symmetry

MDPI

Article

A Semi-Supervised Model for Top-N Recommendation

Shulong Chen * and Yuxing Peng

Science and Technology on Parallel and Distributed Laboratory, College of Computer, National University of Defense Technology, Changsha 410073, China; pengyuxing@nudt.edu.cn

* Correspondence: chenshulong@yeah.net

Received: 11 September 2018; Accepted: 8 October 2018; Published: 12 October 2018

Abstract: Top-N recommendation is an important recommendation technique that delivers a ranked top-N item list to each user. Data sparsity is a great challenge for top-N recommendation. In order to tackle this problem, in this paper, we propose a semi-supervised model called Semi-BPR (Semi-Supervised Bayesian Personalized Ranking). Our approach is based on the assumption that, for a given model, users always prefer items ranked higher in the generated recommendation list. Therefore, we select a certain number of items ranked higher in the recommendation list to construct an intermediate set and optimize the metric Area Under the Curve (AUC). In addition, we treat the intermediate set as a teaching set and design a semi-supervised self-training model. We conduct a series of experiments on three popular datasets to compare the proposed approach with several state-of-the-art baselines. The experimental results demonstrate that our approach significantly outperforms the other methods for all evaluation metrics, especially for sparse datasets.

Keywords: top-N recommendation; collaborative filtering; matrix factorization; semi-supervised learning; self-training

1. Introduction

Along with the rapid expansion of the Internet, recommender systems are becoming very popular tools to help users discover information that interests them and to alleviate information overload problems. In recent years, recommender systems have received more and more attention and have been widely applied in many areas, such as e-commerce, social network and online video sites.

Generally speaking, there are two main categories of recommendation tasks: rating prediction tasks and top-N recommendation tasks. The goal of rating prediction tasks is to predict users' ratings for unrated items. Early research mostly focused on this kind of task, for example a Netflix prize competition [1]. Models for rating prediction tasks mainly rely on users' explicit feedback (i.e., numerical ratings) to make predictions. However, users' explicit feedback is usually very scarce and hard to acquire. On the contrary, implicit feedback (such as purchase history, browsing history, etc.) is much easier to collect, and the amount is abundant. Top-N recommendation models only need implicit feedback to make recommendations. The objective of top-N recommendation is to generate a ranked item list for each user. The application scenario is very common in real life, for example product recommendations on Amazon [2], video recommendations on YouTube and friend recommendations on Facebook. Therefore, many recent studies have switched from using a rating prediction task to a top-N recommendation task. In this paper, we mainly consider the top-N recommendation issue.

For top-N recommendation with implicit feedback, the existing approaches can be further divided into two categories: pointwise methods and pairwise methods. Pointwise methods learn the model parameters by minimizing a pointwise loss function to approximate the absolute rating values. In contrast, based on the assumption that users prefer rated items to unrated items, pairwise methods

directly optimize the ranking-oriented metrics, such as the AUC (Area Under the Curve), the MAP (Mean Average Precision), the MRR (Mean Reciprocal Rank), etc. Empirically, the pairwise methods always achieve much better performance than the pointwise methods.

An important challenge is that top-N recommendation with implicit feedback faces a lack of negative samples. In fact, the missing ratings are a mixture of unobserved positive feedback and negative feedback and are difficult to distinguish. Most existing pairwise methods directly treat the missing ratings the same as negative feedback and assume that users prefer rated over unrated items. However, this neglects users' relative preference relations among the unrated items, which may cause performance degradation.

In order to exploit users' fine-grained preference over the unobserved feedback, we assumed that for a given recommendation model, users always prefer items ranked higher in the generated recommendation list. To test this assumption, we used the Bayesian Personalized Ranking (BPR) model to generate a recommendation list for each user in the Movielens 1M dataset. Figure 1 shows the average hit rate at different positions of the list. We observed that items ranked higher in the recommendation list had a higher hit rate. That is to say, users always preferred items ranked higher than items ranked lower in the recommendation list.

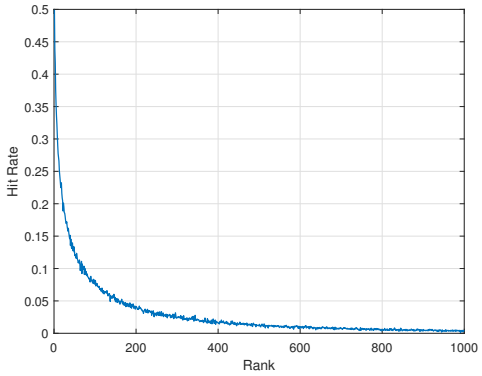


Figure 1. Average hit rate at different positions of the recommendation list.

For each user, we first randomly selected a certain number of unrated items to construct an intermediate set, and the remaining unrated items were treated as negative feedback. We assumed that users preferred items with positive feedback over items in the intermediate set and that they preferred items in the intermediate set over items with negative feedback. This assumption not only takes account of users' preferences over observed and unobserved items, but also users' preferences among the unobserved items. Based on this assumption, we directly optimized the AUC metric to place the item with positive feedback at the top, items belonging to the intermediate set in the middle and items with negative feedback at the bottom of the recommendation list. Then, we selected some items ranked higher in the newly generated recommendation list to update the intermediate set. It is possible to repeat the above cycle to iteratively improve the performance in a self-training paradigm.

The contributions of our work can be summarized as follows:

1. We made an assumption about users' relative preferences among unrated items and put forward an approach to build an intermediate set and optimize the AUC metric.
2. We used the intermediate set as a teaching set and designed a semi-supervised self-training model.
3. We conducted extensive experiments on three popular datasets, and the experimental results demonstrated the effectiveness of our approach.

The remainder of this paper is constructed as follows: We discuss the related work in Section 2. Section 3 describes our approach in detail. In Section 4, we analyze the experimental results. Finally, Section 5 concludes the paper.

2. Related Work

In this paper, we focus primarily on top-N recommendation with implicit feedback, and the key innovation of our approach is a semi-supervised self-training paradigm. Hence, we discuss the related work about the top-N recommendation and semi-supervised recommendation methods separately.

2.1. Top-N Recommendation

In top-N recommendation tasks, the objective is to recommend a list of products to each user that is most favorable. There are two major categories of top-N recommendation methods: pointwise methods and pairwise methods.

Pointwise methods learn the model parameters by minimizing a pointwise loss function to fit users' absolute rating values. Pan et al. [3] formulated the One-Class Collaborative Filtering (OCCF) problem and presented two methods to solve this problem: one is based on negative example weighting and the other on negative example exampling. Hu et al. [4] treated users' rating data as indications of positive and negative preferences, which are associated with different confidence values. They also put forward a scalable optimization procedure tailored to implicit feedback recommenders. Ning et al. [5] presented the Sparse Linear Method (SLIM) for top-N recommendation, which learns a sparse aggregation coefficient matrix for items by solving a regularized optimization problem. To improve the effectiveness of top-N recommendation, Kabbur et al. [6] presented Factored Item Similarity Models (FISM) for top-N recommendation. In their model, the item-item similarity matrix is decomposed into the product of two low rank latent factor matrices.

