# Semi-supervised Collaborative Ranking with Push at Top

Iman Barjasteh[†*], Rana Forsati[‡*], Abdol-Hossein Esfahanian[‡], Hayder Radha[†]
[†]Department of Electrical and Computer Engineering, Michigan State University
[‡]Department of Computer Science and Engineering, Michigan State University
{forsati,esfahanian}@cse.msu.edu, {barjaste,radha}@msu.edu

## Abstract

Existing collaborative ranking based recommender systems tend to perform best when there is enough observed ratings for each user and the observation is made completely at random. Under this setting recommender systems can properly suggest a list of recommendations according to the user interests. However, when the observed ratings are extremely sparse (e.g. in the case of cold-start users where no rating data is available), and are not sampled uniformly at random, existing ranking methods fail to effectively leverage side information to transduct the knowledge from existing ratings to unobserved ones. We propose a **semi-supervised collaborative ranking** model, dubbed SCR, to improve the quality of cold-start recommendation. SCR mitigates the sparsity issue by leveraging side information about *both observed and missing* ratings by collaboratively learning the ranking model. This enables it to deal with the case of missing data not at random, but to also effectively incorporate the available side information in transduction. We experimentally evaluated our proposed algorithm on a number of challenging real-world datasets and compared against state-of-the-art models for cold-start recommendation. We report significantly higher quality recommendations with our algorithm compared to the state-of-the-art.

## 1 Introduction

Due to the popularity and exponential growth of e-commerce and online streaming websites, a compelling demand has been created for efficient recommender systems to guide users toward items of their interests (e.g. products, books, movies) [1]. In collaborative filtering (CF) methods such as matrix factorization [13, 5, 18], where the aim is to accurately predict the ratings, the latent features are extracted in a way to minimize the prediction error measured in terms of popular performance measures such as root mean square error (RMSE). In spark contrast to CF, in collaborating ranking (CR) models [13, 8, 31, 9], where the goal is to rank the unrated items in the order of relevance to the user, the popular ranking measures such as as discounted cumulative gain (DCG), normalized discounted cumulative gain (NDCG), and average precision (AP) [12] are often employed to collaboratively learn a ranking model for the latent features.

Recent studies have demonstrated that CR models lead to significantly higher ranking accuracy over their traditional CF counterparts that optimize rating prediction. This is important considering the fact that what we really care in recommendation is not the actual values of ratings, but the order of items to be recommended to a specific user. Therefore, the error measures such as RMSE are often hopelessly insufficient, as they place equal emphasis on all the ratings. Among ranking models, the methods that mainly concentrate on the *top of the list* have received a considerable amount of attention, due to the higher probability of examining the top portion of the list of recommendations by users. Therefore, the introduction of ranking metrics such as push norm or infinite norm [21, 3, 8, 14], sparked a widespread interest in CR models and has been proven to be more effective in practice [30, 8].

Although CR models for recommender systems has been studied extensively and some progress has been made, however, the state of affairs remains unsettled: the issue of handling *cold-start* items in ranking models and coping with *not missing at random* assumption of ratings are elusive open issues. First, in many real world applications, the rating data are very sparse (e.g., the density of the data is around 1% for many publicly available datasets) or for a subset of users or items the rating data is entirely missing

[*]These authors contributed equally to this work.

(knows as cold-start user and cold-start item problem, respectively) [23]. Second, collaborative filtering and ranking models rely on the critical assumption that the missing ratings are sampled uniformly at random. However, in many real applications of recommender systems, this assumption is not believed to hold, as invariably some users are more active than others and some items are rated by many people while others are rarely rated [29]. These issues have been investigated in factorization based methods, nonetheless, it is not straightforward to adapt them to CR models and are left open [8].

In this paper, we introduce a *semi-supervised collaborative ranking* model, dubbed `SCR` , by leveraging side information about *both observed and missing ratings* in collaboratively learning the ranking model. In the learned model, unrated items are conservatively pushed after the relevant and before the irrelevant items in the ranked list of items for each individual user. This crucial difference greatly boosts the performance and limits the bias caused by learning only from sparse non-random observed ratings.

