Furthermore, the heterogeneity of profiles between users and items intensifies the challenge. In this paper, we propose a content-based top-N recommender system by learning the global term weights in profiles. To achieve this, we bring in PathSim, which could well measures the node similarity with heterogeneous relations (between users and items). Starting from the original TF-IDF value, the global term weights gradually converge, and eventually reflect both profile and activity information. To facilitate training, the derivative is reformulated into matrix form, which could easily be paralleled. We conduct extensive experiments, which demonstrate the superiority of the proposed method.

1 Introduction

Recommender systems typically leverage two types of signals to effectively recommend *items* to *users* - user activities, and content matching between user and item profiles. Depending on what to use, the recommendation models in literature are usually categorized into collaborative filtering, content-based and hybrid models [1]. In real-world applications, solely employing collaborative filtering or content-based models can not achieve desirable results, as is often the case that single source of information tends to be incomplete.

To better illustrate, we motivate the following example in architecture. Recently, `Vanke`, a leading real-estate corporation in China, started a Uber-For-Architects project, namely `NOSPPP`, which tries to match architects with appropriate projects based on previous project information of architects and firms. During the running of `NOSPPP`, the data collected are two-fold, with the participated projects and the resumes, respectively. In terms of recommender system, the former is named “feedback” (of users) while the latter is referred as “profiles” (of items). Due to the sparsity of feedback, collaborative filtering based recommenders would face the *cold-start* problem, and hence, we have to resort to content-based or hybrid models [9]. However, unlike the applicant-job scenario therein, where the profiles of users and jobs could well match, the profiles of architects and projects describe things in two different worlds. Specifically, the profiles of architects presents the working experience and skills, while the profiles of projects tells the area, the interior and exterior constructions, etc. This is rational, as designing architectures is in the form of art, where it is hard to specify the conditions or requirements through decomposition. In response to applications like `NOSPPP`, we explore recommendation utilizing both *sparse feedback* and *heterogeneous profiles*.

In this paper, we exploit a hybrid recommendation method that ensembles both sources of information. For ease of exposition, we consider the case that auxiliary information exists only on the side of items, and propose a item-based top-N recommendation algorithm [1]. Classic item-based collaborative filtering uses the direct link information for recommendation without diffusing the influence of other user-item links. By regarding user-item interactions as a bi-type information network, we observe that such influence can be captured by item node similarity, where PathSim [18] via *meta-path* is served. Moreover, while content matching between heterogeneous profiles of users and items does not produce explicable results, methods including [23, 22, 19, 21, 20] suggest high similarity among objects within the same subspace, thus we contend that it can be employed for matching profiles between item-item or user-user, since profiles of same type is naturally homogeneous. A standard way to measure similarity between two profiles is computing the cosine similarity of the two bags of words, and each word is weighted by term frequency tf (within the document) $\times$ inverted document frequency *idf* (of the term within the corpus). While the local term frequency could be computed offline, it has been suggested [9] that the global term weights *idf* requires further optimization to achieve better precision. Thus, we optimize the global term weights with the guidance of the similarity from PathSim.

In summary, the major contribution of the paper is a novel hybrid recommendation method, the overview of which is outlined as follows:

(1) Derive item similarity measured by meta-paths using PathSim;
(2) Optimize the global term weights guided by PathSim; and
(3) Recommend top-N items based on nearest neighbor collaborative filtering.

Organization. Section 2 discusses related work. We present the method for deriving initial similarity between items in Section 3, and then, introduce the learning method for optimizing global term weights in Section 4. Experiment results are in Section 5, followed by conclusion in Section 6.

2 Related Work

Top-N recommender systems have been extensively studied during the last few years, which could be classified into three categories.

[1] Without loss of generality, it is straightforward to extend the idea to the case of auxiliary information on both sides of users and items.

The first category is neighborhood-based collaborative filtering, which could be further classified into three classes: item-based and user-based. Given a certain user, *user-based-nearest-neighbor* (userkNN) [12, 7, 15] first identifies a set of similar users, and then recommends top-N items based on what items those similar users have purchased. Similarly, *item-based-nearest-neighbor* (itemkNN) [16] identifies a set of similar items for each of the items that the user has purchased, and then recommends top-N items based on those similar items.There are plenty of ways to measure user/item similarity, e.g., Pearson correlation, cosine similarity, and so forth.

