Hierarchical Itemspace Rank: Exploiting hierarchy to alleviate sparsity in ranking-based recommendation

Article *in* Neurocomputing · July 2015

DOI: 10.1016/j.neucom.2014.09.082

Athanasios N. Nikolakopoulos
University of Minnesota Twin Cities
14 PUBLICATIONS **72** CITATIONS
SEE PROFILE

Marianna Kouneli
University of Patras
3 PUBLICATIONS **27** CITATIONS
SEE PROFILE

Hierarchical Itemspace Rank: Exploiting Hierarchy to Alleviate Sparsity in Ranking-based Recommendation

Athanasios N. Nikolakopoulos[a,b], Marianna A. Kouneli[a], John D. Garofalakis[a,b]

[a]*Computer Engineering and Informatics Department, University of Patras, University of Patras, Panepistimiopoli, GR-26500, Greece*
[b]*Computer Technology Institute and Press "Diophantus", N.Kazantzaki, GR-26504, Greece*

Abstract

Sparsity is an innate characteristic of recommender system databases and it is known to present one of the most challenging difficulties collaborative filtering methods have to deal with. In this paper, we propose *Hierarchical Itemspace Rank* (HIR); a novel recommendation algorithm that exploits the intrinsic hierarchical structure of the itemspace to tackle this problem, and to alleviate the related limitations it imposes to the quality of recommendation. A comprehensive set of experiments on the `MovieLens100K`, the `MovieLens1M` and the `Yahoo!R2Music` datasets indicates that our method is very effective in handling sparsity, even in its most extreme manifestation – the *cold-start problem*. Our tests show that HIR outperforms several state-of-the-art recommendation algorithms in widely used metrics, having at the same time the advantage of being computationally efficient and easily implementable.

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

1. Introduction

Recommender Systems (RS) are information filtering systems designed to help online users "clear the fog" of information overload. Given a set of users, a set of items and implicit or explicit ratings that express how much a user likes or dislikes the items he has already seen, recommender systems try to either predict the ratings of the unseen user-item pairs, or provide a list of items that the user might find preferable. Out of several different approaches to building RS, Collaborative Filtering (CF) is widely regarded as one of the most successful ones. 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 being proposed [1, 2, 3].

In most of these algorithms, the recommendation task reduces to *predicting* the ratings for all the unseen user-item pairs, using the Root Mean Squared Error (RMSE) between the predicted and actual ratings as the evaluation metric [2, 4]. Latent factor models, and in particular matrix factorization techniques, were shown to be particularly effective for this problem [5, 6].

Email addresses: `nikolako@ceid.upatras.gr` (Athanasios N. Nikolakopoulos), `kounelim@ceid.upatras.gr` (Marianna A. Kouneli), `garofala@ceid.upatras.gr` (John D. Garofalakis)

Recently, however, many leading researchers have pointed out that the use of RMSE criteria to evaluate recommender systems is not an adequate performance index [7, 8, 9]; they showed that even sophisticated methods, such as Asymmetric SVD and SVD++, trained to perform extremely well on RMSE, do not behave particularly well on the – much more common in practice – top-N recommendation task [8]. These observations have turned significant research attention to *ranking-based* techniques which are believed to conform more naturally with how the recommender system will actually be used in practice [7, 8, 10, 11, 12, 13].

Freno et al. [14] proposed a Hybrid Random Fields model [15] which they applied, together with a number of well known probabilistic graphical models, including Dependency Networks, Markov Random Fields and Naive Bayes [16], to predict top-N items for users. Shi et al. [17] proposed a ranking-based recommendation method that learns a latent factor model by directly maximizing a smoothed version of the mean reciprocal rank metric. Recently, Cremonesi et al. [8] proposed PureSVD; an algorithm that uses the truncated singular value decomposition to approximate the ratings matrix in order to produce recommendation vectors for the users. Extensive experiments conducted by the authors show that PureSVD outperforms several state-of-the-art matrix factorization techniques as well as standard neighborhood models in the top-N recommendation task.

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 makes many 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. The latter is commonly referred to as the *cold-start* problem and is known to be responsible for significant degradation of CF performance [18, 19, 13].

Among the most promising approaches in dealing with sparsity are graph-based methods [2, 11, 20, 21]. The methods of this family exploit transitive relations in the data, which makes them able to estimate relationships between users and items that are not directly connected. Fouss et al. [10, 11] create a graph model of the RS database and they present a number of methods to compute node similarity measures, including the random walk-related average Commute Time and average First Passage Time, as well as the pseudo-inverse of the Laplacian matrix. They compare their methods against other state-of-the-art graph-based approaches such as, the sophisticated node similarity measure that integrates indirect paths in the graph, based on the matrix-forest theorem [22], and a similarity measure based on the well known Katz algorithm [23]. Gori and Pucci [21] proposed ItemRank; a PageRank-inspired scoring algorithm that produces a personalized ranking vector using a random walk with restarts on an items' correlation graph induced by the ratings.

In this work[1], based on the intuition behind a recently proposed Web ranking framework [25], we try to exploit the innately hierarchical nature of the underlying spaces to characterize inter-item relations in a macroscopic level. We decompose the itemspace to define blocks of closely related elements, and we use this decomposition to exploit the indirect proximity properties hidden in the structure of the itemspace. Central to our approach is the idea that blending

[1]This paper presents an extended version of the results published in [24].

together the direct as well as the indirect components can refine the inter-item relations, reduce the sensitivity to sparseness, and improve the quality of recommendations. Having this in mind, we develop **Hierarchical Itemspace Rank** (HIR); a novel recommender framework that brings together the above components in a generic and mathematically attractive way.

After describing formally the core components of our model, we proceed to the rigorous mathematical definition of the involved matrices (Section 2.2), we present HIR algorithm (Section 2.3) and we discuss its computational and storage needs (Section 2.4). In order to evaluate experimentally its quality in realistic scenarios, we apply HIR on two different recommendation domains; the classic *movie recommendation problem* using the standard `MovieLens1M`[2] dataset and the *music recommendation problem* using the `Yahoo!R2Music` dataset[3] (Section 3). We test the performance of our method in dealing with the problems caused by the low density of the underlying space, by conducting a number of experiments that simulate the sparsity phenomenon (Section 3.2), and we confirm that HIR displays great insensitivity, even when sparsity is severe (*new community* problem). The same is true in the case when sparsity is localized; a problem commonly occurring in recommender systems in operation because of the frequent introduction of *new users* and *new items* to the database. For completeness, we also run HIR on the standard evaluation benchmark dataset `MovieLens100K` (Section 3.3), using the publicly available predefined splittings that allow direct comparisons to the many different results to be found in the literature. Finally, we outline directions for future work (Section 4) and we conclude this work (Section 5).

2. The HIR Framework

2.1. Notation

Throughout this paper, all vectors are represented by bold lowercase letters and they are column vectors (e.g., $\boldsymbol{\omega}$). All matrices are represented by bold upper case letters (e.g., $\mathbf{C}$). The j^{th} row of matrix $\mathbf{C}$ is denoted $\mathbf{c}_j^\intercal$. The ij^{th} element of matrix $\mathbf{C}$ is denoted C_{ij}. We use calligraphic letters to denote sets (e.g., $\mathcal{U}, \mathcal{V}$). Finally, symbol $\triangleq$ is used in definition statements.

2.2. Model Definition

Let $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$ be a set of *users* and $\mathcal{V} = \{v_1, v_2, \dots, v_m\}$ a set of *items*. Let $\mathcal{R}$ be a set of tuples $t_{ij} = (u_i, v_j, r_{ij})$, where r_{ij} can either be a positive number referred to as the *rating* given by user u_i to the item v_j, or simply a 0-1 valued variable (binary ratings). 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:

$$\mathcal{L}_i \triangleq \{v_k : t_{ik} \in \mathcal{L}\} \text{ and } \mathcal{T}_i \triangleq \{v_l : t_{il} \in \mathcal{T}\}. \tag{1}$$

Each user u_i, for whom $\mathcal{L}_i \neq \emptyset$ holds, is associated with a vector

$$\boldsymbol{\omega}^i \triangleq [\omega_1^i, \omega_2^i, \dots, \omega_m^i], \tag{2}$$

[2]http://www.movielens.umn.edu .
[3]http://webscope.sandbox.yahoo.com/ .

whose non-zero 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 non-empty sets

$$\mathcal{D} \triangleq \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}, \tag{3}$$

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 $\mathcal{G}_v$. Formally, the set $\mathcal{G}_v$ is given by:

$$\mathcal{G}_v \triangleq \bigcup_{v \in \mathcal{D}_k} \mathcal{D}_k \tag{4}$$

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 the *Direct Association Matrix* **C**, and the *Hierarchical Proximity Matrix* **D**, that bring together the direct and the hierarchical components of the underlying space, to enable our method to map each user's preference vector to a personalized distribution over the itemspace.

