A Novel Hierarchical Approach to Ranking-Based Collaborative Filtering[⋆]

Athanasios N. Nikolakopoulos[1,2], Marianna Kouneli[1], and John Garofalakis[1,2]

[1] Computer Engineering and Informatics Department, University of Patras
[2] CTI and Press "Diophantus"
{nikolako,kounelim,garofala}@ceid.upatras.gr

Abstract. In this paper, we propose a novel recommendation method that exploits the intrinsic hierarchical structure of the item space to overcome known shortcomings of current collaborative filtering techniques. A number of experiments on the `MovieLens` dataset, suggest that our method alleviates the problems caused by the sparsity of the underlying space and the related limitations it imposes on the quality of recommendations. Our tests show that our approach outperforms other state-of-the-art recommending algorithms, having at the same time the advantage of being computationally attractive and easily implementable.

Keywords: Recommender Systems, Collaborative Filtering, Sparsity, Ranking Algorithms, Experiments

1 Introduction

Collaborative Filtering (CF) is widely regarded as one of the most successful approaches to building Recommender Systems (RS). The great impact of CF on Web applications, and its wide deployment in important commercial environments, have led to the significant development of the theory over the past decade, with a wide variety of algorithms and methods being proposed [3]. In the majority of these algorithms the recommendation task, reduces to *predicting* the ratings for all the unseen user-item pairs, using the root mean square error (RMSE) between the predicted and actual ratings as the evaluation metric [3,8,9]. Recently, however, many leading researchers have pointed out that the use of RMSE criteria to evaluate RS is not an adequate performance index [1,2,10]; they showed that even sophisticated methods trained to perform well on MSE/RMSE, do not behave particularly well on the – much more common in practice – top-k recommendation task [2].

These observations have turned significant research attention to *ranking-based* recommendation methods which are believed to conform more naturally with how the recommender system will actually be used in practice [1,2]. Fouss et al. [5] follow a graph representation of the data and, using an approach based on random walks, they present a number of methods to compute node similarity

[⋆] The final publication is available at link.springer.com measures, including the average commute time (normal CT and PCA-CT), and the pseudo-inverse of the Laplacian matrix ($\mathbf{L}^{\dagger}$), which they compare against standard approaches such as MaxF and Katz. Gori and Pucci [7] propose ItemRank (IR); a PageRank-inspired scoring algorithm that produces a personalized ranking vector using an items' correlation graph. Zhang et al. [13] propose TR, an improvement of ItemRank based on topical PageRank algorithm that takes into account item genre and user interest profiles. Recently, Freno et al. [6] proposed a Hybrid Random Fields model (HRF) which they applied, together with a number of well-known probabilistic graphical models, including Dependency Networks (DN), Markov Random Fields (MRF) and Naive Bayes (NB), to predict top-N items for users.

Despite their success in many application settings, ranking-based CF techniques encounter a number of problems that remain to be resolved. The unprecedented growth of the number of users and listed items in modern e-commerce applications, make many current techniques suffer serious computational and scalability issues that restrain their applicability in realistic scenarios. Additionally, an even more important problem that limits the quality of recommendations arises when available data are insufficient for identifying similar elements and is commonly referred to as the *Sparsity* problem. Sparsity is intrinsic to recommender systems because users typically interact with only a small portion of the available items, and the problem is aggravated by the fact that new items with no ratings at all, are regularly added to the system.

In this work, based on the intuition behind a recently proposed Web ranking framework [11], we describe a new recommendation method that exploits the innately hierarchical nature of the underlying spaces to characterize inter-item relations in a macroscopic level. Central to our approach is the idea that blending together the *direct* as well as the *indirect* inter-item relations can help reduce the sensitivity to sparseness and improve the quality of recommendations. To this end, we develop *Hierarchical Itemspace Rank* (HIR); a novel recommender method that brings together the above components in a generic and mathematically attractive way. To verify the merits of our approach we run a number of experiments on the standard `MovieLens` dataset, which show that HIR outperforms other state-of-the-art recommending techniques in widely used metrics, proving at the same time to be less susceptible to sparsity related problems.

2 The HIR Framework

In this section, after describing formally the core components of our model (Section 2.1), we proceed to the rigorous mathematical definition of the involving matrices, and finally, we present the Hierarchical Itemspace Rank algorithm and discuss its computational and storage needs (Section 2.2).