The second category is model-based collaborative filtering, in which the latent factor models have achieved the state-of-the-art performance. Cremonesi et.al. proposed a simple model-based algorithm PureSVD [6], where users' features and items' features are represented by the principle singular vectors of the user-item matrix. Koren proposed the well-known SVD++ model [6]. Wu applied Regularized Matrix Factorization (RMF), Maximum Margin Matrix Factorization (MMMF), and Nonnegative Matrix Factorization (NMF) to recommender systems [24]. Weighted Regularized Matrix Factorization (WRMF) was introduced by Hu et al. [10]. The key idea of these methods is to factorize the user-item matrix to represent the users preferences and items characteristics in a common latent space, and then estimate the user-item matrix by the dot product of user factors and item factors. All these methods assume that only a few variables impact users preference and items features, which means the low-rank structure of user-item matrix.

Another model-based method, SLIM, proposed by Ning et. al. [13], predicts the user-item matrix by multiplying the observed user-item matrix by the aggregation coefficient matrix. SLIM estimates the coefficient matrix by learning from the observed user-item matrix with a simultaneous regression model. Specifically, it introduces sparsity with ℓ_1-norm regularizer into the regularized optimization and formed an elastic net problem to benefit from the smoothness of ℓ_2-norm and the sparsity of ℓ_1-norm. Later, plenty of research has been done based on SLIM. SSLIM [14] integrates the side information. LorSLIM [4] involves the nuclear-norm to induce the low-rank property of SLIM. HOSLIM [5] uses the potential higher-order information to generate better recommendation.

The last category is hybrid methods. Hybrid method is used to combine the virtue of different recommender algorithms to generate better performance. A hybrid method was used to deal with the cold-start scenarios by mapping entities (user/item attributes) to latent features of a matrix factorization model [8].

3 Initializing Item Similarity

This section present the method for measuring item similarity.

3.1 Preliminaries

Definition 1 (Information Network). *An information network is defined as a directed graph $G = (V, E)$ with an object type mapping function $\sigma : V \to \mathcal{A}$*

*and a link type mapping function $\varphi : E \to \mathcal{R}$, where each object $v \in V$ belongs to one particular object type $\sigma(v) \in \mathcal{A}$, each link $e \in E$ belongs to a particular relation $\varphi(e) \in \mathcal{R}$, and if two links belong to the same relation type, the two links share the same starting object type as well as the ending object type. Note that, if a relation exists from type A to type B, denoted as ARB, the inverse relation R^{-1} holds naturally for $BR^{-1}A$. R and its inverse R^{-1} are usually not equal, unless the two types are the same and R is symmetric. When the types of objects $|\mathcal{A}| > 1$ or the types of relations $|\mathcal{R}| > 1$, the network is called **heterogeneous** information network; otherwise, it is a **homogeneous** information network.*

Definition 2 (Network Schema). *The network schema, denoted as $T_G = (\mathcal{A}, \mathcal{R})$, is a meta template for a heterogeneous network $G = (V, E)$ with the object type mapping $\sigma : V \to \mathcal{A}$ and the link mapping $\varsigma : E \to \mathcal{R}$, which is a directed graph defined over object types $\mathcal{A}$, with edges as relations from $\mathcal{R}$.*

Definition 3 (Meta-path). *A meta-path $\mathcal{P}$ is a path defined on the graph of network schema $T_G = (\mathcal{A}, \mathcal{R})$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_{l+1}$. For simplicity, the meta-path can be denoted by the type names if there exist no multiple relations between the same pair of types: $\mathcal{P} = (A_1 A_2 \ldots A_{l+1})$. A path $p = (a_1 a_2 \ldots a_{l+1})$ between a_1 and a_{l+1} is said to follow the meta-path $\mathcal{P}$, if $\forall i, a_i \in A_i$ and each link $e_i = \langle a_i a_{i+1} \rangle$ belongs to each relation R_i in $\mathcal{P}$. These paths are called path* instances *of $\mathcal{P}$, denoted as $p \in \mathcal{P}$.*

In the following, meta-path is confined to symmetric, namely *round trip meta-paths* in the form of $\mathcal{P} = (\mathcal{P}_l \mathcal{P}_l^{-1})$.