Based on the assumption that users prefer the rated items to the unrated items, pairwise methods attempt to directly optimize ranking-oriented metrics. The work in [7] is a seminal work for pairwise recommendation. A general OPTimization criterion for Bayesian Personalized Ranking (BPR-OPT) is proposed in this paper. By introducing richer interactions among users, Pan et al. [8] put forward the Group Bayesian Personalized Ranking (GBPR) model. Shi et al. [9] presented the Collaborative Less is More Filtering (CLiMF) model by directly maximizing the metric MRR. The work in [10] proposed the Tensor Factorization for MAP Maximization (TFMAP) model as a context-aware top-N recommendation model, by maximizing the MAP metric. Empirically, these pairwise methods can achieve much better performances than pointwise methods. However, these methods treat all of the unobserved feedback as negative feedback, which neglects users' relative preferences among the unobserved items. In addition, this separation causes an extreme imbalance between positive and negative samples.

Recently, a few studies have tried to solve this problem. Based on the assumption that users tend to prefer items that their friends select, Zhao et al. [11] developed a model called SBPR (Social Bayesian Personalized Ranking). Song et al. [12] put forward a Generalized AUC (GAUC) metric to quantify the ranking performance in a signed social network. Based on the idea that users tend to prefer items selected by their neighbors, Liu et al. [13] proposed a top-N recommendation algorithm called Collaborative Pairwise Learning to Rank (CPLR). Lu Yu et al. [14] tried to incorporate multiple types of user-item relationships into a unified pairwise ranking model to optimize approximately the MAP and MRR ranking metrics. Although these few methods exploit the user's preference among the unobserved items, they are all based on heuristic rules, and their performance is not satisfactory. Compared with these methods, our approach is more flexible and does not require any auxiliary information.

2.2. Semi-Supervised Recommendation

Semi-supervised learning [15] has been investigated thoroughly in the traditional data mining fields, such as classification and regression. However, little research has been conducted in the field of recommender systems.

Zhang et al. [16] proposed a semi-supervised ensemble recommendation model. By employing the co-training strategy, this model allows two weak prediction models to learn from each other. To further improve its performance, The work in [17] designed a tri-training framework to incorporate more recommender models. The work in [18] presented a background-based semi-supervised tri-training model. These semi-supervised recommendation methods are all designed for item rating prediction tasks, but are inappropriate for top-N recommendation tasks.

The work in [19] proposed a semi-supervised multi-view ranking algorithm for document ranking, which takes advantage of the global consistency between view-specific ranking functions on unlabeled samples. This algorithm can enhance the document ranking performance, but it relies on multiple view information, which is usually not available in the recommender system field. The work in [20] presented a semi-supervised model for bipartite ranking based on the self-training paradigm. Although this model and our model are both based on the self-training paradigm, they are actually quite different. First, the model of [20] is designed for bipartite ranking tasks, where both positive and negative samples are available. However, for top-N recommendation tasks with implicit feedback, there are only positive samples. Second, the ranking cost functions are quite different. In our model, we directly optimize the AUC metric, which can allow a higher recommendation performance to be obtained.

3. Our Approach

In this section, we first formulate the top-N recommendation with implicit feedback problem and then introduce the motivation of our approach. After that, we describe the design of our approach in detail. Finally, we present the model learning method and analyze its computational complexity.

3.1. Problem Definition

Some symbols that are frequently used in the rest of the paper are summarized in Table 1. The top-N recommendation problem can be described as follows: given the user-item implicit feedback matrix $R_{m\times n}$ from m users and n items, the goal is to learn a scoring function, $f_u : I/I_u \rightarrow \mathbb{R}$, for each user ($u \in U$), and to generate a ranking list that is sorted by the scoring values in descending order.

Table 1. Summary of symbols used in this paper.

Symbol	Description
U	User set
I	Item set
$m = \|U\|$	User number
$n = \|I\|$	Item number
$u \in U$	Used to index a user
$i, j, k \in I$	Used to index an item
U_i	Users that have rated item i
I_u	Items that user u has rated
$R_{m\times n}$	Rating matrix. $R_{u,i} = 1$, if the feedback is observed; otherwise, $R_{u,i} = 0$.
f_u	The scoring function of user u. $f_u(i)$ refers to the predicted scoring value of item i.

3.2. Overview

If a user's feedback for an item is observed, we can infer that he/she is interested in the item to a large extent. Thus, we also assume that each user prefers items with positive feedback to items with unobserved feedback.

$$(u, i) \succ (u, j),\ i \in I_u, j \in I/I_u \tag{1}$$

where (u, i) and (u, j) refer to the user's preferences for item i and item j, respectively. The symbol $\succ$ represents the user's relative preference.

In many cases, the user-item rating matrix is very sparse. If a user has not rated an item, he may not like this item. Therefore, treating all the unrated items as negative samples has a certain degree of rationality. However, this separation neglects users' relative preferences over unrated items.

In the recommendation list of user u, if item j is ranked higher than item k, we think the probability that u likes item j is greater than that of item k. If their ranking difference is large enough, we can almost infer user u prefers item j to item k. This can be represented as the following equation:

$$(u,j) \succ (u,k),\ j,k \in I/I_u, rank_u(j) < rank_u(k) \tag{2}$$

where $rank_u(j)$ and $rank_u(k)$ refer to the ranking positions of item j and item k in the recommendation list of user u, respectively.

According to the generated recommendation list, we can divide the unobserved items of user u into two subsets: intermediate item set T_u and negative item set N_u. Specifically, we select the top $0 < r < 1$ proportion of the items in the recommendation list to construct an intermediate item set, and the remaining unobserved items construct a negative item set. Then, it is assumed that each user prefers items in the intermediate item set to items in the negative item set.

$$(u,j) \succ (u,k),\ j \in T_u, k \in N_u \tag{3}$$

where $T_u \cap N_u = \emptyset$, $T_u \cup N_u = I/I_u$.

According to Equation (1) and Equation (3), we can get the following equation:

$$(u,i) \succ (u,j),\ (u,i) \succ (u,k),\ (u,j) \succ (u,k),\ i \in I_u, j \in T_u, k \in N_u. \tag{4}$$

Based on the transitive property of the user's relative preference ($\succ$), Equation (4) can be simplified to:

$$(u,i) \succ (u,j),\ (u,j) \succ (u,k),\ i \in I_u, j \in T_u, k \in N_u. \tag{5}$$

According to the above formula, our model assumption can be summarized as follows: each user (u) prefers his/her rated items (I_u) over items in the intermediate set (T_u) and prefers items in the intermediated set (T_u) to items in the negative item set (N_u).

From the model's assumptions, we can see that the division of the intermediated item set (T) and the negative item set (N) is critical in our model. However, for a given model, the recommendation list it generates may not be accurate. Division, according to this inaccurate recommendation list, may introduce errors. In order to solve this problem, a self-training paradigm was adopted to improve the model iteratively. An overview of our approach is shown in Figure 2.

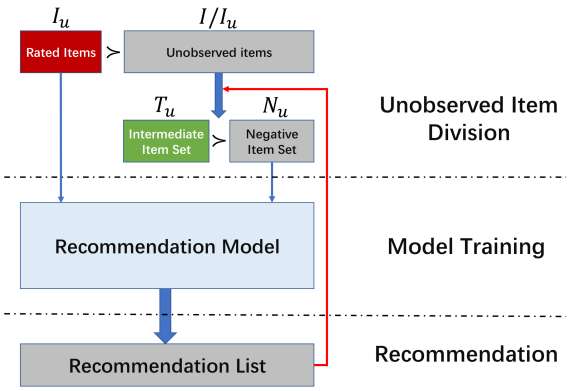


Figure 2. Overview of our approach.