To build the intuition on how incorporating missing ratings in `SCR` is beneficial in handling cold-start problem and mitigating data sparsity issue, we note that in many real world applications the available feedback on items is extremely sparse, and therefore the ranking models fail to effectively leverage the available side information in transducting the knowledge from existing ratings to unobserved ones. This problem becomes especially eminent in cases where surrogate ranking models such as pairwise models are used due to their computational virtues, where the unobserved ratings do not play any role in learning the model. As a result, by leveraging rich sources of information about all items, one can potentially bridge the gap between existing items and new items to overcome the cold-start problem.

Turning to the non-random sampling issue of observed ratings, we note that the non-randomness is observing the ratings creates a bias in learning the model that negatively impacts the future predictions and may degrade the resulting recommendation accuracy if ignored. Therefore, the nature of missing ratings has to be modeled precisely as to obtain correct results. To reduce the effect of bias, the proposed ranking model takes a conservative approach and pushes the items with unknown ratings to the middle of ranked list, i.e., after the relevant and before the irrelevant items. This is equivalent to assuming a prior about the unknown ratings which is believed to perform well as investigated in [10].

We conduct thorough experiments on real datasets and compare our results with the state-of-the-art models for cold-start recommendation to demonstrate the effectiveness of our proposed algorithm in recommendation at the top of the list and mitigating the data sparsity issue.

**Organization.** This paper is organized as follows. We briefly review related work in Section 2. We establish the notation and formally define the problem in Section 3. In Section 4, we propose the semi-supervised collaborative ranking model with a push at the top of the list. We empirically evaluate the proposed method in Section 5, and conclude in Section 6.

## 2 Related Work

**Collaborative ranking for recommendation.** The last few years have seen a resurgence in collaborative ranking centered around the technique of exploiting low-rank structures, an approach we take as well. Several approaches to CR have recently been proposed that are mainly inspired by the analogy between query-document relations in IR and user-item relations in recommender systems. The PMF-based approach [4] uses the latent representations produced by matrix factorization as user-item features and learns a ranking model on these features. CofiRank [32] learns latent representations that minimize a ranking-based loss instead of the squared error. ListRankMF [25] aims at minimizing the cross entropy between the predict item permutation probability and true item permutation probability. In [14] a method for Local Collaborative Ranking (LCR) where ideas of local low-rank matrix approximation were applied to the pairwise ranking loss minimization framework is introduced.

**Cold-start recommendation with side information.** Due in part to its importance, there has been an active line of work to address difficulties associated with cold-start users and items, where a common theme among them is to exploit auxiliary information about users or items besides the rating data that are usually available [26]. A feature based regression ranking model for predicting the values (rates) of user-item matrix in cold-start scenarios by leveraging all information available for users and items is proposed in [19]. The kernelized matrix factorization approach studied in [33], which incorporates the auxiliary information into the MF. In [22] joint factorization of the user-item and item-feature matrices by using the same item latent feature matrix in both decompositions is utilized.

**Recommendation with not missing at random ratings.** Substantial evidence for violations of the missing at random condition in recommender systems is reported in [17] and it has been showed that incorporating an explicit model of the missing data mechanism can lead to significant improvements in prediction performance.The first study of the effect of non-random missing data on collaborative ranking is presented in [16]. In [27] an EM algorithm to optimize in turn the factorization and the estimation of missing values.

## 3 Preliminaries

In this section we establish the notation used throughout the paper and formally describe our problem setting.