Definition 4 (Commuting Matrix). *Given a network $G = (V, E)$ and its network schema T_G, a commuting matrix M for a meta-path $\mathcal{P} = (A_1 A_2 \ldots A_l)$ is defined as $M = W_{A_1 A_2} W_{A_2 A_3} W_{A_{l-1} A_l}$, where $W_{A_i A_j}$ is the weight matrix between type A_i and type A_j.*

Definition 5 (PathSim). *Given a symmetric meta-path $\mathcal{P}$, PathSim between two objects v_i and v_j of the same type is:*

$$s_{ij} = \frac{2M_{ij}}{M_{ii} + M_{jj}}, \tag{1}$$

where M_{ij} represents the i^{th} row and j^{th} column element of matrix M.

3.2 Measuring Item Similarity

To measure item similarity through PathSim, we first define the meta-path in the form of $\mathcal{P}_n = (A(BA)^n)$ (the mined frequent patterns [3]). For instance, $n = 1$ corresponds to $\mathcal{P}_1 = (ABA)$ and $n = 2$ corresponds to $\mathcal{P}_2 = (ABABA)$. It is easy to verify the symmetry of $\mathcal{P}_n$ and thus PathSim can be applied. The associated commuting matrix for $\mathcal{P}_n$ is $M = (W_{AB} W_{BA})^n$ and consequently the similarity between item i and item j can be computed by Equation (1).

Suppose we define N meta-paths $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N$, with the corresponding similarities $s_1, s_2, \ldots, s_N$, the overall similarity should be measured as the weighted aggregation, e.g. $s = \sum_{n=1}^{N} \alpha_n s_n$, where $\sum_{n=1}^{N} \alpha_n = 1$. As is suggested in [18] that the meta-path with relatively short length is good enough to measure similarity, and a long meta-path may even reduce the quality, we set smaller weights for longer meta-paths. We naturally set the weights as $\alpha_n = \frac{2^{N-n}}{2^N-1}$. We further denote $\boldsymbol{S}^p$ for the matrix of PathSim, where s_{ij}^p represents the element of $\boldsymbol{S}^p$ in the i^{th} row and j^{th} column.

4 Optimizing Profile Similarity

We prompt to measure the item similarity based on the profiles. Prior to the discussion, we first list the notations used in this section in Table 1. Note that vectors and matrices are made **bold**.

Table 1. Table of Notations

N_u	Number of users
N_v	Number of items
N_w	Number of terms
λ	ℓ_2 norm weight
$\boldsymbol{S}^f, s_{ij}^f$	similarity derived from item profiles
$\boldsymbol{S}^p, s_{ij}^p$	similarity derived form pathsim
$\boldsymbol{W}^l, \boldsymbol{w}_i^l, w_{ik}^l$	local term weights
$\boldsymbol{w}^g, w_{ik}^g$	global term weights
$\boldsymbol{w}, w_k$	the weights to learn, where $w_k = (w_k^g)^2$
$\boldsymbol{P}, \boldsymbol{p}_k$	the normalized $tf \times idf$ weights

Each profile contains rich text to describe the feature. Thus more effective content analysis methods and text similarity measures are crucial for the recommendation. Most designed recommender systems involving text similarity measure applied cosine similarity of two bags of words, where each word is weighted by $tf \times idf$ [2, 17]. Nevertheless, it is possible to go beyond the definition of $tf \times idf$, where tf represents the local term weights and idf the global term weights. While tf could be derived offline with various methods, idf requires further optimization as suggested by [9]. Thus the global term could be optimized with the guidance of PathSim and the similarity derived from profiles could be calculated by the following Equation:

$$s_{ij}^f = \frac{\boldsymbol{d}_i \cdot \boldsymbol{d}_j}{\|\boldsymbol{d}_i\|_2 \|\boldsymbol{d}_j\|_2}, \tag{2}$$