In order to illustrate our idea clearly, we performed a case study on a particular user (u), as shown in Figure 3. Suppose there are $n = 9$ items, namely $i_1 \sim i_9$, and user u has the rated items

i_1 and i_4. The goal is to recommend $topN = 2$ unrated items that are most favorable to user u. Based on the assumption that user u prefers the rated items ($I_u = \{i_1, i_4\}$) to the unrated items ($I/I_u = \{i_2, i_3, i_5, i_6, i_7, i_8, i_9\}$), a ranked item list (Rank List 1) can be formed by the Matrix Factorization (MF) model. According to Rank List 1, the unrated items of u can be divided into two subsets: the intermediate item set, $T_u = \{i_7, i_5, i_8\}$, and the negative item set, $N_u = \{i_2, i_9, i_6, i_3\}$. Based on our assumption in Equation (5), user u prefers the items in I_u to the items in T_u and prefers the items in T_u to the items in N_u. By optimizing the AUC metric to put items in $I_u = \{i_1, i_4\}$ at the top, items in $T_u = \{i_7, i_5, i_8\}$ in the middle and items in $N_u = \{i_2, i_9, i_6, i_3\}$ at the bottom of the ranked item list, we can get a new ranked item list: Rank List 2. Then, according to Rank List 2, the unrated items (I/I_u) are divided into T_u and N_u. The above process is repeated until the ranked item list does not change. Finally, the top two unrated items in the ranked items list are selected to recommend to user u.

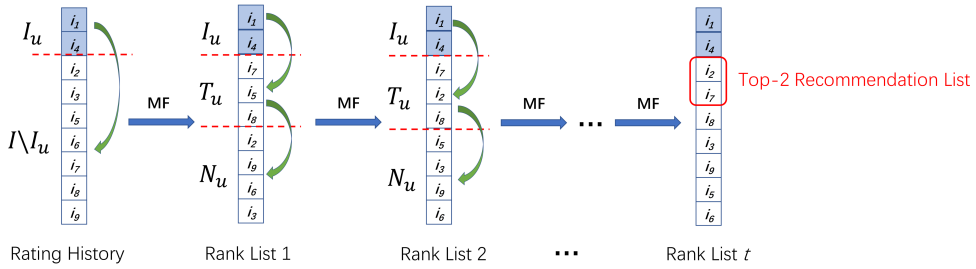


Figure 3. A case study on a user. MF, Matrix Factorization.

3.3. Objective Function

For a given user u, the likelihood of the user's pairwise preference can be presented by the following equation:

$$\begin{aligned}\prod_{(i,j,k)\in I\times I\times I} & P(f_u(i) > f_u(j) > f_u(k))^{\delta((u,i)\succ(u,j)\succ(u,k))} \\ & \times [1 - P(f_u(i) > f_u(j) > f_u(k))]^{1-\delta((u,i)\succ(u,j)\succ(u,k))} \\ = \prod_{i\in I_u}\prod_{j\in T_u}\prod_{k\in N_u} & P(f_u(i) > f_u(j))P(f_u(j) > f_u(k)) \\ & \times [1 - P(f_u(i) < f_u(j))][1 - P(f_u(j) < f_u(k))]\end{aligned} \tag{6}$$

where $\delta(x)$ is the indicator function of the Boolean variable x.

$$\delta(x) = \begin{cases} 1, \ if \ x \ is \ true \\ 0, \ else \end{cases} \tag{7}$$

It is assumed that the preferences of different users are independent. Therefore, the overall likelihood of all users can be presented as follows:

$$\begin{aligned}\prod_{u\in U}\prod_{i\in I_u}\prod_{j\in T_u}\prod_{k\in N_u} & P(f_u(i) > f_u(j))P(f_u(j) > f_u(k)) \\ & \times [1 - P(f_u(i) < f_u(j))][1 - P(f_u(j) < f_u(k))].\end{aligned} \tag{8}$$

Like the BPR model [7], the function $\sigma(f_u(i) - f_u(j)) = \frac{1}{1+e^{-(f_u(i)-f_u(j))}}$ is used to approximate probability $P(f_u(i) > f_u(j))$. Based on the properties of the sigmoid function ($\sigma(\cdot)$), we have:

$$\begin{aligned} &P(f_u(i) > f_u(j))P(f_u(j) > f_u(k))[1 - P(f_u(i) < f_u(j))][1 - P(f_u(j) < f_u(k))] \\ &= \sigma(f_u(i) - f_u(j))\sigma(f_u(j) - f_u(k))[1 - \sigma(f_u(j) - f_u(i))][1 - \sigma(f_u(k) - f_u(j))] \\ &= [\sigma(f_u(i) - f_u(j))]^2 \cdot [\sigma(f_u(j) - f_u(k))]^2. \end{aligned} \tag{9}$$

Based on Equations (8) and (9), the log of the overall likelihood can be obtained, as shown below:

$$\sum_{u \in U} \sum_{i \in I_u} \sum_{j \in T_u} \sum_{k \in N_u} 2 \cdot [ln\sigma(f_u(i) - f_u(j)) + ln\sigma(f_u(j) - f_u(k))]. \tag{10}$$

Maximizing the log-likelihood is equivalent to maximizing the following object function:

$$\frac{1}{2} \sum_{u \in U} \sum_{i \in I_u} \sum_{j \in T_u} \sum_{k \in N_u} [ln\sigma(f_u(i) - f_u(j)) + ln\sigma(f_u(j) - f_u(k))] - \frac{\lambda}{2} \|\Theta\|_F^2 \tag{11}$$

where Θ are the model parameters and $-\frac{\lambda}{2}\|\Theta\|_F^2$ is the regularization term to avoid overfitting.

AUC is an important metric for measuring the recommendation performance. However, the standard AUC metric only considers binary cases and is not suitable for our situation where there are three kinds of items. Similar to [11] and [12], we defined the AUC of user u as follows:

$$AUC(u) = \frac{1}{|I_u| \cdot |T_u| \cdot |N_u|} \sum_{i \in I_u} \sum_{j \in T_u} \sum_{k \in N_u} \frac{\delta(f_u(i) - f_u(j) > 0) + \delta(f_u(j) - f_u(k) > 0)}{2}. \tag{12}$$

By comparing Equations (11) and (12), the log-likelihood and the AUC metric are shown to be very similar. If the normalization term $\frac{1}{|I_u| \cdot |T_u| \cdot |N_u|}$ of Equation (12) is neglected, the only difference is the loss functions $ln\sigma(x)$ and $\delta(x > 0)$. Because the function $\delta(x > 0)$ is non-differential and difficult to optimize, the log-likelihood can be regarded as an approximation of the AUC metric, by replacing $\delta(x > 0)$ with the differentiable function $ln\sigma(x)$.