Scalars are denoted by lower case letters and vectors by bold face lower case letters such as $\mathbf{u}$. We use bold face upper case letters such as $\mathbf{M}$ to denote matrices. The Frobenius norm of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ is denoted by $\|\mathbf{M}\|_{\mathrm{F}}$, i.e, $\|\mathbf{M}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|M_{ij}|^2}$ and its $(i,j)$th entry is denoted by $A_{i,j}$. The trace norm of a matrix is denoted by $\|\mathbf{M}\|_*$ which is defined as the sum of its singular values. The transpose of a vector and a matrix denoted by $\mathbf{u}^\top$ and $\mathbf{U}^\top$, respectively. We use $[n]$ to denote the set on integers $\{1, 2, \cdots, n\}$. The set of non-negative real numbers is denoted by $\mathbb{R}_+$. The indicator function is denoted by $\mathbb{I}[\cdot]$. For a vector $\mathbf{u} \in \mathbb{R}^p$ we use $\|\mathbf{u}\|_1 = \sum_{i=1}^{p}|\mathbf{u}_i|$, $\|\mathbf{u}\|_2 = \left(\sum_{i=1}^{p}|\mathbf{u}_i|^2\right)^{1/2}$, and $\|\mathbf{u}\|_\infty = \max_{1\leq i\leq p}\mathbf{u}_i$ to denote its $\ell_1$, $\ell_2$, and $\ell_\infty$ norms, respectively. The dot product between two vectors $\mathbf{u}$ and $\mathbf{u}'$ is denoted by either $\langle\mathbf{u}, \mathbf{u}'\rangle$ or $\mathbf{u}^\top\mathbf{u}'$.

In collaborative filtering we assume that there is a set of $n$ users $\mathcal{U} = \{u_1, \cdots, u_n\}$ and a set of $m$ items $\mathcal{I} = \{i_1, \cdots, i_m\}$ where each user $u_i$ expresses opinions about a set of items. The rating information is summarized in an $n \times m$ matrix $\mathbf{R} \in \{-1, +1, ?\}^{n\times m}, 1 \leq i \leq n, 1 \leq j \leq m$ where the rows correspond to the users and the columns correspond to the items and $(p, q)$th entry is the rate given by user $u_p$ to the item $i_q$. We note that the rating matrix is partially observed and it is sparse in most cases. We are mainly interested in recommending a set of items for an active user such that the user has not rated these items before.

## 4 Transductive Collaborating Ranking

We now turn our attention to the main thrust of the paper where we present our transductive collaborative ranking algorithm with accuracy at top by exploiting the features of unrated data. We begin with the basic formulation and then extend it to incorporate the unrated items.

### 4.1 A basic formulation

We consider a ranking problem, where, given a set of users $\mathcal{U}$ and known user feedback on a set of items $\mathcal{I}$, the goal is to generate rankings of unobserved items, adapted to each of the users' preferences. Here we consider the bipartite setting in which items are either relevant (positive) or irrelevant (negative). Many ranking methods have been developed for bipartite ranking, and most of them are essentially based on pairwise ranking. These algorithms reduce the ranking problem into a binary classification problem by treating each relevant/irrelevant instance pair as a single object to be classified [15].

As mentioned above, most research has concentrated on the rating prediction problem in CF where the aim is to accurately predict the ratings for the unrated items for each user. However, most applications that use CF typically aim to recommend only a small ranked set of items to each user. Thus rather than concentrating on rating prediction we instead approach this problem from the ranking viewpoint where the goal is to rank the unrated items in the order of relevance to the user. Moreover, it is desirable to concentrate aggressively on top portion of the ranked list to include mostly relevant items and push irrelevant items down from the top. Specifically, we propose an algorithm that maximizes the number of relevant items which are pushed to the absolute top of the list by utilizing the P-Norm Push ranking measure which is specially designed for this purpose [21] .