where $\|\cdot\|_2$ is the ℓ_2 norm of a vector, and $\boldsymbol{d}_i$ represents the term vector, each dimension represents a term, and the value in each dimension represents the weight of the term. $\boldsymbol{d}_i$ could be decomposed as $\boldsymbol{w}_i^l \circ \boldsymbol{w}^g$, where $\boldsymbol{w}_i^g$ denotes for the local weights for item i. $\boldsymbol{w}^g$ denotes for the global term weights, which is initially set with the original Inverted Document Frequency, and optimized gradually. $\circ$ is a binary operation, conducting the element-wise product of two vectors, thus the result is also a vector. By letting $w_k = (w_k^g)^2$, we could further formalize Equation 2 as follows:

$$s_{ij}^f = \frac{\sum_{k=1}^t w_{ik}^l w_{jk}^l w_k}{\left[\sum_{k=1}^t (w_{ik}^l)^2 w_k\right]^{\frac{1}{2}} \left[\sum_{k=1}^t (w_{jk}^l)^2 w_k\right]^{\frac{1}{2}}},$$

and the partial derivative could be derived as:

$$\begin{aligned}\frac{\partial s_{ij}^f}{\partial w_k} &= \frac{1}{\|\boldsymbol{d}_i\|_2^2 \|\boldsymbol{d}_j\|_2^2} \left\{ w_{ik}^l w_{jk}^l \|\boldsymbol{d}_i\|_2 \|\boldsymbol{d}_j\|_2 - \left[\frac{\|\boldsymbol{d}_j\|_2}{2\|\boldsymbol{d}_i\|_2} (w_{ik}^l)^2 + \frac{\|\boldsymbol{d}_i\|_2}{2\|\boldsymbol{d}_j\|_2} (w_{jk}^l)^2 \right] \boldsymbol{d}_i \cdot \boldsymbol{d}_j \right\} \\ &= \frac{w_{ik}^l w_{jk}^l}{\|\boldsymbol{d}_i\|_2 \|\boldsymbol{d}_j\|_2} - \frac{s_{ij}^f}{2} \left[\frac{(w_{ik}^l)^2}{\|\boldsymbol{d}_i\|_2^2} + \frac{(w_{jk}^l)^2}{\|\boldsymbol{d}_j\|_2^2} \right].\end{aligned}$$

To optimize the global term weights, we should define the loss function to measure the difference between s_{ij}^p and s_{ij}^f. We develop the squared loss function and the associated optimization methods.

4.1 Squared Error Loss Function

Due to the sparsity of the user-item information network, we could also expect the sparsity of similarities measured by PathSim. If item i can not reach item j through the bi-type information network, according to Equation 1, $s_{ij} = 0$.

In this section, the loss function is defined as the squared error, given by

$$\mathcal{L} = \sum_{i=1}^{N_v} \sum_{j=1}^{N_v} (s_{ij}^f - s_{ij}^p)^2 = \|\boldsymbol{S}^f - \boldsymbol{S}^p\|_F^2,$$

based on which, we could minimize the following objective function to optimize the global term frequency:

$$\begin{aligned}\min_{\boldsymbol{w}} J &= \frac{1}{2}\|\boldsymbol{S}^f - \boldsymbol{S}^p\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\ s.t.\ &\boldsymbol{w} \geq 0\end{aligned} \tag{3}$$

where $\|\cdot\|_F$ is the Frobenius norm, which is actually the squared sum of all elements of the matrix. $\boldsymbol{w}$ stands for the vector of w_k, and we penalize ℓ_2 norm on the global term weights $\boldsymbol{w}$ to avoid over fitting and sparsity result. $\boldsymbol{S}^p$ is denoted for PathSim matrix, whereas $\boldsymbol{S}^f$ for the profile similarity matrix. We reformulate the problem into the following element-wise form, to facilitate the deduction of partial derivative over w_k, e.g., $\frac{\partial J}{\partial w_k}$.