However, there is a problem with the objective function. According to Equation (11), each sample pair has the same weight and contribution to the model in the training phase. In fact, users' relative preferences between different sample pairs are quite different. Therefore, we introduced a coefficient to control the weights of each sample pair. The ultimate objective function of our model is defined as the following:

$$\mathcal{L} = \frac{1}{2} \sum_{u \in U} \sum_{i \in I_u} \sum_{j \in T_u} \sum_{k \in N_u} [a \cdot ln\sigma(f_u(i) - f_u(j)) + (1 - a) \cdot ln\sigma(f_u(j) - f_u(k))] - \frac{\lambda}{2} \|\Theta\|_F^2 \tag{13}$$

where $a \in [0, 1]$ is the weight of a user's relative preference between item $i \in I_u$ and item $j \in T_u$ and $1 - a$ is the weight of a user's relative preference between item $j \in T_u$ and item $k \in N_u$. The larger the probability of item $j \in T_u$ being a positive sample, the smaller coefficient a is.

From Figure 1, it can be observed that items ranked higher in the recommendation list have a higher hit rate, which means items ranked higher are more likely to be a positive sample. Thus, coefficient a can be defined as follows:

$$a = \gamma + \beta(x_u^j)^\alpha \tag{14}$$

where $x_u^j = \frac{rank_u(j)}{|T_u|}$ is the relative ranking position of item j in the intermediate item set T_u. α, β, γ are adjustable parameters that satisfy $\alpha > 0$, $0 \leq \beta, \gamma \leq 1$ and $\beta + \gamma \leq 1$.

From Equations (13) and (14), we can find that the standard BPR model can be regarded as a special case of our model when we set $r = 1$, $\alpha = \beta = 0$, and $\gamma = 1$.

3.4. Model Learning

We adopted the Stochastic Gradient Descent (SGD) algorithm to learn the model's parameters. Specifically, we select a user ($u \in U$) randomly and then selected items $i \in I_u$, $j \in T_u$, $k \in N_u$ randomly. The stochastic gradient of the objective function $\mathcal{L}$ with respect to the model parameters Θ is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \Theta} &= \frac{a}{2} \cdot \frac{\partial ln\sigma(f_u(i,j))}{\partial \Theta} + \frac{(1-a)}{2} \cdot \frac{\partial ln\sigma(f_u(j,k))}{\partial \Theta} - \frac{\lambda}{2} \frac{\|\Theta\|_F^2}{\partial \Theta} \\ &= \frac{a}{2} \cdot \frac{1}{1+e^{f_u(i,j)}} \cdot \frac{\partial f_u(i,j)}{\partial \Theta} + \frac{(1-a)}{2} \cdot \frac{1}{1+e^{f_u(j,k)}} \cdot \frac{\partial f_u(j,k)}{\partial \Theta} - \lambda \Theta \end{aligned} \tag{15}$$

where $f_u(i,j) = f_u(i) - f_u(j)$, $f_u(j,k) = f_u(j) - f_u(k)$. Then, we can update the model's parameters Θ by walking a step along the ascending gradient direction.

$$\Theta^{t+1} = \Theta^t + \eta \cdot \frac{\partial \mathcal{L}}{\partial \Theta} \tag{16}$$

where η is the learning rate. The pseudocode of our model is shown in Algorithm 1.

Algorithm 1: Semi-supervised Bayesian personalized ranking.

Input: the implicit feedback matrix ($R_{m \times n}$), the proportion parameter r, parameters α, β and γ that control the weight of sample pairs, the number of sample pairs (S)
Output: the model parameters Θ
Initialization: initialize model parameter Θ with a random variable of a Gaussian distribution.
for $u \in U$ **do**
  Randomly select r proportion of the user's unrated items (I/I_u) to construct the intermediate item set (T_u^0); the remaining unrated items are used to construct the negative item set (N_u^0)
end
Self-training
for $t < rounds$ **do**
  for $s = 0; s < S; s$++ **do**
    Uniformly sample a user, $u \in U$;
    Uniformly sample item $i \in I_u$;
    Uniformly sample item $j \in T_u^{t-1}$;
    Uniformly sample item $k \in N_u^{t-1}$;
    Compute the stochastic gradient, $\frac{\partial \mathcal{L}}{\partial \Theta}$, according to Equation (15);
    Update the model's parameters (Θ) according to Equation (16);
  end
  for $u \in U$ **do**
    for $i \in I/I_u$ **do**
      Compute the ranking score, $f_u^t(i)$;
    end
    List the user's unrated items in descending order by their ranking scores;
    Select the top r proportion of the ranked item list as intermediate item set (T_u^t) and the remaining unobserved items as the negative item set (N_u^t);
  end
end
Return: Θ

3.5. Matrix Factorization Model with Semi-BPR

The Semi-BPR is a general framework that can be applied to many recommendation models to improve their performance. We adopted the matrix factorization algorithm [21] as the given model, and the generated model is called Semi-BPR-MF. Matrix factorization is one of the popular recommendation models. The scoring function is defined as:

$$f_u(i) = b_i + P_u^T Q_i \tag{17}$$

where $b \in \mathbb{R}^n$ is the item's bias vector. $P \in \mathbb{R}^{d \times m}$ is the user latent factor matrix, and $Q \in \mathbb{R}^{d \times n}$ is the item's latent factor matrix. d is the number of latent dimensions.

According to Equation (15), the stochastic gradients of the objective function $\mathcal{L}$ with respect to the model parameters are:

$$\frac{\partial \mathcal{L}}{\partial P_u} = \frac{a}{2} \cdot \frac{Q_i - Q_j}{1 + e^{f_u(i,j)}} + \frac{1-a}{2} \cdot \frac{Q_j - Q_k}{1 + e^{f_u(j,k)}} - \lambda_P P_u, \tag{18}$$

$$\frac{\partial \mathcal{L}}{\partial Q_i} = \frac{a}{2} \cdot \frac{P_u}{1 + e^{f_u(i,j)}} - \lambda_Q Q_i, \tag{19}$$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \frac{a}{2} \cdot \frac{1}{1 + e^{f_u(i,j)}} - \lambda_b b_i, \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial Q_j} = -\frac{a}{2} \cdot \frac{P_u}{1 + e^{f_u(i,j)}} + \frac{1-a}{2} \cdot \frac{P_u}{1 + e^{f_u(j,k)}} - \lambda_Q Q_j, \tag{21}$$

$$\frac{\partial \mathcal{L}}{\partial b_j} = -\frac{a}{2} \cdot \frac{1}{1 + e^{f_u(i,j)}} + \frac{1-a}{2} \cdot \frac{1}{1 + e^{f_u(j,k)}} - \lambda_b b_j, \tag{22}$$

$$\frac{\partial \mathcal{L}}{\partial Q_k} = -\frac{1-a}{2} \cdot \frac{P_u}{1 + e^{f_u(j,k)}} - \lambda_Q Q_k, \tag{23}$$

$$\frac{\partial \mathcal{L}}{\partial b_k} = -\frac{1-a}{2} \cdot \frac{1}{1 + e^{f_u(j,k)}} - \lambda_b b_k, \tag{24}$$

where $f_u(i,j) = (b_i + P_u^T Q_i) - (b_j + P_u^T Q_j)$ and $f_u(j,k) = (b_j + P_u^T Q_j) - (b_k + P_u^T Q_k)$. λ_P, λ_Q and λ_b are regularization parameters for the user's latent factor matrix (P), the item's latent factor matrix (Q) and the item's bias vector (b), respectively.