For simplicity of exposition, let us first consider the ranking model for a single user $u$. Let $\mathcal{X}^+ = \{\mathbf{x}_1^+, \cdots, \mathbf{x}_{n_+}^+\}$ and $\mathcal{X}^- = \{\mathbf{x}_1^-, \cdots, \mathbf{x}_{n_-}^-\}$ be the set of feature vectors of $n_+$ relevant and $n_-$ irrelevant items to user $u$, respectively. We consider linear ranking functions where each item features vector $\mathbf{x} \in \mathbb{R}^d$ is mapped to a score $\mathbf{w}^\top\mathbf{x}$ . The goal is to find parameters $\mathbf{w}$ for each user such that the ranking function best captures past feedback from the user. The goal of ranking is to maximize the number of relevant items ranked above the highest-ranking irrelevant item. We cast this idea for each user $u$ individually into the following optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n^+}\sum_{i=1}^{n^+} \mathbb{I}\left[\langle \mathbf{w}, \mathbf{x}_i^+\rangle \leq \max_{1\leq j\leq n^-}\langle \mathbf{w}, \mathbf{x}_j^-\rangle\right] \tag{1}$$

where $\mathbb{I}[\cdot]$ is the indicator function which returns 1 when the input is true and 0 otherwise, $n^+$ and $n^-$ are the the number of relevant and irrelevant items to user $u$, respectively.

Let us now derive the general form of our objective. We hypothesize that most users base their decisions about items based on a number of latent features about the items. In order to uncover these latent feature dimensions, we impose a low-rank constraint on the set of parameters for all users. To this end, let $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n]^\top \in \mathbb{R}^{n\times d}$ denote the matrix of all parameter vectors for $n$ users. Let $\mathcal{I}_i^+ \subseteq \{1, 2, \ldots, m\}$ and $\mathcal{I}_i^- \subseteq \{1, 2, \ldots, m\}$ be the set of relevant and irrelevant items of $i$th user, respectively. The overall objective for all users is formulated as follows:

$$\begin{aligned}\mathcal{F}(\mathbf{W}) &= \lambda\|\mathbf{W}\|_* \\ &+ \sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^+|}\sum_{j\in\mathcal{I}_i^+}\mathbb{I}\left[\langle \mathbf{w}_i, \mathbf{x}_j\rangle \leq \max_{k\in\mathcal{I}_i^-}\langle \mathbf{w}_i, \mathbf{x}_k\rangle\right]\right)\end{aligned} \tag{2}$$

where $\|\cdot\|_*$ is the trace norm (also known as nuclear norm) which is the sum of the singular values of the input matrix.

The objective in Eq. (2) is composed of two terms. The first term is the regularization term and is introduced to capture the factor model intuition discussed above. The premise behind a factor model is that there is only a small number of factors influencing the preferences, and that a user's preference vector is determined by how each factor applies to that user. Therefore, the parameter vectors of all users must lie in a low-dimensional subspace. Trace-norm regularization is a widely-used and successful approach for collaborative filtering and matrix completion. The trace-norm regularization is well-known to be a convex surrogate to the matrix rank, and has repeatedly shown good performance in practice [28, 7]. The second term is introduced to push the relevant items of each user to the top of the list when ranked based on the user parameter vector and item features.

The above optimization problem is intractable due to the non-convex indicator function. To design practical learning algorithms, we replace the indicator function in (2) with its convex surrogate. To this end, define the convex loss function $\ell : \mathbb{R} \mapsto \mathbb{R}_+$ as $\ell(x) = [1 - x]_+$. This is the widely used hinge loss in SVM classification (see e.g., [6]) [1]. This loss function reflects the amount by which the constraints are not satisfied. By replacing the non-convex indicator function with this convex surrogate leads to the following tractable convex optimization problem:

$$\begin{aligned}\mathcal{F}(\mathbf{W}) &= \lambda\|\mathbf{W}\|_* \\ &+ \sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^+|}\sum_{j\in\mathcal{I}_i^+}\ell\left(\langle \mathbf{w}_i, \mathbf{x}_j\rangle - \|\mathbf{X}_i^-\mathbf{w}_i\|_\infty\right)\right)\end{aligned} \tag{3}$$

where $\mathbf{X}_i^- = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_i^-}]^\top$ is the matrix of features of $n_i^-$ irrelevant items in $\mathcal{I}_i^-$ and $\|\cdot\|_\infty$ is the max norm of a vector.

