"Told You I Didn't Like It": Exploiting Uninteresting Items for Effective Collaborative Filtering

Won-Seok Hwang*, Juan Parc*, Sang-Wook Kim*, Jongwuk Lee†, and Dongwon Lee‡

*Department of Electronics and Computer Science, Hanyang University, Seoul, Korea
{hws23, crystaldia, wook}@hanyang.ac.kr

†Department of Computer Science and Engineering, Hankuk University of Foreign Studies, Yongin, Korea
julee@hufs.ac.kr

‡College of Information Sciences and Technology, The Pennsylvania State University, PA, USA
dongwon@psu.edu

***Abstract*—We study how to improve the accuracy and running time of top-*N* recommendation with *collaborative filtering* (*CF*). Unlike existing works that use mostly *rated* items (which is only a small fraction in a rating matrix), we propose the notion of *pre-use preferences* of users toward a vast amount of *unrated* items. Using this novel notion, we effectively identify *uninteresting* items that were not rated yet but are likely to receive very low ratings from users, and impute them as zero. This simple-yet-novel *zero-injection* method applied to a set of carefully-chosen uninteresting items not only addresses the *sparsity problem* by enriching a rating matrix but also completely prevents uninteresting items from being recommended as top-*N* items, thereby improving accuracy greatly. As our proposed idea is method-agnostic, it can be easily applied to a wide variety of popular CF methods. Through comprehensive experiments using the Movielens dataset and MyMediaLite implementation, we successfully demonstrate that our solution consistently and universally improves the accuracies of popular CF methods (*e.g.*, item-based CF, SVD-based CF, and SVD++) by two to five orders of magnitude on average. Furthermore, our approach reduces the running time of those CF methods by 1.2 to 2.3 times when its setting produces the best accuracy. The datasets and codes that we used in experiments are available at: https://goo.gl/KUrmip**

I. Introduction

The *Collaborative Filtering* (*CF*) technique, one of the most popular and effective techniques in recommender systems, has been extensively studied in recent years (*e.g.*, [1], [11], [19], [24], [16], [4]). By and large, CF can be applied to address two settings: (1) rating prediction and (2) top-*N* recommendation. In this paper, in particular, we focus on the setting of top-*N* recommendation for its practical usage in real-world applications [7]. For the top-*N* recommendation setting, CF predicts *relative* preferences on unrated items, and recommends top-*N* items with the highest preferences.

Due to the popularity of CF techniques in practice and importance of recommender systems in businesses, improving the results of CF can lead to huge positive business ramifications. In this paper, as such, we strive to improve CF techniques with respect to its *accuracy* and *running time*. Our proposal is *method-agnostic* so that it can be applied to a wide variety of CF techniques seamlessly, improving their results universally. Our proposal is based on the following hypothesis in CF:

Hypothesis 1. *Filling values into empty cells (i.e., unrated items) in a rating matrix can improve the accuracy and running time of the top-*N *recommendation by CF methods.*

Below, first, we develop several key ideas that are crucial to develop a solution based on the hypothesis.

Idea 1 (Exploit uncharted unrated items) *Existing CF methods in general attempt to exploit the ratings that users have provided, yet the fraction of known ratings in a rating matrix is in general quite small. Suppose, for a rating matrix R with m users and n items, on average, a user rates k items. Then, the fraction of rated items in R is: $\frac{m \times k}{m \times n} = \frac{k}{n}$. As it is often common for an e-business to sell millions of items with a very long tail and a user on average rates only a small number of items, asymptotically, such a fraction of rated items in R becomes extremely small (i.e., $k \ll n$). As such, we believe that exploiting the vast number of uncharted "unrated" items in R can lead to a significant improvement in CF.*

Several existing works assume that items are not evaluated (*i.e.*, unrated) when a user is not interested in them. Nan Hu et al. [8] and Nilesh Dalvi et al. [5] pointed out that most rated items have either 4 or 5 rating scores out of [1 ... 5], 5 being the best. In other words, users tend to evaluate only those items that they are likely to like (thus high ratings among rated items), while ignore items that they dislike in advance. Similar to this finding, Harald Steck [22] presumed that unrated items are less likely to be preferred by users. However, we emphasize that users are able to know only some of unrated items, not all of them.