2.1 Model Definition

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be a set of *users* and $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ a set of *items*. Let $\mathcal{R}$ be a set of tuples $t_{ij} = (u_i, v_j, r_{ij})$, where r_{ij} is a nonnegative number referred to as the *rating* given by user u_i to the item v_j. These ratings can either come from the explicit feedback of the user or inferred by the user's behavior and interaction with the system. We consider a partition $\{\mathcal{L}, \mathcal{T}\}$ of the ratings into a *training set* $\mathcal{L}$ and a *test set* $\mathcal{T}$. For each user u_i, we denote $\mathcal{L}_i$ the set of items rated by u_i in $\mathcal{L}$, and $\mathcal{T}_i$ the set of items rated by u_i in $\mathcal{T}$. Formally: $\mathcal{L}_i \triangleq \{v_k : t_{ik} \in \mathcal{L}\}$ and $\mathcal{T}_i \triangleq \{v_l : t_{il} \in \mathcal{T}\}$.

Each user u_i is associated with a vector $\boldsymbol{\omega}^i \triangleq [\omega_1^i, \omega_2^i, \dots, \omega_m^i]$, whose nonzero elements contain the user's ratings that are included in the training set $\mathcal{L}$, normalized to sum to one. We refer to this as the *preference vector* of user u_i.

We consider a family of sets $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K\}$, defined over the underlying space, $\mathcal{V}$, according to a given criterion (e.g the categorization of movies into genres), such that $\mathcal{V} = \bigcup_{k=1}^{K} \mathcal{D}_k$, holds. We also define $\mathcal{G}_v$ to be the union of sets $\mathcal{D}_I$ that contain v and we use N_v to denote the number of different sets in $\mathcal{G}_v$. As we will see below, these sets will form the basis for the characterization of the indirect inter-item relations.

Having defined the parameters of our model, we are now ready to introduce matrices $\mathbf{C}$ and $\mathbf{D}$ that bring together the direct as well as the hierarchical structure of the underlying space, *in order to map each user's preference vector to a personalized distribution vector over the item space.*

Direct Association Matrix C: Matrix $\mathbf{C}$ is designed to capture the direct relations between the elements of $\mathcal{V}$. Generally, every such element will be associated with a discrete distribution $\mathbf{c}_v = [c_1, c_2, \cdots, c_m]$ over $\mathcal{V}$, that reflects the similarities between these elements. In our case (and for all the experiments presented in Section 3), we use the weighted mean of: (a) the correlation matrix [7] and (b) a row normalized version of the standard adjusted cosine similarity matrix [12]. These matrices are formally described below.

We first need to define a matrix $\mathbf{Q}$ whose ij^{th} element is $Q_{ij} \triangleq |\mathcal{U}_{ij}|$, where $\mathcal{U}_{ij} \subseteq \mathcal{U}$ denotes the set of users who rated both items v_i and v_j, i.e. $\mathcal{U}_{ij} \triangleq \{u_k : (v_i \in \mathcal{L}_k) \wedge (v_j \in \mathcal{L}_k)\}$ for $i \neq j$. Then, if we use $\hat{\mathbf{Q}}$ to denote the row normalized version of $\mathbf{Q}$, i.e. the matrix where every nonzero row sums up to 1, the resulting matrix will be defined as follows:

$$\mathbf{H} \triangleq \hat{\mathbf{Q}} + \frac{1}{m}\mathbf{a}\mathbf{e}^{\intercal} \tag{1}$$

where $\mathbf{a}$ is a vector that indicates the zero rows of matrix $\mathbf{Q}$ (i.e. its i^{th} element is 1 if and only if $Q_{ij} = 0$ for every j) and $\boldsymbol{e}^{\intercal}$ is a properly sized unit vector. Thus, the final matrix $\mathbf{H}$ becomes a stochastic matrix. Similarly, the modified adjusted cosine similarity matrix is defined by:

$$\mathbf{S} \triangleq \hat{\mathbf{G}} + \frac{1}{m}\mathbf{a}'\mathbf{e}^{\intercal} \tag{2}$$

where matrix $\hat{\mathbf{G}}$ is the row normalized version of matrix $\mathbf{G}$, whose zero rows are indicated by $\mathbf{a}'$, and its ij^{th} element is formally defined by:

$$G_{ij} \triangleq 1 + \frac{\sum_{u_k \in \mathcal{U}} (r_{ki} - \overline{r}_{u_k})(r_{kj} - \overline{r}_{u_k})}{\sqrt{\sum_{u_k \in \mathcal{U}} (r_{ki} - \overline{r}_{u_k})^2} \sqrt{\sum_{u_k \in \mathcal{U}} (r_{kj} - \overline{r}_{u_k})^2}} \tag{3}$$