3.6. Complexity Analysis

Algorithm 1 shows that each round of our self-training algorithm mainly consists of two steps: a training step and a recommendation step. In the training step, the most time-consuming operations are computing the prediction values and gradients of the objective function. For the Semi-BPR-MF model, according to Equation (17), the time complexity required to compute the prediction value $f_u(i)$ is $O(d+1)$. For each training pair (u,i,j,k), according to Equations (18)∼(24), the time complexity required to compute the gradients is $O(6d+4)$. Suppose the total number of training pairs is S. Then, the time complexity of the training step is $O(d \cdot S)$.

In the recommendation step, we first need to compute each user's prediction scores for his/her unrated items, and the time complexity is $O(d \cdot |I/I_u|)$. The complexity of ranking each user's unrated items is $O(|I/I_u| log |I/I_u|)$. Because in most cases, the users' feedback matrix is very sparse, I/I_u is approximately equal to $|I| = n$. The time complexity required to generate a ranked item list for each user is $O(nd + nlogn)$. Therefore, the total time complexity of the recommendation step is $O(mnd + mnlogn)$.

In summary, the time complexity required for one round of our model is $O(d \cdot S + mn(d + logn))$. Though the computational complexity is relatively high for each round, our model converges rapidly after only several self-training rounds.

4. Experiments

This section describes the extensive experiments that were carried out to evaluate the proposed approach. We first introduce the experimental setup in Section 4.1. Next, we investigate the effects of different parameter settings on the performance in Section 4.2. Then, in Section 4.3, we compare our approach with several state-of-the-art baselines. Finally, we evaluate the scalability of our approach in Section 4.4.

4.1. Experimental Setup

4.1.1. Datasets

We evaluated the proposed approach on three popular datasets, the Movielens 1M, Lastfm 2K and Ciao datasets. Movielens 1M consists of 1,000,209 five-star ratings from 6040 users on 3076 movies. All users have 20 or more ratings in this dataset. The Lastfm 2K dataset contains users' music artist listening information, which includes 92,834 records from 1892 users on 17,632 artists. The Ciao dataset, which was collected from a product review website, includes 278,483 ratings of 7375 users on 99,746 products. Because the original Ciao dataset is very sparse, we prefiltered products with at least three ratings. In order to investigate top-N recommendation with implicit feedback, we first needed to convert the explicit ratings into a binary feedback matrix ($R_{m \times n}$). Specifically, we set $R_{u,i} = 1$ if implicit feedback from user u on item i was observed; otherwise, we set $R_{u,i} = 0$. Like [7], we removed the rating scores in the Movielens 1M and Ciao datasets. For the Lastfm 2K dataset, we binarized the feedback by setting all non-zero play counts to 1. The statistics of the experimental datasets are summarized in Table 2.

Table 2. Dataset statistics.

Dataset	User#	Item#	Feedback#	Density
Movielens 1M	6040	3706	1,000,209	4.47%
Lastfm 2K	1892	17,632	92,834	0.28%
Ciao	7267	11,211	147,987	0.18%

4.1.2. Evaluation Metrics

To measure the performance of the top N recommendation method, we adopted six standard ranking-oriented metrics: *Precision@k* (*Pre@k*), *Recall@k* (*Rec@k*), *MAP@k*, *MRR@k*, *AUC@k* and Normalized Discounted Cumulative Gain (*NDCG@k*), where k refers to the number of recommended items for each user. For all evaluation metrics, we first computed the recommendation performance for each user and then obtained the average performance over all users. Suppose, for each user u, the rated items in the testing set are denoted B_u, and the top k items in the recommended list by a given model are L_u^k. The evaluation metrics are defined as follows:

$$Pre@k = \frac{1}{|U|} \sum_{u \in U} \frac{|L_u^k \cap B_u|}{k}, \tag{25}$$

$$Rec@k = \frac{1}{|U|} \sum_{u \in U} \frac{|L_u^k \cap B_u|}{|B_u|}, \tag{26}$$

$$MAP@k = \frac{1}{|U|} \sum_{u \in U} \frac{1}{min(k, |B_u|)} \sum_{i=1}^{k} rel_i^u \frac{|L_u^i \cap B_u|}{i}, \tag{27}$$

$$MRR@k = \frac{1}{|U|}\sum_{u\in U}\frac{1}{min^{k}_{i\in B_u}(p_{ui})}, \tag{28}$$

$$AUC@k = \frac{1}{|U|}\sum_{u\in U}\frac{1}{|B_u \cap L^k_u||L^k_u/(B_u \cap L^k_u)|}\sum_{i\in B_u\cap L^k_u}\sum_{j\in L^k_u/(B_u\cap L^k_u)}\delta(f_u(i) > f_u(j)), \tag{29}$$

$$NDCG@k = \frac{1}{|U|}\sum_{u\in U}\frac{1}{\sum_{i=1}^{min(k,|B_u|)}\frac{1}{log_2(i+1)}}\sum_{i=1}^{k}\frac{2^{rel^u_i}-1}{log_2(i+1)}, \tag{30}$$

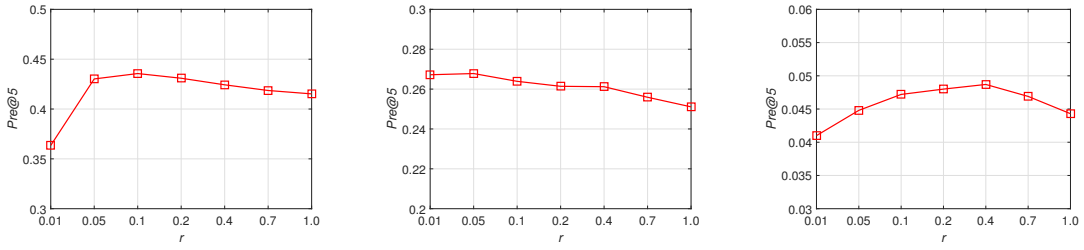
where rel^u_i indicates the preference of user u for the item at position i in the recommended list. p_{ui} is the ranking position of item i in the recommended list of user u and $min^k_{i\in B_u}(p_{ui})$ is the position of the first relevant item in L^k_u.

In our experiments, we conducted a 5-fold cross-validation. Specifically, each experimental dataset was divided into 5 folds randomly. We used four folds as the training set and used the remaining one as the testing set. We repeated this process five times and reported the average performance. All experiments were carried out on the same machine with an Intel Core i5-6300HQ CPU (2.3 GHz, Quad Core) and 16 G RAM. The implementation of our approach was based on an open source JAVA library: Librec [22].

4.2. Impacts of Parameters

The purpose of this experiment was to investigate the impacts of different parameter settings on the performance of our approach.

A key step in our approach is splitting users' unobserved feedback into two subsets: the intermediate item set (T) and the negative item set (N). The parameter r plays a very important role in the split, and it controls the proportion of items in the intermediate set. To study the impacts of different r values on the recommendation performance, we varied r from 0 to 1.0. For each given r value, we adjusted the other parameters (α, β and γ) to achieve the best performance. In this experiment, the number of latent dimensions d was fixed at 100. Figure 4 shows the impact of r on the performance.



(**a**) (**b**) (**c**)

Figure 4. Impact of parameter r on the recommendation performance. (**a**) Impact of r on Movielens 1M dataset; (**b**) Impact of r on Lastfm 2K dataset; (**c**) Impact of r on Ciao dataset.