2.2.1. Direct Association Matrix C

The direct association matrix **C** is defined to capture the direct relations between the elements of $\mathcal{V}$. Generally, every such element will be associated with a discrete distribution $\mathbf{c}_v^\intercal = [c_1, c_2, \cdots, c_m]$ over $\mathcal{V}$, that reflects the similarities between these elements. In our case, we use the stochastic matrix that corresponds to the correlation graph proposed in [21]. Matrix **C** is formally defined bellow.

We first need to define a matrix **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.

$$\mathcal{U}_{ij} \triangleq \begin{cases} \{u_k : (v_i \in \mathcal{L}_k) \wedge (v_j \in \mathcal{L}_k)\} & \text{for } i \neq j \\ \emptyset & \text{otherwise} \end{cases} \tag{5}$$

Then, if we use $\hat{\mathbf{Q}}$ to denote the row normalized version of **Q**, i.e. the matrix where every non-zero row sums up to 1, the resulting matrix will be defined as follows:

$$\mathbf{C} \triangleq \hat{\mathbf{Q}} + \frac{1}{m}\mathbf{a}\mathbf{e}^\intercal \tag{6}$$

where **a** is a vector that indicates the zero rows of matrix **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.

Remark 1. The $\frac{1}{m}\mathbf{a}\mathbf{e}^\intercal$ term in the definition of matrix **C**, is used to replace its $\mathbf{0}^\intercal$ rows with the uniform distribution over all items, thereby transforming **C** to a stochastic matrix. Note that this "stochasticity adjustment" is often needed – especially in case of cold start recommendation scenarios, when it is more likely to find pairs of movies that have been rated by disjoint sets of users.

2.2.2. Hierarchical Proximity Matrix D

This matrix is created to depict the indirect connections between the elements of the itemspace 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 itemspace*.

For example, if Alice gives 5 stars to a specific comedy/romantic movie, except for testifying her opinion about that particular 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 will see in Section 3.

Following this line of thought, we associate each row of matrix $\mathbf{D}$ with a probability vector $\mathbf{d}_v^\intercal$, 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:

$$D_{ij} \triangleq \sum_{\mathcal{D}_k \in \mathcal{G}_{v_i}, v_j \in \mathcal{D}_k} \frac{1}{N_{v_i} |\mathcal{D}_k|} \tag{7}$$

Notice that Equation 7 together with the definition of the family of sets $\mathcal{D}$, ensure that the resulting matrix $\mathbf{D}$ is row stochastic.

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

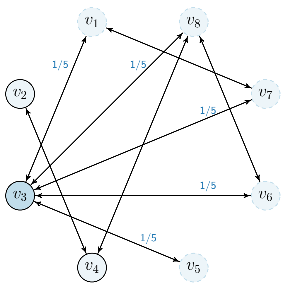
Consider a simple Movie Recommendation System consisting of an itemspace of 8 movies and a userspace of 10 users each having rated at least one movie. Let the set of ratings, $\mathcal{R}$, be the one presented below:

$$\begin{array}{lll}
t_{11} = (u_1, v_1, 3) & t_{38} = (u_3, v_8, 5) & t_{76} = (u_7, v_6, 1) \\
t_{13} = (u_1, v_3, 4) & t_{44} = (u_4, v_4, 5) & t_{78} = (u_7, v_8, 2) \\
t_{17} = (u_1, v_7, 2) & t_{52} = (u_5, v_2, 1) & t_{83} = (u_8, v_3, 3) \\
t_{21} = (u_2, v_1, 5) & t_{54} = (u_5, v_4, 3) & t_{93} = (u_9, v_3, 2) \\
t_{27} = (u_2, v_7, 3) & t_{66} = (u_6, v_6, 3) & t_{103} = (u_{10}, v_3, 3) \\
t_{34} = (u_3, v_4, 3) & t_{73} = (u_7, v_3, 3) & t_{105} = (u_{10}, v_5, 4)
\end{array}$$

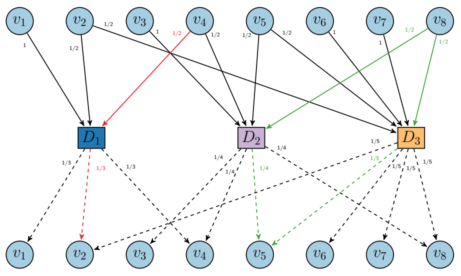
Assume also that the 8 movies of the itemspace belong to 3 genres as seen below:

$$\begin{array}{c|ccc||c|c}
 & \mathcal{D}_1 & \mathcal{D}_2 & \mathcal{D}_3 & N_v & \mathcal{G}_v \\
v_1 & \checkmark & - & - & 1 & \{v_1, v_2, v_4\} \\
v_2 & \checkmark & - & \checkmark & 2 & \{v_1, v_2, v_4, v_5, v_6, v_7, v_8\} \\
v_3 & - & \checkmark & - & 1 & \{v_3, v_4, v_5, v_8\} \\
v_4 & \checkmark & \checkmark & - & 2 & \{v_1, v_2, v_3, v_4, v_5, v_8\} \\
v_5 & - & \checkmark & \checkmark & 2 & \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\} \\
v_6 & - & - & \checkmark & 1 & \{v_2, v_5, v_6, v_7, v_8\} \\
v_7 & - & - & \checkmark & 1 & \{v_2, v_5, v_6, v_7, v_8\} \\
v_8 & - & \checkmark & \checkmark & 2 & \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}
\end{array}$$

In Figure 1, we present the direct association matrix $\mathbf{C}$ and the hierarchical proximity matrix $\mathbf{D}$ that correspond to our simple recommender system. In the first row we see the correlation



$$\Longrightarrow \quad \mathbf{C} = \begin{pmatrix} 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & 0 & 0 & 0 & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{2}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \end{pmatrix}$$



$$\Longrightarrow \quad \mathbf{D} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{4}{15} & 0 & \frac{1}{6} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{8} & \frac{7}{24} & \frac{1}{8} & 0 & 0 & \frac{1}{8} \\ 0 & \frac{1}{10} & \frac{1}{8} & \frac{1}{8} & \frac{9}{40} & \frac{1}{10} & \frac{1}{10} & \frac{9}{40} \\ 0 & \frac{1}{5} & 0 & 0 & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & \frac{1}{5} & 0 & 0 & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & \frac{1}{10} & \frac{1}{8} & \frac{1}{8} & \frac{9}{40} & \frac{1}{10} & \frac{1}{10} & \frac{9}{40} \end{pmatrix}$$

Figure 1: We see the matrices $\mathbf{C}, \mathbf{D}$ that correspond to Example 1. In the first row we highlight with blue color the computation of the $\boldsymbol{c}_{\boldsymbol{v}_3}$. In the second row we highlight with red and green color the computation of the D_{42} and D_{85}, respectively.

graph and the related matrix $\mathbf{C}$. In the second row we see a graph presenting the detailed computation of the hierarchical proximity matrix $\mathbf{D}$.