$$\min_{w_k} J = \frac{1}{2}\sum_{i=1}^{N_v}\sum_{j=1}^{N_v}(s_{ij}^f - s_{ij}^p)^2 + \frac{\lambda}{2}\sum_{i=1}^{N_w} w_k^2$$
$$w_k \geq 0, k = 1, \ldots, N_w$$

Solution. The partial derivative is given in Equation 4.

$$\frac{\partial J}{\partial w_k} = \sum_{i=1}^{N_v}\sum_{j=1}^{N_v}(s_{ij}^f - s_{ij}^p)\left\{\frac{w_{ik}^l w_{jk}^l}{\|\boldsymbol{d}_i\|_2\|\boldsymbol{d}_j\|_2} - \frac{\tilde{s}_{ij}}{2}\left[\frac{(w_{ik}^l)^2}{\|\boldsymbol{d}_i\|_2^2} + \frac{(w_{jk}^l)^2}{\|\boldsymbol{d}_j\|_2^2}\right]\right\} + \lambda w_k. \quad (4)$$

We further define $q_{ij} = s_{ij}^f - s_{ij}^p, p_{ik} = \frac{w_{ij}^l}{\|\boldsymbol{d}_i\|_2}$ and $r_{ij} = (s_{ij}^f - s_{ij}^p)s_{ij}^f$. Thus we have:

$$\sum_{i=1}^{N_v}\sum_{j=1}^{N_v}(s_{ij}^f - s_{ij}^p)\frac{w_{ik}^l w_{jk}^l}{\|\boldsymbol{d}_i\|_2\|\boldsymbol{d}_j\|_2} = \sum_{i=1}^{N_v}\sum_{j=1}^{N_v} q_{ij}p_{ik}p_{jk} = \boldsymbol{p}_k^T\boldsymbol{Q}\boldsymbol{p}_k$$

$$\sum_{i=1}^{N_v}\sum_{j=1}^{N_v}(s_{ij}^f - s_{ij}^p)\frac{s_{ij}^f}{2}\left[\frac{(w_{ik}^l)^2}{\|\boldsymbol{d}_i\|_2^2} + \frac{(w_{jk}^l)^2}{\|\boldsymbol{d}_j\|_2^2}\right] = \sum_{i=1}^{N_v}\sum_{j=1}^{N_v}\frac{1}{2}r_{ij}(p_{ik}^2 + p_{jk}^2)$$
$$= \sum_{i=1}^{N_v}\sum_{j=1}^{N_v} r_{ij}p_{ik}^2 = \boldsymbol{p}_k^T\boldsymbol{R}\boldsymbol{p}_k.$$

where $\boldsymbol{p}_k$ is a vector of p_{ik}, $\boldsymbol{Q}$ is $N_v \times N_v$ matrix of q_{ij} and $\boldsymbol{R}$ is a diagonal matrix with the i-th element of principal diagonal equals $\sum_j r_{ij}$. By defining $\boldsymbol{L} = \boldsymbol{Q} - \boldsymbol{R}$, we find the following close form of derivative:

$$\frac{\partial J}{\partial w_k} = \boldsymbol{p}_k^T\boldsymbol{L}\boldsymbol{p}_k + \lambda w_k.$$

It could be further represented into the matrix form:

$$\frac{\partial J}{\partial \boldsymbol{w}} = \mathsf{diag}(\boldsymbol{P}^T\boldsymbol{L}\boldsymbol{P}) + \lambda\boldsymbol{w}, \quad (5)$$

where $\mathsf{diag}(\cdot)$ extracts the principal diagonal and form as a vector.

Following the common practices for top-N recommendation [11], the loss function is computed over all entries of $\boldsymbol{S}$. The summation above contains $n \times n$ terms, namely all pairwise items in the dataset. To ensure good performance while achieve reasonable training time, the algorithm is paralleled by CUDA.

5 Experimental Evaluation

To evaluate our proposed method, extensive experiments have been conducted. However, due to space limitation, we only present part of the results.