## 4.2 Semi-supervised collaborative ranking

In this part, we extend the proposed ranking idea to learn both from rated as well as unrated items. The motivation of incorporating unrated items comes from the following key observations. First, we note that commonly there is a small set of rated (either relevant or irrelevant) items for each user and a large number of unrated items. As it can be seen from Eq. (2), the unrated items do not play any role in learning the model for each user as the learning is only based on the pair of rated items. When the feature information for items is available, it would be very helpful if one can leverage such unrated items in the learning-to-rank process to effectively leverage the available side information. By leveraging both types of rated and unrated items, we can compensate for the lack of rating data. Second, the non-randomness in observing the observed ratings creates a bias in learning the model that may degrade the resulting recommendation accuracy. Therefore, finding a precise model to reduce the effect of bias introduced by non-random missing ratings seems essential.

To address these two issues, we extend the basic formulation in Eq. (2) to incorporate items with missing ratings in ranking of items for individual users. A conservative solution is to push the items with unknown ratings to the middle of ranked list, i.e., after the relevant and before the irrelevant items. To do so, let $\mathcal{I}_i^\circ = \mathcal{I}\setminus\left(\mathcal{I}_i^+ \cup \mathcal{I}_i^-\right)$ denote the set of items unrated for user $i \in \mathcal{U}$. We introduce two extra terms in the objective in Eq. (2) to push the unrated items

[1]We note that other convex loss functions such as exponential loss $\ell(x) = \exp(-x)$, and logistic loss $\ell(x) = \log(1 + \exp(-x))$ also can be used as the surrogates of indicator function, but for the simplicity of derivation we only consider the hinge loss here.

$\mathcal{I}_\circ^i$ below the relevant items and above the irrelevant items, which yields the following objective:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) = \quad & \frac{1}{|\mathcal{I}_i^+|}\sum_{i\in\mathcal{I}_i^+}\ell\left(\langle\mathbf{w},\mathbf{x}_i\rangle \le \max_{j\in\mathcal{I}_i^-}\langle\mathbf{w},\mathbf{x}_j\rangle\right)\\ + \quad & \frac{1}{|\mathcal{I}_i^+|}\sum_{i\in\mathcal{I}_i^+}\ell\left(\langle\mathbf{w},\mathbf{x}_i\rangle \le \max_{j\in\mathcal{I}_i^\circ}\langle\mathbf{w},\mathbf{x}_j\rangle\right)\\ + \quad & \frac{1}{|\mathcal{I}_i^\circ|}\sum_{i\in\mathcal{I}_i^\circ}\ell\left(\langle\mathbf{w},\mathbf{x}_i\rangle \le \max_{j\in\mathcal{I}_i^-}\langle\mathbf{w},\mathbf{x}_j\rangle\right)\end{aligned} \tag{4}$$

Equipped with the objective of individual users, we now turn to the final collaborating ranking objective as:

$$\begin{aligned}&\mathcal{F}(\mathbf{W}) = \lambda\|\mathbf{W}\|_*\\ &+\sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^+|}\sum_{j\in\mathcal{I}_i^+}\ell\left(\langle\mathbf{w}_i,\mathbf{x}_j\rangle - \|\mathbf{X}_i^-\mathbf{w}_i\|_\infty\right)\right)\\ &+\sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^+|}\sum_{j\in\mathcal{I}_i^+}\ell\left(\langle\mathbf{w}_i,\mathbf{x}_j\rangle - \|\mathbf{X}_i^\circ\mathbf{w}_i\|_\infty\right)\right)\\ &+\sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^\circ|}\sum_{j\in\mathcal{I}_i^\circ}\ell\left(\langle\mathbf{w}_i,\mathbf{x}_j\rangle - \|\mathbf{X}_i^-\mathbf{w}_i\|_\infty\right)\right),\end{aligned} \tag{5}$$

where $\mathbf{X}_i^\circ = [\mathbf{x}_1,\ldots,\mathbf{x}_{n_i^\circ}]^\top$ is the matrix of $n_i^\circ$ unrated items in $\mathcal{I}_i^\circ$.