2.3. 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:

Algorithm 1 Hierarchical Itemspace Rank (HIR)

Input: Matrices $\mathbf{C}, \mathbf{D} \in \Re^{m \times m}$, parameters $\alpha, \beta > 0$ such that $\alpha + \beta < 1$, and the personalized preference vector $\boldsymbol{\omega} \in \Re^m$

Output: The ranking vector for the user, $\boldsymbol{\pi} \in \Re^m$

```
1: $\boldsymbol{\pi}^\intercal \leftarrow (1 - \alpha - \beta)\boldsymbol{\omega}^\intercal$
2: for all $\omega_j \neq 0$ do
3:     $\boldsymbol{\pi}^\intercal \leftarrow \boldsymbol{\pi}^\intercal + \omega_j(\alpha \mathbf{c}_j^\intercal + \beta \mathbf{d}_j^\intercal)$
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 is a well-defined normalized recommendation vector that denotes a probability distribution over the item set $\mathcal{V}$.*

PROOF. Clearly, for every preference vector $\boldsymbol{\omega}$, and for $\alpha, \beta > 0$ such that $\alpha + \beta < 1$ holds, $\boldsymbol{\pi}$ is a non-negative vector. Thus, it suffices to show that $\boldsymbol{\pi}^\intercal \mathbf{e} = 1$. We have:

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

However, since the elements of the preference vector $\boldsymbol{\omega}$ are by definition normalized to sum to one[4], and matrices $\mathbf{C}$ and $\mathbf{D}$ are row stochastic, we get:

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

and the proof is complete. □

2.4. Storage Aspects

Direct Association Matrix 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 non-zero elements of $\mathbf{C}$, since the dimension of the matrix depends solely on the cardinality of the itemspace, which in most real applications increases slowly [21].

Hierarchical Proximity Matrix 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 its definition, suggests a very useful factorization of matrix $\mathbf{D}$, that can be exploited in order to achieve efficient storage and computability.

In particular, let us define an "aggregation" matrix $\mathbf{A} \in \Re^{m \times K}$, such that

$$A_{ik} \triangleq \begin{cases} 1 & \text{if } v_i \in \mathcal{D}_k \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

It is then easy to observe that matrix $\mathbf{D}$ can be written as the product of the row-stochastic versions of the aggregation matrix and its transpose respectively. Rigorously,

$$\mathbf{D} = \mathbf{XY}, \quad \mathbf{X} \in \Re^{m \times K}, \mathbf{Y} \in \Re^{K \times m} \tag{9}$$

with matrices $\mathbf{X}, \mathbf{Y}$ defined by

$$\mathbf{X} \triangleq \mathbf{S}^{-1}\mathbf{A} \tag{10}$$

and

$$\mathbf{Y} \triangleq \mathbf{T}^{-1}\mathbf{A}^\intercal \tag{11}$$

where $\mathbf{S} \triangleq \mathrm{diag}(\mathbf{Ae})$ and $\mathbf{T} \triangleq \mathrm{diag}(\mathbf{A}^\intercal \mathbf{e})$ are diagonal matrices.

[4]Note also that the definition of preference vector asserts that it contains at least one element greater than zero (see Section 2.2).

Notice here that the definition of $\mathcal{D}$ ensures the invertibility of matrices $\mathbf{S}$ and $\mathbf{T}$, thus matrices $\mathbf{X}, \mathbf{Y}$ are well defined irrespectively of any particular set $\mathcal{R}$ and decomposition $\mathcal{D}$. The result is given formally in the following lemma.

Lemma 1. *Matrices $\mathbf{X}$ and $\mathbf{Y}$ are well defined for any decomposition $\mathcal{D}$ satisfying the definition given in Section 2.2.*

Proof. It suffices to show that matrices $\mathbf{S}, \mathbf{T}$ are necessarily invertible for every possible matrix $\mathbf{A}$. Notice that the definition of $\mathcal{D}$ ensures that matrix $\mathbf{A}$ has the following properties:

(a) Every row of $\mathbf{A}$ denotes a non-zero vector in $\Re^K$.
(b) Every column of $\mathbf{A}$ denotes a non-zero vector in $\Re^m$.

Property (a) holds since the existence of a zero row in matrix $\mathbf{A}$ implies $\mathcal{V} \neq \bigcup_{k=1}^{K} \mathcal{D}_k$, which contradicts the definition of $\mathcal{D}$. Property (b) holds also since the sets comprising $\mathcal{D}$ are defined to be non-empty[5].

Now, notice that these two properties of $\mathbf{A}$ ensure that both $\mathbf{Ae}$ and $\mathbf{A}^\intercal\mathbf{e}$ denote vectors of strictly positive elements, which makes the diagonal matrices $\mathbf{S}, \mathbf{T}$ invertible, as needed.

2.5. *Computational Aspects*

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.

Furthermore, the factorization of matrix $\mathbf{D}$ into a product of two extremely sparse matrices and the innately low density of the ratings matrix, suggest a very efficient batch computation scheme of HIR recommendation vectors. In particular, if we define a matrix $\mathbf{\Omega} \in \Re^{n\times m}$ such that:

$$\mathbf{\Omega} \triangleq \begin{bmatrix} (\omega^1)^\intercal \\ (\omega^2)^\intercal \\ \vdots \\ (\omega^n)^\intercal \end{bmatrix} \tag{12}$$

we can compute the recommendation vectors for every user in the system without ever needing matrix $\mathbf{D}$ to be explicitly computed, allowing at the same time, the exploitation of the highly optimized sparse BLAS3 set of low-level kernel subroutines. The batch computation is presented below:

Algorithm 2 Batch HIR Computation

Input: Matrices $\mathbf{C}, \mathbf{X}, \mathbf{Y}$, parameters $\alpha, \beta > 0$ such that $\alpha+\beta < 1$, and the personalized preference matrix $\mathbf{\Omega}$
Output: The ranking matrix for users, $\mathbf{\Pi} \in \Re^{n\times m}$