Idea 2 (Some unrated items are uninteresting) *We observe that unrated items in R can be categorized into several types: (1) unrated items that users were not aware of their existence, (2) unrated items that users knew and purchased but did not rate, and (3) unrated items that users knew but did not like and thus did not purchase. In particular, we note that the unrated items of the third type, named as* ***uninteresting items*** *(I^{un}), clearly indicate users' latent "negative" preference. Therefore, we believe that it is better* not *to recommend those uninteresting items. Challenge is however that it is not easy to identify I^{un} in R.*

Idea 3 (Uninteresting items have low pre-use preferences) *In order to identify I^{un} in R, we first have to understand a user's* pre-use preference *to items–i.e., a user's impression on items before purchasing and using them. Then, based on this novel notion, we can rephrase I^{un} as* those items whose pre-use preferences are not high. *Unfortunately, ratings that users leave in R do not indicate pre-use preferences but*

the preferences "after" using the items, named as post-use preferences. *However, also note that users must have had high pre-use preferences to all rated items (otherwise, users would have not purchased them in the first place). Therefore, we believe that we can utilize the pre-use preferences of rated items to infer the latent pre-use preferences of unrated items, and finally identify I^{un} with low pre-use preferences.*

With I^{un} in R found, next, we propose to improve top-*N* recommendation via two strategies: (1) excluding I^{un} from the top-*N* recommendation results, and (2) exploiting both I^{un} and ratings to predict the relative preferences of items better. The first strategy is likely to improve top-*N* recommendation. As I^{un} with low pre-use preferences are by definition the items that users were aware of their existence but did not like in the first place, they are likely to be false positives if included in top-*N* recommendation. Therefore, proactively, we exclude I^{un} from the top-*N* recommendation results.

Our second strategy is also likely to improve top-*N* recommendation by recommending preferred items more accurately. We can explain the effect of second strategy through the concept of typical memory-based CF methods that depend on the fact that a user u is more likely to prefer items that her neighbors also like. Suppose a few neighbors rated an item i highly but most neighbors considered i as an uninteresting item (thus left i unrated in R). In this situation, existing CF methods are likely to recommend i to u as her a few neighbors rated them high. On the other hand, if we consider the fact that many more neighbors actually viewed i as *uninteresting*, we could avoid recommending i to u.

Based on the three ideas and two strategies, now, we propose a novel solution to apply the notion of *uninteresting items* to the framework of any existing CF methods. To the best of our knowledge, ours is the first work to exploit the notion of uninteresting items in improving CF methods. Our proposed solution consists of three steps: (1) it infers the pre-use preferences of unrated items by solving the *one-class collaborative filtering* (*OCCF*) problem [15], [20], (2) it assigns zero ratings to the found uninteresting items in R whose pre-use preferences are not high, yielding an augmented matrix Z, and (3) it applies any of existing CF methods to Z, instead of R. This simple-yet-novel imputation not only addresses the sparsity problem by enriching a rating matrix but also completely prevents uninteresting items from being recommended as top-*N* items, thereby improving the overall accuracy greatly.

To summarize, our main contributions are as follows:

- We introduce a new notion of *uninteresting items*, and divide a user's preference into *pre-use* and *post-use preferences* to identify uninteresting items.
- We successfully identify uninteresting items via the pre-use preferences inferred by solving the OCCF problem and show its implications and effectiveness.
- We propose to exclude uninteresting items by means of zero-injection, and improve the prediction of relative preferences by considering uninteresting items.
- We conduct comprehensive experiments using the real-world Movielens dataset and show that our solution improves the accuracy of baseline CF methods (*e.g.*, item-based CF, SVD-based CF, and SVD++) by 2.5 to 5 times on average, and reduces the running time of those CF methods by 1.2 to 2.3 times when its setting produces the best accuracy.