In the Movielens 1M and Ciao datasets, the metric *Pre@5* gradually improves with the growth of parameter r, and the best performances are achieved when r approximates 0.1 for the Movielens 1M dataset and 0.4 for the Ciao dataset. If r continues to increase, the performance experiences a decrease. In the Lastfm 2K dataset, the peak performance is obtained when r is around 0.05, and the performance gradually deteriorates as r increases. In extreme cases, when $r = 1.0$, which means we add all of the users' unrated items into the intermediate set, our Semi-BPR model reduces to the standard BPR model.

Another important parameter in the Semi-BPR-MF model is the dimension of latent factors d. It controls the capability of the matrix factorization algorithm and, thus, may have an important effect on the recommendation performance. In this experiment, we chose the dimension of latent factors from {5, 10, 20, 40, 60, 80, 100}. For each given dimension d, we tuned the other parameters to achieve the optimal performance.

Figure 5 shows the recommendation quality with different dimensions of latent factors. It can be observed that the performances of BPR-MF and Semi-BPR-MF gradually improved with the increase of dimension d in all datasets. In addition, we can see that our Semi-BPR-MF model consistently outperformed the BPR-MF model in all cases, especially for the spare Lastfm 2K and Ciao datasets. Setting dimension d to 100 was shown to be the best choice for all datasets to balance the recommendation performance and computational complexity.

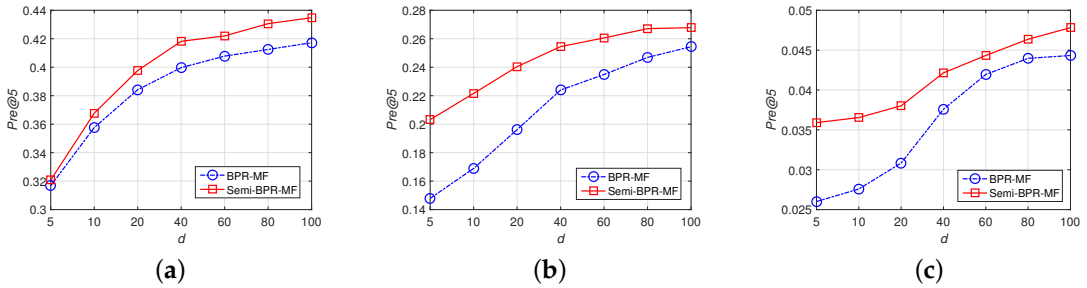


Figure 5. Recommendation performance with different dimensions d. (**a**) Recommendation performance variation on Movielens 1M dataset; (**b**) Recommendation performance variation on Lastfm 2K dataset; (**c**) Recommendation performance variation on Ciao dataset.

4.3. Performance Comparison

4.3.1. Baselines

In order to demonstrate the effectiveness of the proposed approach, we compared our model with several state-of-the-art baselines.

Most Popular (MostPop): This is a basic recommendation model, which ranks items according to their popularity and recommends popular items to each user. MostPop does not consider users' preferences and, thus, cannot provide personalized recommendations.

Neighborhood approaches: User-based K-Nearest Neighbor (UserKNN) and Item-based K-Nearest Neighbor (ItemKNN) are two typical neighborhood-based models. UserKNN recommends items to each user that have been rated by similar users, while ItemKNN recommends items to each user that are similar to his/her rated items. For neighborhood approaches, the binary-cosine was adopted as the similarity measure.

Pointwise approaches: The Weighted Regularized Matrix Factorization (WRMF) is a state-of-the-art pointwise recommendation model [4]. WRMF treats the rating data as indications of positive and negative feedback associated with different confidence levels and learns model parameters by fitting the rating data.

Pairwise approaches: BPR is a generic optimization criterion for personalized recommendations. Matrix Factorization with BPR optimizations (BPR-MF) and K-Nearest Neighbor with BPR optimizations (BPR-KNN) are two representative pairwise recommendation models [7]. Our Semi-BPR-MF model is based on the BPR-MF model and also belongs to this category. For the pairwise approaches, 100 pairs were randomly sampled for each user in the training phase.

4.3.2. Parameter Settings

For each compared approach, we determined the optimal parameter settings through a grid search. The MostPop model has no parameters to be set. For the UserKNN model and ItemKNN

model, the number of neighbors knn was chosen from $\{40, 60, 80, 100, 120, 160, 200\}$. For the WRMF model, the weight (α) was chosen from $\{0, 1, 5, 10\}$. For our Semi-BPR-MF model, we chose the proportion of the intermediate item set r from $\{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and the weight coefficient parameters (α, β, and γ) from $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. For all methods, the dimension of latent factors d was chosen from $\{10, 20, 50, 100\}$, and the regularization coefficient (λ) was chosen from $\{0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}$. The optimal parameter settings are shown in Table 3.

Table 3. Parameter settings of different methods.

Method	Movielens 1M	Lastfm 2K	Ciao
MostPop	-	-	-
UserKNN	$knn = 200$	$knn = 100$	$knn = 100$
ItemKNN	$knn = 120$	$knn = 80$	$knn = 100$
WRMF	$d = 50, \alpha = 1,$ $\lambda_U = \lambda_I = 5$	$d = 50, \alpha = 1,$ $\lambda_U = \lambda_I = 10$	$d = 50, \alpha = 5,$ $\lambda_U = \lambda_I = 100,$
BPR-MF	$d = 100,$ $\lambda_U = \lambda_I = 0.01, \lambda_b = 0.005$	$d = 100,$ $\lambda_U = \lambda_I = \lambda_b = 0.005$	$d = 100,$ $\lambda_U = \lambda_I = \lambda_b = 0.02$
BPR-KNN	$\lambda = 0.01$	$\lambda = 0.01$	$\lambda = 10$
Semi-BPR-MF	$d = 100, r = 0.1,$ $\lambda_U = \lambda_I = \lambda_b = 0.02,$ $\alpha = 0.5, \beta = 0.2, \gamma = 0.4$	$d = 100, r = 0.05,$ $\lambda_U = \lambda_I = \lambda_b = 0.01,$ $\alpha = 1, \beta = 0.1, \gamma = 0.1$	$d = 100, r = 0.4,$ $\lambda_U = \lambda_I = 0.02, \lambda_b = 0.01,$ $\alpha = 0.3, \beta = 0.4, \gamma = 0.6$

4.3.3. Recommendation Performance

The top-N recommendation performances of the different approaches are shown in Table 4. From the experimental results, we can make the following conclusions:

1. MostPop was the worst of all compared approaches, which implies that generating personalized recommendations for each user is very necessary.
2. UserKNN, ItemKNN approaches are popular in recommender systems. Their performance depends on the choice of a heuristic similarity measure. In most cases, neighborhood approaches were shown to be worse than pointwise or pairwise approaches.
3. WRMF is the state-of-the-art pointwise approach for top-N recommendation tasks. However, WRMF cannot directly optimize the ranking-oriented metrics and is slightly worse than the BPR-MF pairwise approach. This demonstrates that pairwise assumptions are more reasonable than pointwise assumptions.
4. In all three datasets, the proposed Semi-BPR-MF model outperformed the other baselines in all evaluation metrics. For sparse datasets, like the Ciao dataset, only considering user preference between the observed feedback and unobserved feedback can not achieve a satisfactory level of performance. Compared with the best baseline BPR-MF model, our approach can obtain a significant performance improvement.

Table 4. Recommendation performances of different models. The best results are highlighted in bold font and the best baseline results are underlined.