```
1: $\mathbf{\Pi} \leftarrow (1-\alpha-\beta)\mathbf{\Omega}$
2: $\mathbf{\Pi} \leftarrow \mathbf{\Pi} + \alpha(\mathbf{\Omega C}) + \beta((\mathbf{\Omega X})\mathbf{Y})$
3: return $\mathbf{\Pi}$
```

The rows of matrix $\mathbf{\Pi} \in \Re^{n\times m}$ contain the recommendation vectors for every user in $\mathcal{U}$.

[5]Note that allowing the existence of empty sets would be meaningless from a modeling perspective.

3. Experimental Evaluation

To evaluate HIR, we apply it to two problems originating from different recommendation domains, namely

- *Movie Recommendation.* We test our method using the standard `MovieLens100K` and `MovieLens1M` datasets, that have been widely used for the performance evaluation of recommender systems. `MovieLens` datasets also come with information that relates the movies to genres; this was chosen as the criterion of decomposition for the definition of the hierarchical proximity matrix.

- *Music Recommendation.* To evaluate HIR on the music recommendation problem we make use of the `Yahoo!R2Music` dataset. This is a relatively new dataset representing a snapshot of the Yahoo!Music community's preferences for different songs. The dataset contains over 717 million ratings and for every song there exists useful meta-information about the artists, the albums and the music genre it belongs to. In our experiments we used the artists as the criterion for the decomposition behind the definition of matrix **D**.

More details about the datasets are presented in Table 1.

Table 1: The MovieLens and Yahoo! datasets used for the experiments.

Dataset	#Users	#Items	#Ratings
`MovieLens100K`	943	1,682	100,000
`MovieLens1M`	6,040	3,883	1,000,209
`Yahoo!R2Music`	1,823,179	136,736	717,872,016

3.1. Competing Recommendation Methods and Evaluation Metrics

We test HIR against seven state-of-the-art ranking-based recommendation methods: the node similarity algorithms $\mathbf{L}^{\dagger}$, and **Katz**; the random walk approaches **First Passage Time** (FP) and **Commute Time** (CT); the **Matrix Forest Algorithm** (MFA); and finally, the well known algorithms **ItemRank** (IR) and **PureSVD**[6]. Notice that all algorithms except ItemRank and PureSVD can also take advantage of the hierarchy of the itemspace, which – as we will see in the following sections – can provide significant help in cases of extreme sparsity.

3.1.1. Implementation of Competing Methods

All competing algorithms were implemented in Matlab. For the computation of $\mathbf{L}^{\dagger}$ we used the formula

$$\mathbf{L}^{\dagger} = (\mathbf{L} - \frac{1}{n+m+K}\mathbf{e}\mathbf{e}^{\intercal})^{-1} + \frac{1}{n+m+K}\mathbf{e}\mathbf{e}^{\intercal} \tag{13}$$

where **L** is the Laplacian of the graph model of the recommender system (see [11] for details), n, the number of users, m, the number of items, and K, the number of blocks. MFA similarity matrix is computed by

$$\mathbf{M} = (\mathbf{I} + \mathbf{L})^{-1} \tag{14}$$

[6]The choice of the number of latent factors for PureSVD was determined experimentally for each dataset. In the following sections we report the best results achieved.

and Katz similarity matrix is computed by

$$\mathbf{K} = \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \cdots + \alpha^n\mathbf{A}^n + \cdots = (\mathbf{I} - \alpha\mathbf{A})^{-1} - \mathbf{I} \tag{15}$$

where **A** is the adjacency matrix of the graph and α measures the attenuation in a link (see [10]). For the inversions we used the `inv` function of Matlab. The Average First Passage Time is computed by iteratively solving the recurrence

$$\begin{cases} \mathrm{FP}(k|k) & = & 0 \\ \mathrm{FP}(k|i) & = & 1 & + & \sum_{j=1}^{n+m+K} p_{ij}\,\mathrm{FP}(k|j), & \text{for } i \neq k \end{cases} \tag{16}$$

where p_{ij} is the conditional probability a random walker visits node j next, given that he is currently in node i. Having computed the Average First Passage times the Average Commute Time can be obtained by

$$\mathrm{CT}(i, j) = \mathrm{FP}(i|j) + \mathrm{FP}(j|i) \tag{17}$$

Finally, ItemRank was computed using the Power Method as suggested by the authors in [21], and PureSVD by applying Matlab's `svds` function on the Ratings matrix.

Note here that all random-walk approaches require to handle a graph of $(n + m + K)$ nodes and to compute $2nm$ first passage time scores. L†, Katz and MFA, involve the inversions of an $(n + m + K)$-dimensional square matrix. Similarly, PureSVD methods involves the computation of the truncated singular value decomposition of an $(n \times m)$ matrix; problems that easily become intractable as the number of users in the system grows. In fact, only HIR and ItemRank involve matrices whose dimensions depend solely on the cardinality of the itemspace, which in most realistic applications increases slowly. However, in our experiments, we observe that HIR runs at least 10-15 times faster than ItemRank[7]. This is true for every dataset we experimented on.

3.1.2. Metrics

To evaluate the performance of HIR, except for the standard **Spearman's $\boldsymbol{\rho}$** and **Kendall's $\boldsymbol{\tau}$** metrics [26, 27], we also use two other well known ranking measures, namely the **Degree of Agreement** (DOA) [10, 14, 21] and the **Normalized Distance-based Performance Measure** (NDPM) [27], outlined below. Table 2 contains all the necessary definitions.

Kendall's $\boldsymbol{\tau}$ is an intuitive nonparametric rank correlation index that has been widely used in the literature. The τ of ranking lists $\boldsymbol{r^i}, \boldsymbol{\pi^i}$ is defined to be:

$$\tau \triangleq \frac{C - D}{\sqrt{N - T_r}\sqrt{N - T_\pi}} \tag{18}$$

and takes the value of 1 for perfect match and -1 for reversed ordering.

Spearman's $\boldsymbol{\rho}$ is another widely used non-parametric measure of rank correlation. The ρ of ranking lists $\boldsymbol{r^i}, \boldsymbol{\pi^i}$ is defined to be:

$$\rho \triangleq \frac{1}{m}\frac{\sum_{v_j}(r^i_{v_j} - \bar{r}^i)(\pi^i_{v_j} - \bar{\pi}^i)}{\sigma(\boldsymbol{r^i})\sigma(\boldsymbol{\pi^i})} \tag{19}$$

[7]Choosing convergence tolerance 10^{-5}, damping factor 0.85 (as suggested by the authors [21]), and computing the recommendation vectors for all the users in batch for both methods, in order to exploit the highly optimized BLAS3 kernels.