# 5 Experiments

In this section, we conduct exhaustive experiments to demonstrate the merits and advantages of the proposed algorithm. We conduct our experiments on three well-known datasets MovieLens, Amazon and CiteULike.

## 5.1 Datasets

- **ML-IMDB.** We used ML-IMDB which is a dataset extracted from the IMDB and the MovieLens 1M datasets by mapping the MovieLens and IMDB and collecting the movies that have plots and keywords.
- **Amazon.** We used the dataset of best-selling books and their ratings in Amazon. Each book has a one or two paragraphs of textual description, which has been used to have a set of features of the books.

Table 1: Statistics of real datasets used in our experiments.

| Statistics | ML-IMDB | Amazon | CiteULike |
|---|---|---|---|
| # users | 2,113 | 13,097 | 3,272 |
| # items | 8,645 | 11,077 | 21,508 |
| # ratings | 739,973 | 175,612 | 180,622 |
| # features | 8,744 | 5,766 | 6,359 |
| Density | 4.05% | 0.12% | 0.13% |

- **CiteULike.** It is an online free service for managing and discovering scholarly references. Users can add those articles that they are interested in to their libraries. Collected articles in a user's library will be considered as relevant items for that user. This dataset does not have explicit irrelevant items and was chosen to illustrate the effect of considering missing data while only having relevant items.

For all above datasets, the description about the items were tokenized and after removing the stop words, the rest of the words were stemmed. Then those words that have been appeared in less than 20 items and more that 20% of the items were also removed [24]. At the end, the TF-IDF was applied on the remaining words and the TF-IDF scores represented the features of the items. The statistics of the datasets are given in Table 1. As it is shown in Table 1, all these three datasets have high dimensional feature space.

## 5.2 Metrics

We adopt the widely used metrics, Discounted Cumulative Gain at $n$ and Recall at $n$, for assessing the performance of our and baseline algorithms. For each user $u$, given an item $i$, let $s_k$ be the relevance score of the item ranked at position $k$, where $s_k = 1/n$ if the item is relevant to the user $u$ and $s_k = 0$ otherwise. Discounted Cumulative Gain at $n$, is defined as:

$$\mathrm{DCG_u}@n = s_1 + \sum_{k=2}^{n}\frac{s_k}{\log_2(k)}$$

If we divide the $\mathrm{DCG_u}@n$ by its maximum value, we get the $\mathrm{NDCG_u}@n$ value. Given the list of top-$n$ item recommendations for each user $u$, Recall at $n$ will count the number of relevant items appeared in that list. Recall at $n$ is defined as:

$$\mathrm{REC_u}@n = \frac{|\{\text{relevant items to } u\} \cap \{\text{top-}n \text{ items}\}|}{|\{\text{top-}n \text{ items}\}|}$$

DCG@$n$, $\mathrm{NDCG_u}@n$ and REC@$n$ will be computed for each user and then will be averaged over all users.

## 5.3 Methodology

Given the partially observed rating matrix, we transformed the observed ratings of all datasets from a multi-level relevance scale to a two-level scale $(+1, -1)$ while 0 is considered for unobserved ratings. We randomly selected 60% of the observed ratings for training and 20% for validation set and consider the remaining 20% of the ratings as our test set. To better evaluate the results, we performed a 3-fold-cross validation and reported the average value for our results.

## 5.4 Baseline Algorithms

The proposed `SCR` algorithm is compared to the following algorithms:

- **Feature Based Factorized Bilinear Similarity Model (FBS) [24]:** This algorithm uses bilinear model to capture pairwise dependencies between the features.
- **Collaborative User-specific Feature-based Similarity Models (CUFSM):** By using the history of ratings for users, it learns personalized user model across the dataset [11].
- **Regression based Latent Factor Model (RLF):**[2] This method incorporates the features of items in factorization process by transforming the features to the latent space using linear regression [2]. If the learning method is Markov Chain Monte Carlo, we name it RLF-MCMC.
- **Cosine Similarity Based Recommender (CSR):** Using the similarity between features of items, the preference score of a user on an item will be estimated.