Here, $\overline{r}_{u_k}$ is the average of the ratings of user u_k, and r_{kj} is the rating assigned by user u_k to the item v_j (see [12]). The resulting direct proximity matrix is:

$$\mathbf{C} \triangleq \phi \mathbf{S} + (1 - \phi) \mathbf{H}$$

where $\phi < 1$ is a free parameter.

Hierarchical Proximity Matrix D: This matrix is created to depict the indirect connections between the elements of the item space that arise from its innate hierarchical structure. The existence of such connections is rooted in the idea that a user's rating, *except for expressing his direct opinion about a particular item, also gives a clue about his preferences regarding related elements of the item space*. For example, if Alice gives 5 stars to a specific comedy/romantic movie, except for testifying her opinion about that movie, also "hints" about her opinion regarding, firstly, comedy/romantic movies and, secondly, comedies and romantic movies in general. In the presence of sparsity the assistance of these indirect relations could be extremely helpful, as we sill see in Section 3.

Following this line of thought, we associate each row of matrix $\mathbf{D}$ with a probability vector $\mathbf{d}_v$, that distributes evenly its mass between the N_v different sets of $\mathcal{D}$, comprising $\mathcal{G}_v$, and then, uniformly to the included items of every such set. Formally, the ij^{th} element of matrix $\mathbf{D}$, that relates item v_i with item v_j, is defined as:

$$D_{ij} \triangleq \sum_{\mathcal{D}_k \in \mathcal{G}_{v_i}, v_j \in \mathcal{D}_k} \frac{1}{N_{v_i} |\mathcal{D}_k|} \tag{4}$$

Example 1. To clarify the definition of matrix $\mathbf{D}$, we give the following example:

$$\begin{array}{c|ccc|c|c}
 & \mathcal{D}_1 & \mathcal{D}_2 & \mathcal{D}_3 & \mathcal{N}_v & \mathcal{G}_v \\
\hline
v_1 & \checkmark & - & - & 1 & \{v_1, v_2, v_4, v_6\} \\
v_2 & \checkmark & \checkmark & - & 2 & \{v_1, v_2, v_4, v_5, v_6\} \\
v_3 & - & - & \checkmark & 1 & \{v_3, v_6\} \\
v_4 & \checkmark & \checkmark & - & 2 & \{v_1, v_2, v_4, v_5, v_6\} \\
v_5 & - & \checkmark & - & 1 & \{v_2, v_4, v_5\} \\
v_6 & \checkmark & - & \checkmark & 2 & \{v_1, v_2, v_3, v_4, v_6\}
\end{array}$$

The item space $\mathcal{V}$ consists of 6 movies, that belong to 3 Genres. In the last column we have computed the proximal sets. The corresponding matrix $\mathbf{D}$ is presented in Figure below:

2.2 The Hierarchical Itemspace Rank Algorithm

We are now ready to define the personalized ranking vector of HIR as the probability distribution over the itemspace produced by the algorithm presented below:

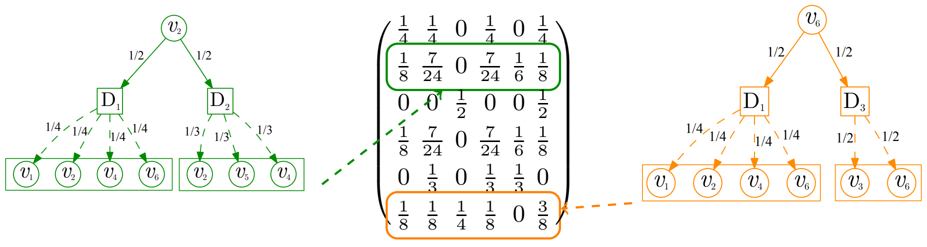
v_2
1/2
1/2
D1
D2
1/4
1/4
1/4
1/4
1/3
1/3
1/3
v_1 v_2 v_4 v_6
v_2 v_5 v_4
$\frac{1}{4}$ $\frac{1}{4}$ 0 $\frac{1}{4}$ 0 $\frac{1}{4}$
$\frac{1}{8}$ $\frac{7}{24}$ 0 $\frac{7}{24}$ $\frac{1}{6}$ $\frac{1}{8}$
0 0 $\frac{1}{2}$ 0 0 $\frac{1}{2}$
$\frac{1}{8}$ $\frac{7}{24}$ 0 $\frac{7}{24}$ $\frac{1}{6}$ $\frac{1}{8}$
0 $\frac{1}{3}$ 0 $\frac{1}{3}$ $\frac{1}{3}$ 0
$\frac{1}{8}$ $\frac{1}{8}$ $\frac{1}{4}$ $\frac{1}{8}$ 0 $\frac{3}{8}$
v_6
1/2
1/2
D1
D3
1/4
1/4
1/4
1/4
1/2
1/2
v_1 v_2 v_4 v_6
v_3 v_6

Fig. 1: We see the matrix $\mathbf{D}$ that corresponds to example 1, as well as a detailed computation of $\mathbf{d}_{v_2}$ and $\mathbf{d}_{v_6}$.

Algorithm 1 Hierarchical Itemspace Rank (HIR)

Input: Matrices $\mathbf{C}, \mathbf{D} \in \mathfrak{R}^{m\times m}$, parameters $\alpha, \beta > 0$ such that $\alpha + \beta < 1$, and the personalized preference vector $\boldsymbol{\omega} \in \mathfrak{R}^m$
Output: The ranking vector for the user, $\boldsymbol{\pi} \in \mathfrak{R}^m$

```
1: $\boldsymbol{\pi}^\intercal \leftarrow (1-\alpha-\beta)\boldsymbol{\omega}^\intercal$
2: for all $\omega_j \neq 0$ do
3:     $\boldsymbol{\pi}^\intercal \leftarrow \boldsymbol{\pi}^\intercal + \omega_j(\alpha\mathbf{c}_j^\intercal + \beta\mathbf{d}_j^\intercal)$     ▷ where $\mathbf{c}_j^\intercal, \mathbf{d}_j^\intercal$ the $j^{\text{th}}$ row of matrices $\mathbf{C}, \mathbf{D}$
4: end for
5: return $\boldsymbol{\pi}$
```

Theorem 1. *For every preference vector $\boldsymbol{\omega}$, the personalized vector $\boldsymbol{\pi}$ produced by the HIR algorithm denotes a distribution vector.*

Proof. Clearly, $\boldsymbol{\pi}$ is a non-negative vector. Thus, it suffices to show that $\boldsymbol{\pi}^\intercal\mathbf{e} = 1$.

$$\begin{aligned}\boldsymbol{\pi}^\intercal\mathbf{e} &= \left((1-\alpha-\beta)\boldsymbol{\omega}^\intercal + \alpha\sum_{j:\omega_j\neq 0}\omega_j\mathbf{c}_j^\intercal + \beta\sum_{j:\omega_j\neq 0}\omega_j\mathbf{d}_j^\intercal\right)\mathbf{e}\\ &= (1-\alpha-\beta)\boldsymbol{\omega}^\intercal\mathbf{e} + \alpha\sum_{j:\omega_j\neq 0}\omega_j\mathbf{c}_j^\intercal\mathbf{e} + \beta\sum_{j:\omega_j\neq 0}\omega_j\mathbf{d}_j^\intercal\mathbf{e}\end{aligned}$$

But since the elements of the preference vector $\boldsymbol{\omega}^\intercal$ are by definition normalized to sum to 1, and matrices $\mathbf{C}$ and $\mathbf{D}$ are row stochastic, we get:

$$\boldsymbol{\pi}^\intercal\mathbf{e} = (1-\alpha-\beta) + \alpha\sum_{j:\omega_j\neq 0}\omega_j + \beta\sum_{j:\omega_j\neq 0}\omega_j = (1-\alpha-\beta) + \alpha + \beta = 1$$