Table 2: A summary of the notation used for the definition of the evaluation metrics.

Notation	Meaning
$\boldsymbol{r^i}$	User's u_i reference ranking
$\boldsymbol{\pi^i}$	RS generated ranking
$r^i_{v_j}$	Ranking score of the item v_j in user's u_i ranking list (reference ranking)
$\pi^i_{v_j}$	Ranking score of the item v_j in user's u_i ranking list (RS generated ranking)
C	Number of pairs that are concordant
D	Number of discordant pairs
N	Total number of pairs
T_r	Number of tied pairs in the reference ranking
T_π	Number of tied pairs in the system ranking
X	Number of pairs where the reference ranking does not tie, but the recommender system's ranking ties ($N - T_r - C - D$)
m	Total number of items

where the $\bar{\cdot}$ and $\sigma(\cdot)$ denote the mean and standard deviation. The ρ takes values from -1 to 1. A ρ of 1 indicates perfect rank association, a ρ of zero indicates no association between the ranking lists and a ρ of -1 indicate a perfect negative association of the rankings.

Normalized Distance-based Performance Measure The NDPM of ranking lists $\boldsymbol{r^i}$, $\boldsymbol{\pi^i}$ is defined to be:

$$\text{NDPM} \triangleq \frac{D + 0.5X}{N - T_r} \tag{20}$$

The NDPM measure gives a perfect score of 0 to RS that correctly predict every preference relation asserted by the reference. The worst score of 1 is assigned to recommendation vectors that contradict every preference relation in $\boldsymbol{r^i}$ [27, 28].

Degree of Agreement (DOA) is a performance index commonly used in the recommendation literature to evaluate the quality of ranking-based CF methods [10, 14, 21, 29]. DOA is a variant of the Somers' D statistic [30], defined as follows:

$$\text{DOA}_i \triangleq \frac{\sum_{v_j \in \mathcal{T}_i \wedge v_k \in \overline{\mathcal{W}_i}} [\pi^i_{v_j} > \pi^i_{v_k}]}{\mid \mathcal{T}_i \mid * \mid \overline{(\mathcal{L}_i \cup \mathcal{T}_i)} \mid} \tag{21}$$

where $[S]$ equals 1, if statement S is true and zero otherwise. 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 item pairs in the correct order and the total number of item pairs checked (for further details see [10, 14]).

In the following section we evaluate the performance of the algorithms on three repeatedly occurring manifestations of the sparsity problem, often referred to as the *Cold-Start Problems*. For this evaluation we use `Movielens1M` dataset as well as the larger and sparser `Yahoo!R2Music` dataset. Then, in Section 3.3, for completeness we also run HIR on the standard benchmark `MovieLens100K` in order to allow direct comparisons with the many different results that have been published in the literature.

3.2. Dealing with Sparsity

The cold-start problems refer to the difficulty of making reliable recommendations due to an initial lack of ratings. This is a very common situation faced by real recommender systems [3] and it usually manifests itself in three different cases:

(a) In the beginning stages of a system – *New Community* problem; the *de facto* small number of ratings results in a dataset with density too low for CF algorithms to uncover meaningful relations between items or users.

(b) With the addition of new users to the system – *New Users* problem; naturally, because these users are new, they have not rated many items and thus, the CF algorithm can not make reliable personalized recommendations yet.

(c) With the introduction of new items to the system's database – *New Items* problem; having little or no ratings at all, the relation of these items with the rest of the elements of the itemspace is not immediately clear.

3.2.1. New Community

In order to demonstrate the merits of our method in dealing with the new community problem, we conduct the following experiment. We simulate the phenomenon of sparseness by randomly selecting to include 10%, 20%, and 30% of the overall ratings on three new artificially sparsified versions of each dataset. The idea is that these modified versions represent early snapshots of the system's evolution, when the community was young and – as a result – the density of the dataset lower.

First, to show the positive effect of the exploitation of hierarchical proximity and the related matrix **D**, we run HIR using different values of β, varying from $\beta = 0$ to $\beta = 0.4$, keeping the sum $\alpha + \beta = 0.9$, and we test the quality of the recommendation list produced by each instance of HIR, using as reference ranking for each user his complete set of ratings from the original dataset. The measures used for this comparison is the Kendall's τ, the Spearman's ρ, the NDPM and the macro-DOA. Low value of the NDPM and high value of the other three metrics suggests that the ranking lists produced using the extremely sparse data are "close" to the preferences of the users, as described by their full set of ratings.

Table 3: HIR performance for the *New Community* problem for varying β on the `Yahoo!R2Music` and `MovieLens1M` datasets.

	`Yahoo!R2Music`							`MovieLens1M`						
	Dataset Density	$\alpha = 0.9$ $\beta = 0$	0.85 0.05	0.8 0.1	0.7 0.2	0.6 0.3	0.5 0.4	**Dataset Density**	$\alpha = 0.9$ $\beta = 0$	0.85 0.05	0.8 0.1	0.7 0.2	0.6 0.3	0.5 0.4
τ	0.33%	0.1295	0.1345	0.1354	**0.1356**	0.1353	0.1349	0.43%	0.1923	0.1886	0.1900	0.1916	**0.1919**	0.1905
	0.49%	0.1367	0.1407	0.1415	**0.1416**	0.1413	0.1408	0.85%	0.1989	0.1990	0.2001	0.2015	**0.2018**	0.2007
	0.66%	0.1430	0.1464	0.1472	**0.1474**	0.1471	0.1466	1.28%	0.2044	0.2050	0.2060	0.2073	**0.2077**	0.2070
NDPM	0.33%	0.1289	0.1165	**0.1148**	**0.1148**	0.1157	0.1167	0.43%	0.1269	0.1215	0.1187	0.1154	**0.1152**	0.1182
	0.49%	0.0937	0.0838	**0.0823**	0.0825	0.0837	0.0851	0.85%	0.0959	0.0924	0.0900	0.0873	**0.0870**	0.0895
	0.66%	0.0738	0.0653	**0.0637**	0.0638	0.0650	0.0665	1.28%	0.0826	0.0796	0.0775	0.0749	**0.0743**	0.0761
ρ	0.33%	0.1549	0.1609	0.1620	**0.1622**	0.1619	0.1614	0.43%	0.2303	0.2305	0.2321	0.2342	**0.2345**	0.2329
	0.49%	0.1672	0.1720	0.1730	**0.1732**	0.1727	0.1721	0.85%	0.2433	0.2449	0.2463	0.2481	**0.2484**	0.2472
	0.66%	0.1754	0.1795	0.1805	**0.1807**	0.1803	0.1797	1.28%	0.2507	0.2523	0.2536	0.2553	**0.2558**	0.2549