## 5.5 Robustness to not missing at random ratings

In this section we compare the effect of incorporating the unobserved ratings in our learning in comparison with excluding them from our learning. Most of the methods in the literature ignore the unobserved ratings and train their model only base on observed ratings. By incorporating the unrated items in ranking, our method can limit the bias caused by learning solely based on the observed ratings and consequently deals with the not missing at random issue of ratings. Table 2 shows results of comparing these two scenarios for `SCR` on ML-IMDB. In order to see the difference between these two scenarios, we considered 70% of the ratings for training and 30% for test to have more ground truth for our testing. Table 2 shows the NDCG@5, 10,15 and 20 for both scenarios and it shows that incorporating the unobserved ratings causes to improve the accuracy of recommendation list. Hence, the NDCG values for top 5, 10, 15 and 20 items improved when unrated items were included as part of the training process.

[2]The implementation of this method is available in LibFM library [20].

## 5.6 Dealing with cold-start items

We now turn to evaluating the effectiveness of `SCR` for cold-start recommendation. To do so, we randomly selected 60% of the items as our training items and 20% for validation set and considered the remaining 20% of the items as our test set. In this scenario, baseline algorithms that are used for comparison are CSR, FBS, CUFSM and RLF. For the experiments, we used ML-IMDB, Amazon and CiteULike datasets. Table 3 shows the measurement results of applying mentioned algorithms on these datasets. For each test, the parameters' values producing the best ranking on the validation set were selected to be used and reported. As it can be seen from the results in Table 3, the proposed `SCR` algorithm outperformed all other baseline algorithms and provided a recommendations with higher quality in comparison to other methods. We can also see from the results of Table 3 that for the ML-IMDB dataset, the improvement in terms of REC@10 is significant compared to other datasets. Since the density of this dataset is much higher than other two datasets, this observation indicates that our method is more effective in utilizing side information compared to other methods. These results demonstrate the effectiveness of `SCR` in comparison with other state-of-the-art algorithms. `SCR` was able to outperform other state-of-the-art algorithms by considering the missing data and focusing on top of the recommendation list for cold-start items.

# 6 Conclusions

In this paper we introduced a semi-supervised collaborative ranking model by leveraging side information about both observed and missing ratings in collaboratively learning the ranking model. In the learned model, unrated items are conservatively pushed after the relevant and before the irrelevant items in the ranked list of items for each individual user. This crucial difference greatly boosts the performance and limits the bias caused by learning only from sparse

Table 2: Results of employing missing ratings versus ignoring them on ML-IMDB. $\lambda = 0.6$ is regularization parameter, $h = 10$ is dimension of latent features, $T = 100$ is the number of iterations.

| **Algorithm:** `SCR` | **NDCG@5** | **NDCG@10** | **NDCG@15** | **NDCG@20** |
|---|---|---|---|---|
| Observed ratings | 1.1690 | 2.2218 | 2.8362 | 3.2849 |
| Observed + missing ratings | **1.1794** | **2.2405** | **2.8585** | **3.3096** |

non-random observed ratings. The proposed algorithm is compared with seven baseline algorithms on three real world datasets that demonstrated the effectiveness of proposed algorithm in addressing cold-start problem and mitigating the data sparsity problem, while being robust to sampling of missing ratings.

## References

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

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *SIGKDD*, pages 19–28. ACM, 2009.

[3] S. Agarwal. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *SDM*, pages 839–850. SIAM, 2011.

[4] S. Balakrishnan and S. Chopra. Collaborative ranking. In *ACM WSDM*, pages 143–152. ACM, 2012.

[5] I. Barjasteh, R. Forsati, F. Masrour, A.-H. Esfahanian, and H. Radha. Cold-start item and user recommendation with decoupled completion and transduction. In *Proceedings of ACM RecSys*, pages 91–98. ACM, 2015.

[6] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[7] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, 2010.

[8] K. Christakopoulou and A. Banerjee. Collaborative ranking with a push at the top. In *WWW*, pages 205–215. International World Wide Web Conferences Steering Committee, 2015.