Dataset	Models	*Pre@5*	*Rec@5*	*MAP@5*	*MRR@5*	*AUC@5*	*NDCG@5*
Movielens 1M	MostPop	0.2088	0.0405	0.1499	0.3525	0.7626	0.2181
	UserKNN	0.3895	0.0982	0.3087	0.6088	0.8975	0.4118
	ItemKNN	0.3311	0.0772	0.2579	0.5390	0.8624	0.3514
	WRMF	0.4138	0.1059	0.3292	0.6330	0.9092	0.4359
	BPR-KNN	0.4018	0.1002	0.3223	0.6166	0.9005	0.4245
	BPR-MF	0.4172	0.1032	0.3382	0.6262	0.9055	0.4387
	Semi-BPR-MF	**0.4345**	**0.1080**	**0.3560**	**0.6464**	**0.9129**	**0.4571**
Lastfm 2K	MostPop	0.0857	0.0448	0.0567	0.1888	0.6458	0.0952
	UserKNN	0.2002	0.1044	0.1542	0.4233	0.7848	0.2300
	ItemKNN	0.2373	0.1234	0.1844	0.4897	0.8246	0.2714
	WRMF	0.2458	0.1278	0.1823	0.4807	0.8365	0.2727
	BPR-KNN	0.2459	0.1263	0.1836	0.4910	0.8381	0.2750
	BPR-MF	0.2544	0.1312	0.1886	0.4948	0.8469	0.2813
	Semi-BPR-MF	**0.2678**	**0.1387**	**0.2075**	**0.5230**	**0.8509**	**0.3007**
Ciao	MostPop	0.0281	0.0289	0.0251	0.0665	0.5533	0.0385
	UserKNN	0.0422	0.0421	0.0369	0.0963	0.5767	0.0562
	ItemKNN	0.0362	0.0325	0.0294	0.0772	0.5642	0.0453
	WRMF	0.0448	0.0433	0.0365	0.0972	0.5817	0.0573
	BPR-KNN	0.0425	0.0415	0.0351	0.0942	0.5790	0.0551
	BPR-MF	0.0443	0.0466	0.0383	0.1013	0.5847	0.0594
	Semi-BPR-MF	**0.0478**	**0.0506**	**0.0415**	**0.1095**	**0.5908**	**0.0642**

In the above experiments, the number of top-N items (N) was fixed at five. To compare the recommendation performances with different values of N, we varied N in the range $[1, 100]$, and the results are shown in Figure 6. We found that our Semi-BPR-MF model performed best in all compared methods.

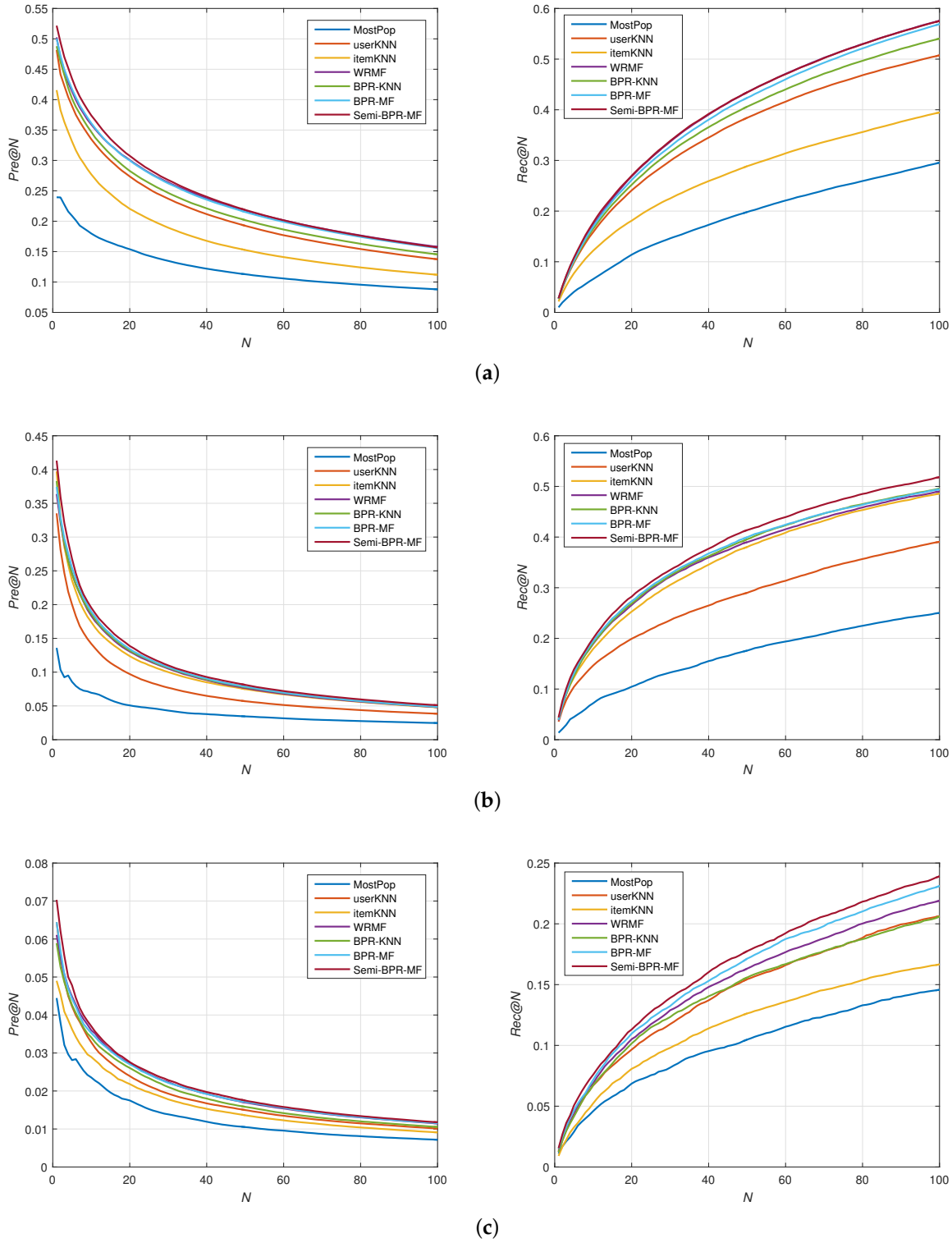


Figure 6. Performance comparison for different values of N. (**a**) Performance comparison on Movielens 1M dataset; (**b**) Performance comparison on Lastfm 2K dataset; (**c**) Performance comparison on Ciao dataset.

4.4. Scalability

In this experiment, we aimed to evaluate the scalability of the proposed approach. The dimension of latent factors d is a key factor that affects the computation complexity of our approach. As analyzed

earlier, the computational complexity of Semi-BPR-MF is $O(d \cdot S + mn(d + logn))$. Figure 7 illustrates the runtime per self-training round on the Movielens 1M, Lastfm 2K and Ciao datasets. We observed that the runtime increased almost linearly with an increase in dimension d.