DOA	0.33%	0.8410	0.8568	**0.8590**	**0.8590**	0.8579	0.8566	0.43%	0.8626	0.8695	0.8726	0.8764	**0.8768**	0.8735
	0.49%	0.8689	0.8832	**0.8855**	0.8852	0.8836	0.8817	0.85%	0.8860	0.8904	0.8935	0.8970	**0.8974**	0.8944
	0.66%	0.8806	0.8948	**0.8974**	**0.8974**	0.8956	0.8933	1.28%	0.8891	0.8934	0.8964	0.9002	**0.9010**	0.8986

Table 3 reports the average values of each metric for every user that has at least one item rated in the modified dataset. The results show that the introduction of even a very small $\beta = 0.05$ induces a positive effect in dealing with the new community problem. The ranking quality continues to increase with β, reaching the best score when β takes a value between 0.1 and 0.3 for all different evaluation metrics considered. Then, the quality begins to drop, as expected, because the direct inter-item relations get increasingly ignored.

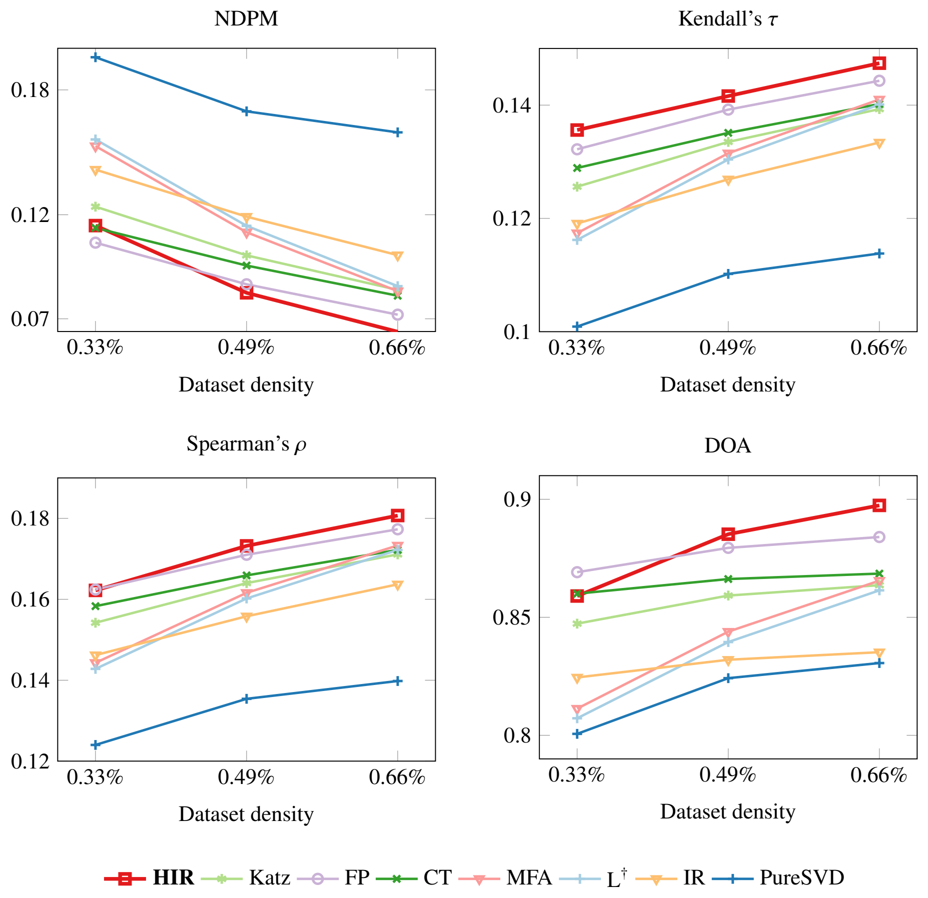


Figure 2: Ranking performance for the *New Community* problem on the `Yahoo!R2Music` dataset.

Figures 2 and 3, show the performance of HIR against the other algorithms for the different stages of the newly emerging system. As expected, L†, MFA, FP, Katz and CT, which can also exploit hierarchy, do better than ItemRank and PureSVD. L† and MFA, in particular in the MovieLens dataset, even though they start rather badly in the sparser dataset, they soon overpass the rest graph-based methods, managing to reach the 2nd and 3rd place, respectively, in the most dense case. FP on the other hand shows consistent performance in both datasets with its advantage being greater in the sparsest cases. Finally, we clearly see that HIR performs very well on both datasets, exhibiting very good results even in the sparsest cases.

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

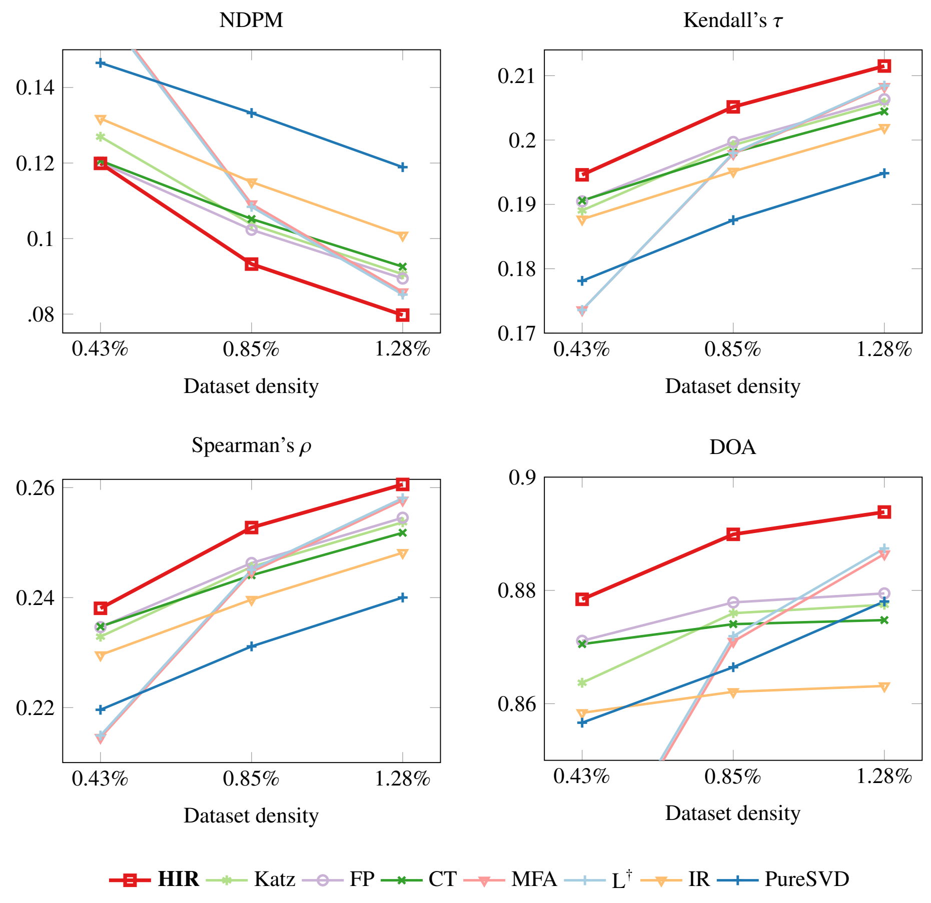


Figure 3: Ranking performance for the *New Community* problem on the `MovieLens1M` dataset.

matrix **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.

3.2.2. New Users

To evaluate the performance of our algorithm in coping with the new users problem, we conduct the following experiment: We randomly select 200 users having rated 100 items or more and then we randomly delete 96%, 94% and 92% of each users' ratings in order to create three datasets. The idea here is that these modified datasets represent "previous versions" of the system, when these users were new, and as such, had rated fewer items.

Following, an approach similar to the new community test presented earlier, we test the positive effect that comes from the introduction of matrix **D**, by running HIR for different values of β, for all the ranking measures, using the complete set of ratings as a reference list. Table 4 reports the average values of each metric over the set of new users, for each dataset. The experiment again verifies that the introduction of even a very small β increases the ranking quality with the best results this time achieved for β around 0.2.

Table 4: HIR performance for the *New Users* problem for varying β on the `Yahoo!R2Music` and `MovieLens1M` datasets.

		`Yahoo!R2Music`							`MovieLens1M`					
	Included Ratings	$\alpha = 0.9$ $\beta = 0$	0.85 0.05	0.8 0.1	0.7 0.2	0.6 0.3	0.5 0.4	**Included Ratings**	$\alpha = 0.9$ $\beta = 0$	0.85 0.05	0.8 0.1	0.7 0.2	0.6 0.3	0.5 0.4
τ	4%	0.2004	0.2031	**0.2032**	0.2028	0.2023	0.2019	4%	0.2538	0.2550	0.2559	**0.2562**	0.2542	0.2496
	6%	0.2074	0.2118	**0.2125**	0.2123	0.2117	0.2111	6%	0.2554	0.2567	0.2577	**0.2585**	0.2574	0.2537
	8%	0.2573	0.2586	0.2598	**0.2610**	0.2604	0.2575	8%	0.2573	0.2586	0.2598	**0.2610**	0.2604	0.2575
NDPM	4%	0.1767	0.1727	**0.1726**	0.1733	0.1741	0.1747	4%	0.1193	0.1169	0.1153	**0.1148**	0.1176	0.1245
	6%	0.1650	0.1583	**0.1574**	0.1578	0.1587	0.1596	6%	0.1166	0.1141	0.1124	**0.1110**	0.1126	0.1179
	8%	0.1570	0.1485	**0.1471**	0.1472	0.1483	0.1495	8%	0.1138	0.1112	0.1093	**0.1073**	0.1080	0.1121
ρ	4%	0.2467	0.2500	**0.2502**	0.2497	0.2491	0.2487	4%	0.3132	0.3150	0.3162	**0.3167**	0.3143	0.3088
	6%	0.2556	0.2610	**0.2618**	0.2616	0.2610	0.2603	6%	0.3154	0.3172	0.3186	**0.3196**	0.3182	0.3139
	8%	0.2618	0.2686	0.2699	**0.2700**	0.2692	0.2683	8%	0.3178	0.3197	0.3212	**0.3227**	0.3220	0.3186
DOA	4%	0.8198	0.8241	**0.8242**	0.8235	0.8228	0.8222	4%	0.8831	0.8857	0.8874	**0.8881**	0.8852	0.8782
	6%	0.8287	0.8360	**0.8370**	0.8366	0.8357	0.8348	6%	0.8837	0.8864	0.8883	**0.8899**	0.8883	0.8827
	8%	0.8336	0.8430	0.8446	**0.8845**	0.8434	0.8422	8%	0.8843	0.8872	0.8894	**0.8917**	0.8910	0.8867

Figures 4 and 5, report the average scores for all different measures and all different percentages of included ratings, for the set of new users. Again, we see that the best results are reached by the methods that can exploit the categorization of items to blocks, as expected. As before, we see that HIR clearly outperforms every other algorithm giving good results even when only 4% of the ratings were included for each user.

The results were expected and are consistent with the way HIR views the problem. Even though new users' tastes are not yet clear, the exploitation of hierarchical proximity can help "propagate" this scarce rating information to the many related elements of the itemspace, giving HIR a "fighting chance" at uncovering new users' preferences.

3.2.3. New Items

Lastly, we evaluate HIR's performance in coping with the new items problem. We randomly select 10%, 12.5% and 15% of the items in the dataset that have at least 30 ratings, and we randomly delete 90% of their ratings in order to create three modified versions of the dataset.

We run HIR and the other algorithms on the original and the three new datasets and we compare the rankings induced on the modified data with their corresponding original rankings resorting to the widely used Kendall's τ, Spearman's ρ and the distance-based NDPM index[8]. Good scores in these tests suggest that the distance of the two ranking lists is small, which means that the new movies are more likely to receive treatment similar to their original one. The results are reported in Table 5.