[9] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys*, pages 39–46. ACM, 2010.

[10] R. Devooght, N. Kourtellis, and A. Mantrach. Dynamic matrix factorization with priors on unknown values. In *ACM SIGKDD*, pages 189–198. ACM, 2015.

[11] A. Elbadrawy and G. Karypis. User-specific feature-based similarity models for top-n recommendation of new items. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):33, 2015.

[12] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR*, pages 41–48. ACM, 2000.

[13] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[14] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96. ACM, 2014.

[15] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[16] B. M. Marlin and R. S. Zemel. Collaborative prediction and ranking with non-random missing data. In *RecSys*, pages 5–12. ACM, 2009.

[17] B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney. Collaborative filtering and the missing at random assumption. In *UAI*, pages 267–275, 2007.

[18] F. Masrour, I. Barjesteh, R. Forsati, A.-H. Esfahanian, and H. Radha. Network completion with node similarity: A matrix completion approach with provable guarantees. In *Proceedings of the 2015 IEEE/ACM ASONAM*, pages 302–307. ACM, 2015.

Table 3: Results on cold-start items. $\lambda$, $\mu_1$ and $\beta$ are regularization parameters, $h$ is dimension of latent features, $l$ is the number of similarity functions and $T$ is the number of iterations.

| | **Algorithms** | **Hyperparameters** | **DCG**@10 | **REC**@10 |
|---|---|---|---|---|
| **ML-IMDB** | CSR | — | 0.1282 | 0.0525 |
| | RLF | $h = 15$ | 0.0455 | 0.0155 |
| | CUFSM | $l = 1, \mu_1 = 0.005$ | 0.2160 | 0.0937 |
| | FBS | $\lambda = 0.01, \beta = 0.1, h = 5$ | 0.2270 | 0.0964 |
| | SCR | $\lambda = 0.6, h = 10, T = 200$ | **0.2731** | **0.2127** |
| **Amazon** | CSR | — | 0.0228 | 0.1205 |
| | RLF | $h = 30$ | 0.0076 | 0.0394 |
| | CUFSM | $l = 1, \mu_1 = 0.25$ | 0.0282 | 0.1376 |
| | FBS | $\lambda = 0.1, \beta = 1, h = 1$ | 0.0284 | 0.1392 |
| | SCR | $\lambda = 0.6, h = 10, T = 200$ | **0.1195** | **0.1683** |
| **CiteULike** | CSR | — | 0.0684 | 0.1791 |
| | RLF | $h = 75$ | 0.0424 | 0.0874 |
| | CUFSM | $l = 1, \mu_1 = 0.25$ | 0.0791 | 0.2017 |
| | FBS | $\lambda = 0.25, \beta = 10, h = 5$ | 0.0792 | 0.2026 |
| | SCR | $\lambda = 0.6, h = 10, T = 200$ | **0.0920** | **0.2243** |

[19] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys*, pages 21–28. ACM, 2009.

[20] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[21] C. Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *The Journal of Machine Learning Research*, 10:2233–2271, 2009.

[22] M. Saveski and A. Mantrach. Item cold-start recommendations: learning local collective embeddings. In *RecSys*, pages 89–96. ACM, 2014.

[23] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260. ACM, 2002.

[24] M. Sharma, J. Zhou, J. Hu, and G. Karypis. Feature-based factorized bilinear similarity model for cold-start top-n item recommendation. In *SDM*, 2015.

[25] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *ACM RecSys*, pages 269–272. ACM, 2010.

[26] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

[27] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *ICDM*, pages 1055–1060. IEEE, 2010.

[28] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2004.

[29] H. Steck. Training and testing of recommender systems on data missing not at random. In *KDD*, pages 713–722. ACM, 2010.

[30] H. Steck. Gaussian ranking by matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 115–122. ACM, 2015.

[31] M. Volkovs and R. S. Zemel. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2012.

[32] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. *NIPS*, 2007.

[33] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, volume 12, pages 403–414. SIAM, 2012.