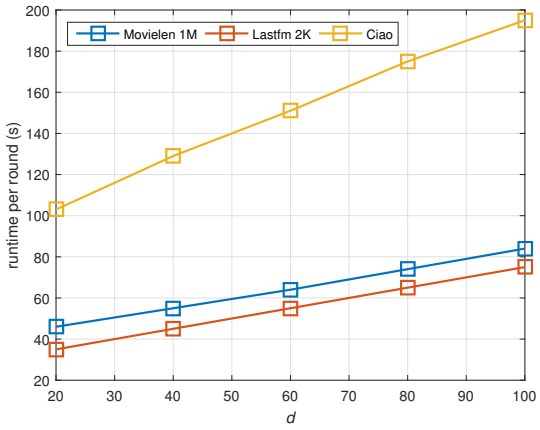


Figure 7. Runtime per self-training round.

Figure 8 shows the convergence of our approach on all datasets. We found that our approach converged after about five self-training rounds and then fluctuated in a small range around the best performance.

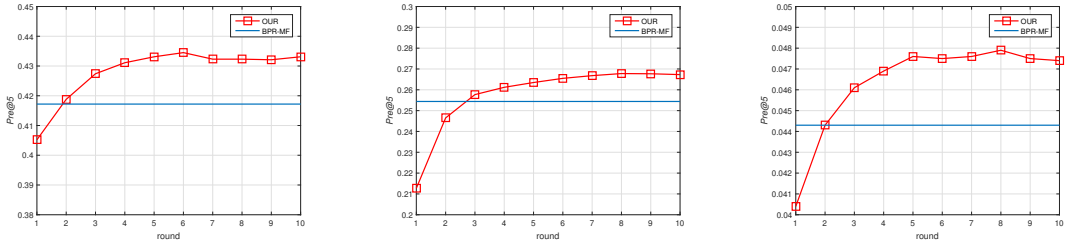


Figure 8. Convergence of our approach. (**a**) Convergence on Movielens 1M dataset; (**b**) Convergence on Lastfm 2K dataset; (**c**) Convergence on Ciao dataset.

5. Conclusions

In this paper, we proposed a semi-supervised model for top-N recommendation tasks. Based on the assumption that users always prefer items that are ranked higher in the recommendation list generated by a given model, we selected a certain number of items ranked higher in the recommendation list to construct an intermediate set and optimize the AUC metric. Our approach adopts the self-training paradigm to improve the recommendation performance iteratively. We conducted extensive experiments on three popular datasets to evaluate the effectiveness of the proposed approach. For future work, there are still several issues to be studied. A promising research direction is the integration of auxiliary information (e.g., description information of items, users' reviews) into our model. In real environments, the user-item rating matrix is very sparse, and some new users or items only have very few rating data, that is the so-called cold-start problem. Therefore, it is worth studying the combination of the traditional recommendation model with semi-supervised learning to solve the cold-start problem.

Author Contributions: Y.P. proposed the research direction and gave the conceptualization. S.C. implemented the proposed approach, conducted experiments and wrote the paper.

Funding: This work was supported by The National Key Research and Development Program of China (2016 YFB1000100).

Conflicts of Interest: The authors declare that there is no conflict of interest regarding the publication of this article.

References

1. Bell, R.M.; Koren, Y. Lessons from the netflix prize challenge. *ACM SIGKDD Explor. Newsl.* **2007**, *9*, 75–79. [CrossRef]
2. Linden, G.; Smith, B.; York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Int. Comput.* **2003**, *7*, 76–80. [CrossRef]
3. Pan, R.; Zhou, Y.; Cao, B.; Liu, N.; Lukose, R.; Scholz, M.; Yang, Q. One-class collaborative filtering. In Proceedings of the 2008 8th IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 502–511.
4. Hu, Y.; Koren, Y.; Volinsky, C. Collaborative filtering for implicit feedback datasets. In Proceedings of the 2008 8th IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 263–272.
5. Ning, X.; Karypis, G. SLIM: Sparse linear methods for top-N recommender systems. In Proceedings of the 2011 11th IEEE International Conference on Data Mining, Vancouver, BC, Canada, 11–14 December 2011; pp. 497–506.
6. Kabbur, S.; Ning, X.; Karypis, G. FISM: Factored item similarity models for top-n recommender systems. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 659–667.
7. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the 23th Conference on Uncertainty in Artificial Intelligence (UAI 2009), Montreal, QC, Cananda, 18–21 June 2009; pp. 452–461.
8. Pan, W.; Chen, L. GBPR: Group preference based bayesian personalized ranking for one-class collaborative filtering. In Proceedings of the 23th International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013; pp. 2691–2697.
9. Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Oliver, N.; Hanjalic, A. CLiMF: Collaborative less-is-more filtering. In Proceedings of the 23th International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013; pp. 3077–3081.
10. Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Hanjalic, A.; Oliver, N. TFMAP: Optimizing map for top-n context-aware recommendation. In Proceedings of the 35th ACM Special Interest Group on Information Retrieval (SIGIR), Portland, OR, USA, 12–16 August 2012; pp. 155–164.
11. Zhao, T.; Mcauley, J.; King, I. Leveraging social connections to improve personalized ranking for collaborative filtering. In Proceedings of the 23th ACM International Conference on Conference on Information and Knowledge Management, Shanghai, China, 3–7 November 2014; pp. 261–270.
12. Song, D.; Meyer, D.A. Recommending positive links in signed social networks by optimizing a generalized AUC. In Proceedings of the 29th AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 290–296.
13. Liu, H.; Wu, Z.; Zhang, X. CPLR: Collaborative pairwise learning to rank for personalized recommendation. *Knowl.-Based Syst.* **2018**, *148*, 31–40. [CrossRef]
14. Yu, L.; Zhang, C.; Pei, S.; Sun, G.; Zhang, X. Walkranker: A unified pairwise ranking model with multiple relations for item recommendation. In Proceedings of the 32th AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
15. Hady, M.F.A.; Schwenker, F. Semi-supervised learning. *J. R. Stat. Soc.* **2006**, *172*, 530.
16. Zhang, M.; Tang, J.; Zhang, X.; Xue, X. Addressing cold start in recommender systems: a semi-supervised co-training algorithm. In Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, Gold Coast, Australia, 6–11 July 2014; pp. 73–82.
17. Zhang, X. Utilizing tri-training algorithm to solve cold start problem in recommender system. *Comput. Sci.* **2016**, *12*, 108–114.
18. Hao, Z.; Cheng, Y.; Cai, R.; Wen, W.; Wang, L. A semi-supervised solution for cold start issue on recommender systems. In Proceedings of the Asia-Pacific Web Conference, Guangzhou, China, 18–20 September 2015; pp. 805–817.

19. Usunier, N.; Amini, M.R.; Goutte, C. Multiview Semi-supervised Learning for Ranking Multilingual Documents. In Proceediings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Athens, Greece, 4–8 September 2011.
20. Truong, T.V.; Amini, M.R.; Gallinari, P. A self-training method for learning to rank with unlabeled data. In Proceedings of the European Symposium on Artificial Neural Networks–Advances in Computational Intelligence and Learning, Bruges, Belgium, 22–24 April 2009.
21. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *IEEE Comput. J.* **2009**, *42*, 30–37. [CrossRef]
22. Guo, G.; Zhang, J.; Sun, Z.; Yorke-Smith, N. Librec: A java library for recommender systems. In Proceedings of the 23th Conference on User Modeling, Posters, Demos, Late-breaking Results and Workshop Adaptation and Personalization (UMAP 2015), Dublin, Ireland, 29 June–3 July 2015.