On the `Yahoo!R2Music` data we see that HIR achieves very good results reaching the first place with respect to the NDPM and the Kendall's τ metrics and the 3rd with respect to the Spearman's ρ metric. On the `MovieLens1M` dataset the results clearly show that HIR outperforms every other recommendation method considered for all different metrics, exhibiting high ranking stability and sparseness insensitivity.

[8]DOA score has only been used in the literature for the evaluation of the quality of a recommendation list with respect to a reference ranking of the user [10, 14, 21]. Its properties as a measure of distance between rankings have not been evaluated. Thus, we decided to avoid using it for our *New Items problem* tests, where we try to evaluate the stability of the ranking vectors.

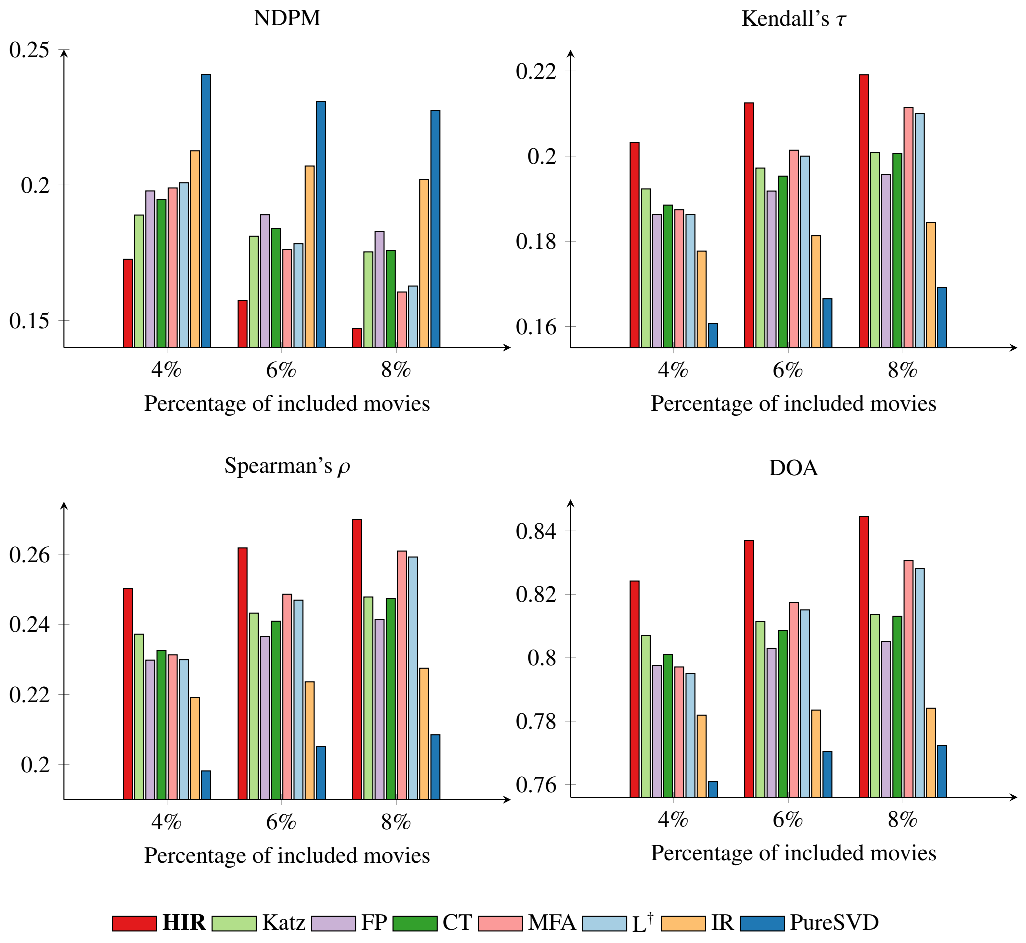


Figure 4: Ranking performance for the *New Users* problem on the `Yahoo!R2Music` dataset.

Again, 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 items to the other elements of the itemspace. Thus, even though the number of ratings is insufficient, the interlevel relations captured by matrix **D**, reduce this gap and make sure that the newly emerging items are treated more fairly.

3.3. Quality of Recommendations

As mentioned earlier, for completeness and in order to make direct comparisons with several other ranking-based methods, we run HIR[9] on the `MovieLens100K` dataset as well. Table 6 presents 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. All the DOA scores included in this table can be found in the literature and refer to

[9]using the formulation of matrix **C** presented in our original paper [24].

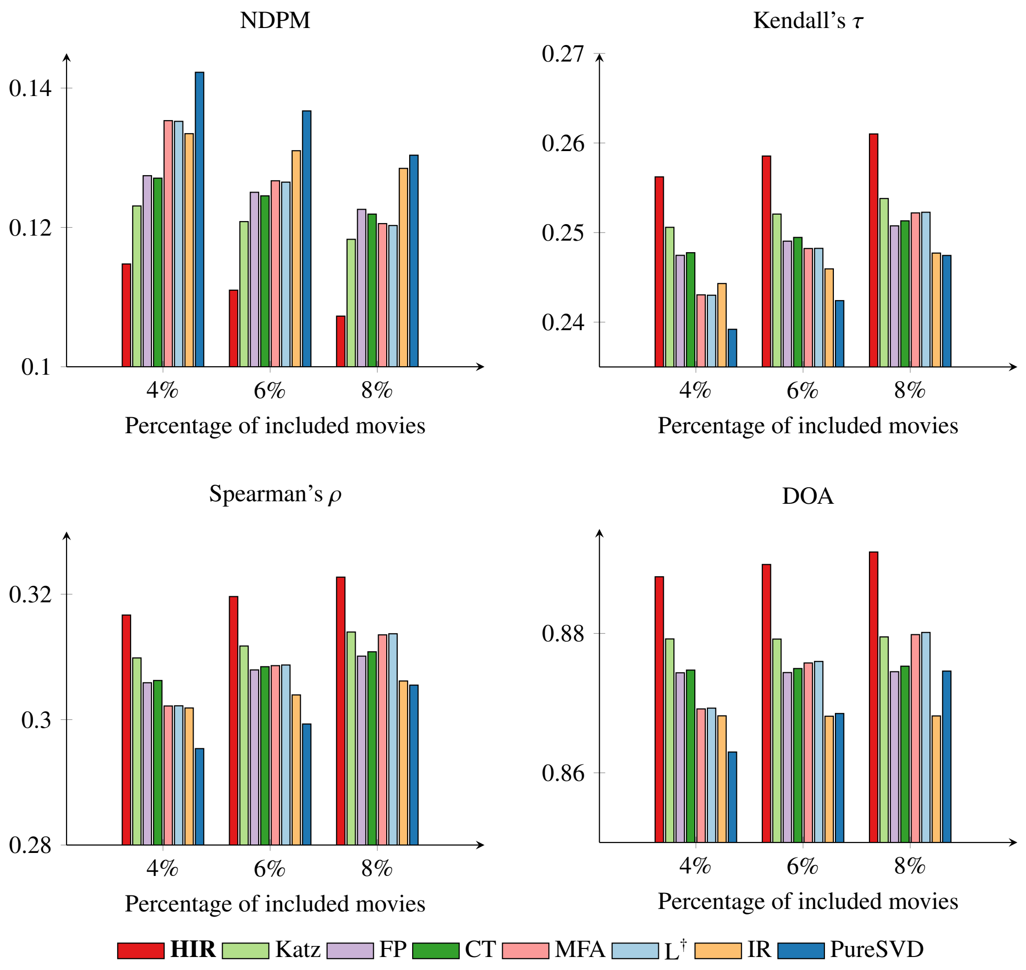


Figure 5: Ranking performance for the *New Users* problem on the `MovieLens1M` dataset.

the same predefined splittings. 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 Markov Random Fields and 88.07 of Hybrid Random Fields models.

4. Future Work

One very interesting research direction we are currently pursuing involves the effect of the granularity of the chosen decomposition of the itemspace. 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

Table 5: Ranking stability for the *New Items* problem on the `Yahoo!R2Music` and `MovieLens1M` datasets.

	`Yahoo!R2Music`								`MovieLens1M`							
	HIR	Katz	FP	CT	MFA	L†	IR	PureSVD	**HIR**	Katz	FP	CT	MFA	L†	IR	PureSVD
10% New Items																
NDPM	**.1593**	.1735	.1644	.1640	.1708	.1692	.1665	.2606	**.0913**	.1211	.1145	.1196	.1201	.1204	.0989	.1609
τ	**.6814**	.6529	.6712	.6719	.6585	.6617	.6669	.4788	**.8174**	.7578	.7710	.7608	.7598	.7593	.8022	.6782
ρ	.6911	.6160	.6041	.6308	.7291	**.7352**	.5849	.6317	**.8881**	.8453	.8539	.8490	.8476	.8479	.8343	.8154
12.5% New Items																
NDPM	**.1904**	.2066	.1938	.1922	.2032	.2016	.2006	.2807	**.1126**	.1396	.1328	.1381	.1373	.1374	.1236	.1841
τ	**.6192**	.5868	.6125	.6156	.5932	.5968	.5987	.4386	**.7749**	.7208	.7345	.7238	.7254	.7253	.7527	.6317
ρ	.6262	.5434	.5330	.5639	.6717	**.6786**	.5056	.5865	**.8576**	.8116	.8223	.8168	.8191	.8196	.7874	.7746
15% New Items																
NDPM	**.2154**	.2325	.2212	.2195	.2277	.2260	.2291	.3067	**.1319**	.1585	.1505	.1563	.1570	.1570	.1460	.1977
τ	**.5691**	.5350	.5577	.5610	.5446	.5479	.5418	.3865	**.7363**	.6829	.6991	.6875	.6859	.6860	.7081	.6046
ρ	.5713	.4760	.4604	.4954	.6241	**.6308**	.4266	.5267	**.8315**	.7768	.7908	.7845	.7838	.7846	.7459	.7493

Table 6: Average performance between HIR and other state-of-the-art recommendation algorithms, computed over the same standard predefined folds using the `MovieLens100K`, as published in the literature [14, 21, 24, 29].

	`macro-DOA`	`micro-DOA`
Hierarchical Itemspace Rank	**89.99**	**88.85**
Hybrid Random Fields	89.83	88.07
Markov Random Fields	89.47	88.09
Topical PageRank	89.08	–
Naive Bayes	88.87	86.66
ItemRank	87.76	87.06
Katz	85.85	–
L†	87.23	–
Commute Time	84.09	–
First Passage	84.08	–
MaxF	84.07	–
PCA Commute Time	84.04	–
Dependency Networks	80.51	81.33

performance metrics. Notice, that in our method this can be achieved readily, through the introduction of new low rank hierarchical proximity matrices, $\mathbf{D_1}, \mathbf{D_2}, \ldots$ and associated parameters $\beta_1, \beta_2, \ldots$, with no effect on the dimensionality of the model.

5. Conclusions

In this paper we proposed *Hierarchical Itemspace Rank* (HIR); a novel recommender method that exploits the innate hierarchical structure of the itemspace to provide an elegant method for generating ranking-based recommendations. HIR decomposes the itemspace into item-blocks and exploits this decomposition to define new levels of indirect proximity between the elements of the dataset. This view gives rise to a corresponding stochastic matrix which can be used to enrich the direct inter-item relations and reduce the sensitivity to sparsity. The resulting model is scalable and computationally efficient due to the fact that its dimension is independent of the number of users, as well as because of the attractive mathematical properties of the hierarchical proximity matrix, which allow easy computational handling.

A comprehensive set of experiments on the `MovieLens` as well as the `Yahoo!R2Music` datasets suggests that our method exhibits low susceptibility to the problems caused by *Sparsity* and proves to be very effective in handling even its most extreme manifestations – the *cold-start problems*. HIR achieves high quality results in both music and movie recommendation domains for various metrics, outperforming several state-of-the-art recommender algorithms, including methods of – the promising for its anti-sparsity properties – graph-based recommender family.

References

[1] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th international conference on World Wide Web, WWW '01, ACM, New York, NY, USA, 2001, pp. 285–295. doi:10.1145/371920.372071.
URL http://doi.acm.org/10.1145/371920.372071

[2] C. Desrosiers, G. Karypis, A comprehensive survey of neighborhood-based recommendation methods, in: F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), Recommender Systems Handbook, Springer US, 2011, pp. 107–144. doi:10.1007/978-0-387-85820-3_4.
URL http://dx.doi.org/10.1007/978-0-387-85820-3_4

[3] J. Bobadilla, F. Ortega, A. Hernando, A. GutiéRrez, Recommender systems survey, Know.-Based Syst. 46 (2013) 109–132. doi:10.1016/j.knosys.2013.03.012.
URL http://dx.doi.org/10.1016/j.knosys.2013.03.012