and the proof is complete. □

Computational and Storage Needs HIR needs to store the direct and the hierarchical proximity matrices. Matrix $\mathbf{C}$ is innately sparse and scales very well with the increase of the number of users; the addition of a new user to the system could result only in an increase of the number of nonzero elements of $\mathbf{C}$, since the dimension of the matrix depends solely on the cardinality of the item space which in most real applications increases slowly. In case of matrix $\mathbf{D}$, from the definition of the family of sets $\mathcal{D}$, it becomes intuitively obvious that whenever $K < m$, matrix $\mathbf{D}$ is a low-rank matrix. Furthermore, a closer look at the definition 4 above, suggests a very useful factorization of matrix $\mathbf{D}$, that can be exploited in order to achieve efficient storage. In particular, if we define an "aggregation" matrix $\mathbf{A} \in \Re^{m \times K}$, whose ik^{th} element is 1, when $v_i \in \mathcal{D}_k$ and zero otherwise, and letting $\mathbf{X}$ and $\mathbf{Y}$ denote the row-normalized versions of $\mathbf{A}$ and $\mathbf{A}^\intercal$ respectively, we observe that matrix $\mathbf{D}$ can be expressed as $\mathbf{D} = \mathbf{XY}$. Now, if we take into account the fact that for any reasonable decomposition of the item space, $K \ll m$ holds, the storage needs for the hierarchical proximity matrix become very small; we just have to store 2 sparse and thin matrices instead of the dense square matrix $\mathbf{D}$. This allows in realistic scenarios the introduction of more than one decompositions, according to different criteria, that could lead to better recommendations. From a computational point of view, we see that step 3 of our algorithm involves $O(|\mathcal{V}|)$ operations and it is executed $|\mathcal{L}_i|$ times. Typically, $|\mathcal{L}_i| \ll m$ since users interact with only a very small fraction of the available items, so the resulting burden is small.

3 Experimental Evaluation

To evaluate HIR, we apply it to the classic movie recommendation problem, employing a number of widely used performance indices. Our experiments were done using the publicly available `MovieLens` dataset.

MovieLens Dataset The `MovieLens` dataset is a standard benchmark for recommender system techniques, containing 100,000 ratings from 943 users for 1,682 movies. Every user has rated 20 or more movies, in order to achieve a greater reliability for user profiling. Rating scores are integers between 1 and 5. The dataset comes with 5 predefined splittings, each with 80% of ratings as training set and 20% as test set (as described in [12]). `MovieLens` dataset also comes with information that relates the included movies to genres. For the experimental evaluation of our method, we use this as the simple criterion of decomposition behind the definition of matrix $\mathbf{D}$. Of course, in realistic situations, when more information about the data is available, there can be more than one decomposition of the underlying space according to different criteria.

To simplify the definition of the metrics presented below, we define $\overline{\mathcal{W}}_i \triangleq \overline{(\mathcal{L}_i \cup \mathcal{T}_i)}$, which denotes a set of movies that are neither in the training set nor in the test set of user u_i (i.e. these movies have not been watched by the user). Unless stated otherwise, the values of the free parameters used for the following experiments are: $\alpha = 0.8, \beta = 0.1, \phi = 0.7$.

3.1 Experiments

Degree of agreement The first performance index we used is the *degree of agreement* (DOA); a measure commonly used in the literature to evaluate the quality of ranking-based recommendation methods [5,6,7,13]. DOA is a variant of the Somers'D statistic, defined as follows:

$$\mathrm{DOA}_i \triangleq \frac{\sum_{v_j \in \mathcal{T}_i \wedge v_k \in \overline{\mathcal{W}}_i} [\pi^i(v_j) > \pi^i(v_k)]}{\mid \mathcal{T}_i \mid * \mid \overline{\mathcal{W}}_i \mid}$$

where $\pi^i(v_j)$ is the ranking score of the movie v_j in user's u_i ranking list, and $[S]$ equals 1, if statement S is true and zero otherwise. Then, macro-averaged DOA (macro-DOA) is the average of all DOA_i and micro-averaged DOA (micro-DOA) is the ratio between the aggregate number of movie pairs in the correct order and the total number of movie pairs checked (for details, see [5,6]).