5.1 Experiment Setup

The results reported in this section is based on the NIPS dataset[2]. It contains paper-author and paper-word matrices extracted from co-author network at the NIPS conference over 13 volumes. We regard authors as users, papers as items and the contents of papers as the profile of items. Thus the data has 2037 users (authors) and 1740 items (papers), where 13649 words have been extracted from the corpus of item profiles. The content of the papers is preprocessed such that all words are converted to lower case and stemmed and stop-words are removed. One may note that NIPS dataset is very sparse, that is, some author may publish only one or two papers, which shows the importance of properly leveraging side information for recommendation.

We applied 5-time Leave-One-Out cross validation (LOOCV) to evaluate our proposed method. In each run, each of the dataset is split into a training set and a testing set by randomly selecting one of the non-zero entries of each user and placing it into the testing set. The training set is used to train a model, then for each user a size-N ranked list of recommended items is generated by the model. We varies N as 5,10,15,20 to compare the result difference. Our method has two parameters, n_p and λ. n_p measures the length of meta-path and λ measures the degree of regularization.

The recommendation quality is measured using Hit Rate (HR) and Average Reciprocal Hit Rank (ARHR) [7]. HR is defined as

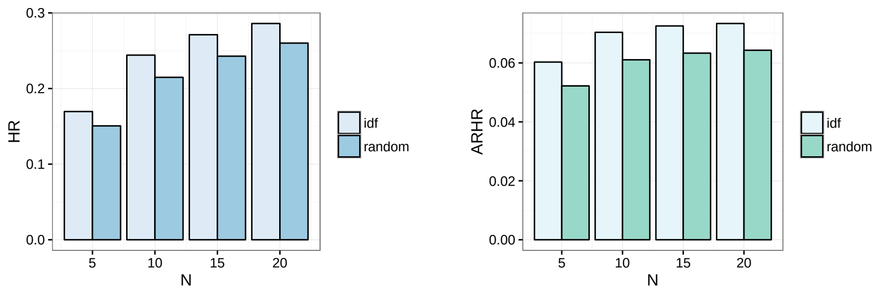
$$HR = \frac{\#hits}{\#users},$$

where $\#users$ is the total number of users and $\#hits$ is the number of users whose item in the testing set is recommended (i.e., hit) in the size-N recommendation list. A second measure for evaluation is ARHR, which is defined as

$$ARHR = \frac{1}{\#users} \sum_{i=1}^{\#hits} \frac{1}{p_i},$$

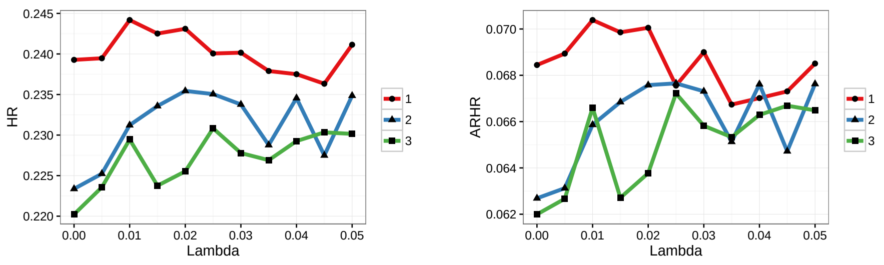
where if an item of a user is hit, p is the position of the item in the ranked recommendation list. ARHR is a weighted version of HR and it measures how strongly an item is recommended, in which the weight is the reciprocal of the hit position in the recommendation list.

We implement our algorithm in C++. As our method involves optimizing the global weights over the whole vocabulary of item profiles, to expedite the training efficiency, the training process is paralleled in GPU and implemented by CUDA[3]. All experiments are done on a machine with 4-core Intel i7-4790 processor at 3.60GHz and Nvidia GeForce GTX TITAN X graphics card.



(a) HR (b) ARHR

Fig. 1. Effect of Initial Value



(a) HR (b) ARHR

Fig. 2. Effect of Parameter λ and n_p

5.2 Effect of Initial Value

We first evaluate the influence of initial value we set for global weights on the performance. We compare two settings, random and idf. The initial value is randomly set in the first setting while it is set as the value of inverted document frequency in the latter one. Here λ is set as 0.01 and n_p as 1. The result is reported in Figure 1, which shows the superiority of idf over random. The result demonstrates the usefulness of side information in this dataset and we set the initial value of global term as idf thereafter.