[4] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37. doi:10.1109/MC.2009.263.
URL http://dx.doi.org/10.1109/MC.2009.263

[5] Y. Koren, R. Bell, Advances in collaborative filtering, in: F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), Recommender Systems Handbook, Springer US, 2011, pp. 145–186. doi:10.1007/978-0-387-85820-3_5.
URL http://dx.doi.org/10.1007/978-0-387-85820-3_5

[6] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, ACM, New York, NY, USA, 2008, pp. 426–434. doi:10.1145/1401890.1401944.
URL http://doi.acm.org/10.1145/1401890.1401944

[7] S. Balakrishnan, S. Chopra, Collaborative ranking, in: Proceedings of the fifth ACM international conference on Web search and data mining, WSDM '12, ACM, New York, NY, USA, 2012, pp. 143–152. doi:10.1145/2124295.2124314.
URL http://doi.acm.org/10.1145/2124295.2124314

[8] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the fourth ACM conference on Recommender systems, RecSys '10, ACM, 2010, pp. 39–46. doi:10.1145/1864708.1864721.
URL http://doi.acm.org/10.1145/1864708.1864721

[9] B. M. Marlin, R. S. Zemel, Collaborative prediction and ranking with non-random missing data, in: Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09, ACM, New York, NY, USA, 2009, pp. 5–12. doi:10.1145/1639714.1639717.
URL http://doi.acm.org/10.1145/1639714.1639717

[10] F. Fouss, A. Pirotte, J.-M. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Trans. on Knowl. and Data Eng. 19 (3) (2007) 355–369. doi:10.1109/TKDE.2007.46.
URL http://dx.doi.org/10.1109/TKDE.2007.46

[11] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, M. Saerens, An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification, Neural Netw. 31 (2012) 53–72. doi:10.1016/j.neunet.2012.03.001.
URL http://dx.doi.org/10.1016/j.neunet.2012.03.001

[12] Y. Koren, J. Sill, Ordrec: An ordinal model for predicting personalized item rating distributions, in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, ACM, New York, NY, USA, 2011, pp. 117–124. doi:10.1145/2043932.2043956.
URL http://doi.acm.org/10.1145/2043932.2043956

[13] A. N. Nikolakopoulos, M. Kalantzi, J. D. Garofalakis, On the use of lanczos vectors for efficient latent factor-based top-n recommendation, in: Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14), WIMS '14, ACM, New York, NY, USA, 2014, pp. 28:1–28:6. doi:10.1145/2611040.2611078.
URL http://doi.acm.org/10.1145/2611040.2611078

[14] A. Freno, E. Trentin, M. 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, ACM, 2009, pp. 319–328. doi:10.1145/1557019.1557059.
URL http://doi.acm.org/10.1145/1557019.1557059

[15] A. Freno, E. Trentin, Hybrid Random Fields - A Scalable Approach to Structure and Parameter Learning in Probabilistic Graphical Models, Vol. 15 of Intelligent Systems Reference Library, Springer, 2011.

[16] P. Domingos, M. Pazzani, On the optimality of the simple bayesian classifier under zero-one loss, Mach. Learn. 29 (2-3) (1997) 103–130. doi:10.1023/A:1007413511361.
URL http://dx.doi.org/10.1023/A:1007413511361

[17] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, A. Hanjalic, Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering, in: Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, ACM, New York, NY, USA, 2012, pp. 139–146. doi:10.1145/2365952.2365981.
URL http://doi.acm.org/10.1145/2365952.2365981

[18] J. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Vol. 4321 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 291–324. doi:10.1007/978-3-540-72079-9_9.
URL http://dx.doi.org/10.1007/978-3-540-72079-9_9

[19] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. on Knowl. and Data Eng. 17 (6) (2005) 734–749. doi:10.1109/TKDE.2005.99.
URL http://dx.doi.org/10.1109/TKDE.2005.99

[20] H. Yildirim, M. S. Krishnamoorthy, A random walk method for alleviating the sparsity problem in collaborative filtering, in: Proceedings of the 2008 ACM conference on Recommender systems, RecSys '08, ACM, 2008, pp. 131–138. doi:10.1145/1454008.1454031.
URL http://doi.acm.org/10.1145/1454008.1454031

[21] M. Gori, A. Pucci, Itemrank: A random-walk based scoring algorithm for recommender engines, in: Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 2766–2771.
URL http://dl.acm.org/citation.cfm?id=1625275.1625720

[22] P. Chebotarev, E. Shamis, The matrix-forest theorem and measuring relations in small social groups, Automation and Remote Control 58 (9) (1997) 1505–1514.

[23] L. Katz, A new status index derived from sociometric analysis, Psychometrika 18 (1) (1953) 39–43. doi:10.1007/BF02289026.
URL http://dx.doi.org/10.1007/BF02289026

[24] A. N. Nikolakopoulos, M. Kouneli, J. Garofalakis, A novel hierarchical approach to ranking-based collaborative filtering, in: L. Iliadis, H. Papadopoulos, C. Jayne (Eds.), Engineering Applications of Neural Networks, Vol. 384 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2013, pp. 50–59. doi:10.1007/978-3-642-41016-1_6.
URL http://dx.doi.org/10.1007/978-3-642-41016-1_6

[25] A. N. Nikolakopoulos, J. 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, ACM, New York, NY, USA, 2013, pp. 143–152. doi:10.1145/2433396.2433415.
URL http://doi.acm.org/10.1145/2433396.2433415

[26] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, 2nd Edition, Addison-Wesley Publishing Company, USA, 2008.

[27] G. Shani, A. Gunawardana, Evaluating recommendation systems, in: F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), Recommender Systems Handbook, Springer US, 2011, pp. 257–297. doi:10.1007/978-0-387-85820-3_8.
URL http://dx.doi.org/10.1007/978-0-387-85820-3_8

[28] Y. Y. Yao, Measuring retrieval effectiveness based on user preference of documents, Journal of the American Society for Information Science 46 (2) (1995) 133–145.
URL http://dx.doi.org/10.1002/(SICI)1097-4571(199503)46:2{<}133::AID-ASI6>3.0.CO;2-Z

[29] L. Zhang, C. Li, A novel recommending algorithm based on topical pagerank, in: Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence, AI '08, Springer-Verlag,

Berlin, Heidelberg, 2008, pp. 447–453. doi:10.1007/978-3-540-89378-3_45.
URL http://dx.doi.org/10.1007/978-3-540-89378-3_45

[30] S. Siegel, N. J. Castellan Jr, Nonparametric statistics for the behavioral sciences ., Mcgraw-Hill Book Company, 1988.