Table 1: Average performance and standard deviation between HIR and other state-of-the-art recommendation algorithms ([4,6,5,7,13]), computed over the same standard predefined folds. The metrics used for the comparison are the micro-DOA and macro-DOA.

micro-DOA		macro-DOA			
DN	81.33 ± 0.43	DN	80.51 ± 1.23	MRF	89.47 ± 0.44
HRF	88.07 ± 0.59	HRF	89.83 ± 0.52	NB	88.87 ± 0.22
IR	87.06 ± 0.10	IR	87.76 ± 0.27	**HIR**	$\mathbf{89.99 \pm 0.20}$
HIR	$\mathbf{88.85 \pm 0.29}$	Katz	85.83 ± 0.24	CT	84.09 ± 0.01
MRF	88.09 ± 0.50	$L^\dagger$	87.23 ± 0.84	PCA CT	84.04 ± 0.76
NB	86.66 ± 0.30	MaxF	84.07 ± 0.00	TR	89.08 ± 0.11

In Table 1, we present the micro-DOA and macro-DOA values measured by 5-fold cross-validation. The test employs the publicly available predefined partitioning of the dataset into five pairs of training and test sets, which allows easy comparisons with the different results to be found in the literature. All the DOA scores included in this table refer to the same predefined splitting. We see that HIR outperforms all other state-of-the-art techniques considered, by obtaining a macro-DOA value of **89.99** which is about 6.8% greater than the baseline (MaxF). The same is true for the micro-DOA measure, where HIR achieves an **88.85** value opposite to 88.09 of MRF and 88.07 of HRF having at the same time better standard deviation.

Sparsity In order to demonstrate the merits of our method in dealing with the problems caused by the low density of the item space, we conduct the following experiment. For each of the 5 predefined splittings, we simulate the phenomenon of sparseness by randomly selecting to include 80%, 60%, and 40% of the ratings on a new artificially sparsified version of the dataset, which we then use to test the quality decay caused by sparseness. To isolate the positive effect of the exploitation of hierarchical proximity and the related matrix $\mathbf{D}$, we run HIR against two algorithms related to its basic sub-components (namely the ItemRank and the SimRank[3] algorithms) and we evaluate their performance running the standard degree of agreement tests.

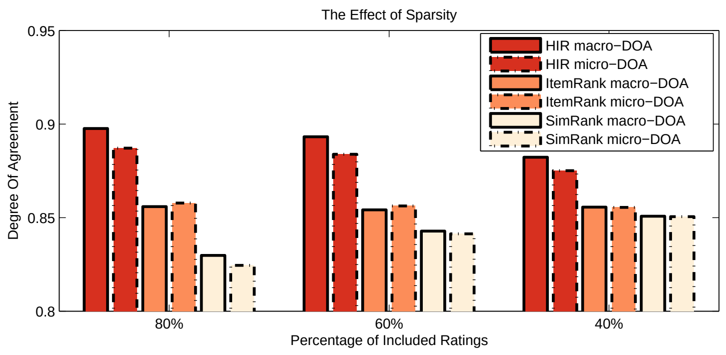

Fig. 2: The effect of sparsity on the quality of recommendations for artificially sparsified versions of the `MovieLens` dataset.

In Figure 2, we clearly see that HIR performs better than the other algorithms in both micro- and macro-DOA metrics, exhibiting very good results even when only 40% of the ratings are available. These results verify the intuition behind HIR; even though the direct item-item relations of the dataset collapse with the exclusion of such many ratings, the indirect relations captured by matrix $\mathbf{D}$ decay harder and thus, preserve longer the coarser structure of the data. This results in a recommendation ranking vector that proves to be less sensitive to the sparsity of the underlying space.

Localized Sparsity One very common and interesting manifestation of the sparsity problem is through the introduction of new items to the system. Naturally, because these items are new, they have been rated fewer times and thus, their relation with the rest of the elements of the item space is not yet clear. This, in many cases, could result in unfair treatment.

To evaluate the performance of our algorithm in coping with the newly added items bias problem, we run the following experiment. We randomly select a number of movies having 30 ratings or more, and we randomly delete 90% of their ratings. The idea is that the modified data represent an "earlier version" of

[3] SimRank is a simple variant of ItemRank that correlates the items using the adjusted cosine-based similarity matrix defined by relation 2, instead of the correlation matrix defined by relation 1.

the dataset, when these movies were new to the system, and as such, had fewer ratings.