5.3 Effect of Parameters

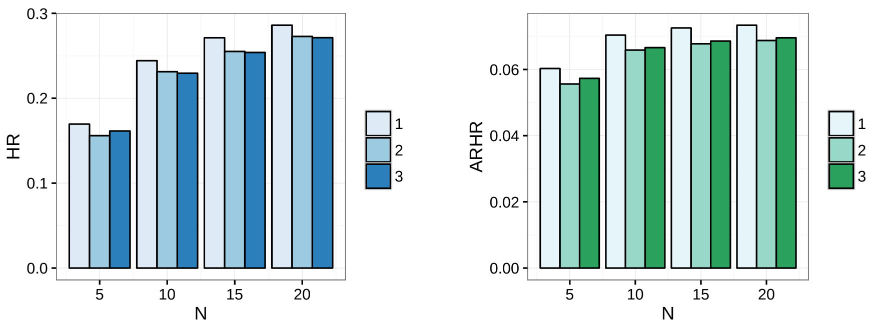
In this set of experiments, the validation is conducted to select the most suitable parameters. λ is varied from 0 to 0.05 and stepped 0.005 and n_p is set as 1,2,3. We draw the lines in Figure 2. Three lines are drawn to distinguish $n_p = 1$ (red line),$n_p = 2$ (blue line) and $n_p = 3$ (green line) respectively. The result shows that n_p should be set 1 to achieve better performance. This result is consistent

[2] http://www.cs.nyu.edu/~roweis/data.html
[3] http://www.nvidia.cn/object/cuda-cn.html

with [18], which suggests shorter length of meta-path is good enough to measure similarity.

As Figure 2 depicts the performance along with λ, we find the best value as 0.01 for $n_p = 1$, 0.02 for $n_p = 1$ and 0.025 for $n_p = 3$. It has also been shown that the method performs more robustly when $n_p = 1$ while it varies dramatically with λ when $n_p > 1$. Based on the observation, we finally pick n_p and λ as 1 and 0.01 for the rest of the experiments.

5.4 Recommendation for Different Top-N



(a) HR (b) ARHR

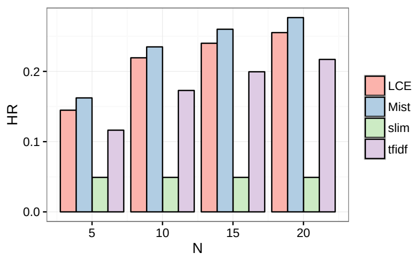
Fig. 3. Performance of Proposed Method

By setting the global term as the inverted document frequency, and letting $\lambda = 0.01$, we evaluate the top-N recommendation performance, the result of which is illustrated in Figure 3. Obviously, with the increase of N, the performance improves. We also compare the different setting of n_p, which further demonstrates $n_p = 1$ could be a better choice. We also found in this set of experiments that when N increase from 5 to 10, the performance shows relatively higher improvement.

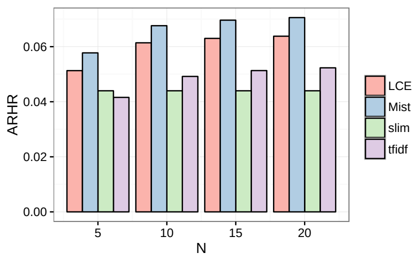
5.5 Comparison of Algorithms

We finally compares our method with other algorithms in this set of experiments. As top-N recommendation methods have been extensively studied, we compare only with some state-of-the-art methods, e.g. Slim [13] and LCE [17]. We also incorporate the pure tfidf method to calculate the item similarity for recommendation. To distinguish, we name our proposed method as Mist (Meta path based item similarity to learn global term weights). We depict the result in Figure 4, where all the compared algorithms were optimized to the best settings.

Figure 4(a) shows the recommendation of Mist is consistently better than other three methods. Note that Slim has the worst performance, this could be



(a) HR



(b) ARHR

Fig. 4. Algorithm Comparison

attributed to the sparsity of dataset. LCE also took advantage of item profiles, thus it has achieved good performance. It is also worth noting that the pure tfidf shows relatively acceptable results and Mist could be regarded as the collaborative optimized tfidf.