We run several instances of HIR for different values of β, varying from $\beta = 0$ (where matrix $\mathbf{D}$ is not included and the ranking is induced by the weighted combination of the direct sub-components) to $\beta = 0.4$, keeping the sum $\alpha + \beta = 0.9$. Then, we compare the rankings induced on the modified data with their corresponding original rankings. The measure used for this comparison is Kendall's τ correlation coefficient. High value of this metric suggests that the two ranking lists are "close", which means that the newly added movies are more likely to receive treatment similar to their original one. Finally, to have a measure of the quality of the original list for each HIR instance, we also run the DOA tests and we present the results in Table 2.

Table 2: HIR performance in dealing with the localized sparsity problem.

# New Movies	$\alpha = 0.9$ $\beta = 0$	$\alpha = 0.85$ $\beta = 0.05$	$\alpha = 0.8$ $\beta = 0.1$	$\alpha = 0.7$ $\beta = 0.2$	$\alpha = 0.6$ $\beta = 0.3$	$\alpha = 0.5$ $\beta = 0.4$
100	0.8736	0.8776	0.8812	0.8878	0.8945	0.9020
200	0.7814	0.7886	0.7949	0.8066	0.8186	0.8317
300	0.6843	0.6940	0.7025	0.7190	0.7369	0.7572
macro-DOA	89.63 ± 0.18	89.91 ± 0.20	$\mathbf{89.99 \pm 0.20}$	89.58 ± 0.22	88.45 ± 0.23	86.62 ± 0.25
micro-DOA	88.47 ± 0.28	88.75 ± 0.29	$\mathbf{88.85 \pm 0.29}$	88.50 ± 0.29	87.42 ± 0.29	85.63 ± 0.29

The experiment shows that the introduction of even a very small β induces a positive effect against localized sparsity. This is in accordance with the way HIR views the problem. In our method, the ranking score of the items is not exclusively determined by their ratings alone; their proximal sets also matter, because they "sketch" the relations of these newly added items to the other elements of the item space. Thus, even though they lack sufficient number of ratings, they are treated more fairly.

As expected, the value of Kendall's τ increases with β for all the parametric range tested. However, when the value of β becomes 0.2 or larger, the quality of the original recommendations begins to drop, as the direct inter-item relations get increasingly ignored. Intuitively, the proper selection of parameters is expected to vary with the sparsity of the underlying space. Our experiments with the `MovieLens` dataset suggest that a choice of β between 0.1 and 0.15 and a choice of α between 0.8 and 0.75, give very good results in both quality of recommendations and sparseness insensitivity.

4 Conclusions and Future Work

In this paper we proposed Hierarchical Itemspace Rank; a novel recommender method that exploits the innate hierarchical structure of the itemspace to provide an elegant and computationally efficient method for generating ranking based recommendations.

One very interesting research direction we are currently pursuing involves the effect of the granularity of the decompositions: intuitively, there seems to be a trade-off between the sparseness insensitivity, which is generally assisted by coarse grained decompositions, and the quality of recommendations, which seems to be supported by more detailed categorizations. Another interesting path that remains to be explored involves the introduction of more than one decompositions based on different criteria, and the effect it has to the various performance metrics. In this work we considered the single decomposition case. Our experiments suggest that HIR with the exploitation of the hierarchical proximity properties of the itemspace, produces recommendations of higher quality than the other state-of-the-art methods considered, and at the same time, helps alleviating commonly occurring sparsity related problems.

References

1. Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 143–152, New York, NY, USA, 2012. ACM.
2. Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.
3. Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.
4. Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, November 1997.
5. F. Fouss, A. Pirotte, J.M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):355–369, 2007.
6. Antonino Freno, Edmondo Trentin, and Marco Gori. Scalable pseudo-likelihood estimation in hybrid random fields. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 319–328, New York, NY, USA, 2009. ACM.
7. Marco Gori and Augusto Pucci. Itemrank: a random-walk based scoring algorithm for recommender engines. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2766–2771, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.
8. Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.
9. Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
10. Benjamin M Marlin and Richard S Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*, pages 5–12. ACM, 2009.
11. Athanasios N. Nikolakopoulos and John D. Garofalakis. NCDawareRank: a novel ranking method that exploits the decomposable structure of the web. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 143–152, New York, NY, USA, 2013. ACM.

12. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.
13. Liyan Zhang, Kai Zhang, and Chunping Li. A topical pagerank based algorithm for recommender systems. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, New York, NY, USA, 2008. ACM.