When it comes to ARHR, showing in Figure 4(b), Mist also behaves the best, which was followed by LCE, tfidf and Slim. In conclusion, the learned global term weights can well capture both structural and textual information.

6 Conclusion

In this paper, we proposed a content-based top-N recommender system by leveraging item profiles. We first employed PathSim to measure the item similarity on the top of heterogeneous relations between users and items, and then optimized the global term weights towards the PathSim similarities. To facilitate training, the derivation was reformulated into matrix form, which could easily be paralleled. We conducted extensive experiments, and the experimental results demonstrate the superiority of the proposed method.

References

1. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Communications of the ACM 40(3), 66–72 (1997)
2. Barjasteh, I., Forsati, R., Masrour, F., Esfahanian, A., Radha, H.: Cold-start item and user recommendation with decoupled completion and transduction. In: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015. pp. 91–98 (2015)
3. Chen, Y., Zhao, X., Lin, X., Wang, Y.: Towards frequent subgraph mining on single large uncertain graphs. In: 2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015. pp. 41–50 (2015)
4. Cheng, Y., Yin, L., Yu, Y.: LorSLIM: Low rank sparse linear methods for top-n recommendations. In: 2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014. pp. 90–99 (2014)

5. Christakopoulou, E., Karypis, G.: HOSLIM: higher-order sparse linear method for top-n recommender systems. In: Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part II. pp. 38–49 (2014)
6. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010. pp. 39–46 (2010)
7. Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Trans. Inf. Syst. 22(1), 143–177 (2004)
8. Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning attribute-to-feature mappings for cold-start recommendations. In: ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010. pp. 176–185 (2010)
9. Gu, Y., Zhao, B., Hardtke, D., Sun, Y.: Learning global term weights for content-based recommender systems. In: Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016. pp. 391–400 (2016)
10. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy. pp. 263–272 (2008)
11. Kabbur, S., Ning, X., Karypis, G.: FISM: factored item similarity models for top-n recommender systems. In: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013. pp. 659–667 (2013)
12. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001. pp. 247–254 (2001)
13. Ning, X., Karypis, G.: SLIM: sparse linear methods for top-n recommender systems. In: 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011. pp. 497–506 (2011)
14. Ning, X., Karypis, G.: Sparse linear methods with side information for top-n recommendations. In: Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012. pp. 155–162 (2012)
15. Papagelis, M., Plexousakis, D.: Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. Engineering Applications of Artificial Intelligence 18(7), 781–789 (2005)
16. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: CSCW '94, Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, October 22-26, 1994. pp. 175–186 (1994)
17. Saveski, M., Mantrach, A.: Item cold-start recommendations: learning local collective embeddings. In: Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014. pp. 89–96 (2014)
18. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: PathSim: Meta path-based top-k similarity search in heterogeneous information networks. PVLDB 4(11), 992–1003 (2011)
19. Wang, Y., Lin, X., Wu, L., Zhang, W.: Effective multi-query expansions: Robust landmark retrieval. In: Proceedings of the 23rd Annual ACM Conference on Multi-

media Conference, MM '15, Brisbane, Australia, October 26 - 30, 2015. pp. 79–88 (2015)

20. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q.: Exploiting correlation consensus: Towards subspace clustering for multi-modal data. In: Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014. pp. 981–984 (2014)
21. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q.: LBMCH: learning bridging mapping for cross-modal hashing. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015. pp. 999–1002 (2015)
22. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q., Huang, X.: Robust subspace clustering for multi-view data by exploiting correlation consensus. IEEE Trans. Image Processing 24(11), 3939–3949 (2015)
23. Wang, Y., Zhang, W., Wu, L., Lin, X., Zhao, X.: Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion. IEEE Transactions on Neural Networks and Learning Systems (2015)
24. Wu, M.: Collaborative filtering via ensembles of matrix factorizations. In: Proceedings of KDD Cup and Workshop. vol. 2007 (2007)