TABLE I. NOTATIONS USED IN THIS PAPER

Notation	Description
p_{ui}	User u's pre-use preference on item i
q_{ui}	User u's post-use preference on item i
r_{ui}	A rating given to item i by user u
P	A pre-use preference matrix whose entry is p_{ui}
Q	A post-use preference matrix whose entry is q_{ui}
R	A rating matrix whose entry is r_{ui}
I_u^{un}	A set of uninteresting items for user u
I_u^{in}	A set of interesting items for user u
I_u^{pre}	A set of preferred items for user u

The organization of this paper is as follows. In Section 2, we explain the preliminaries of our approach. In Section 3, we present our approach in detail. In Section 4, we evaluate our approach in comparison with existing ones via extensive experiments. In Section 5, we present the related work. In Section 6, we finally conclude our work and discuss future research directions.

II. PRELIMINARIES

In this section, we introduce the details of uninteresting items by contrasting them to interesting items, rated items, and unrated items. Furthermore, we clarify the difference between pre-use preferences and post-use preferences.

A user u's pre-use preference to an item i is a different notion from a rating given to i in a rating matrix, which really represents u's post-use preference to i. Existing CF methods mainly attempt to exploit ratings (thus post-use preferences). Let us first introduce a few basic notations used throughout this paper. Let $U = \{u_1, \ldots, u_m\}$ be a set of m users, $I = \{i_1, \ldots, i_n\}$ be a set of n items, and r_{ui} be the rating given to item i by user u. A corresponding rating matrix is referred to as $R = (r_{ui})_{m \times n}$, and p_{ui} (resp. q_{ui}) indicates user u's pre-use (resp. post-use) preference on item i. In theory, both types of preferences p_{ui} and q_{ui} exist on a user-item pair (u, i) although in reality they are not always computable or available. Table I summarizes key notations used in this paper.

Note that u has pre-use preference on i based on its *external* features that u could obtain without actually using i (*e.g.*, genre or director information, in case of a movie). After using i, based on the level of her satisfaction, u then assigns a specific rating to i, indicating her post-use preference for i. The post-use preference is therefore determined by the *inherent* features that u had not known before using i (*e.g.*, storyline or choreography of a movie). Let us contrast two preference types in a more detail using the following example.

EXAMPLE 1 (TWO PREFERENCE TYPES). Figure 1 illustrates the pre-use and post-use preferences of a user u for three movies. Initially, u has a high pre-use preference for Movie #1

Movie	Pre-use preference	Post-use preference	Rating
Movie#1	High	High	5
Movie#2	High	Low	1
Movie#3	Not high	Unknown	Unrated

Fig. 1. Pre-use, post-use preferences, and ratings for three movies.

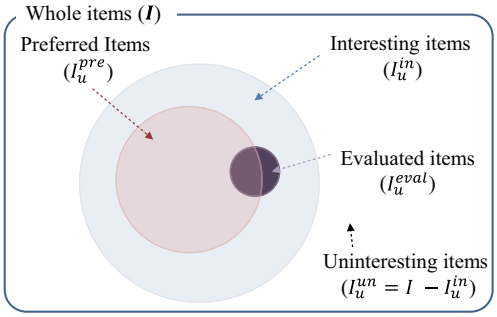

Fig. 2. Venn diagram for the preferences and interestingness of items.

and Movie #2. On the other hand, u does not have high pre-use preference for Movie #3. In that case, Movie #1 and Movie #2 are to be called as *interesting* items to u while Movie #3 as an *uninteresting* item. Thus, u would decide to watch only two movies with high pre-use preferences. After watching the movies, u likes Movie #1 as she expected but is disappointed at Movie #2. Therefore, u assigns a high rating to Movie #1 and a low rating to Movie #2. In contrast, u does not watch Movie #3 as it is an uninteresing movie to her. Her post-use preference to Movie #3 *remains unknown* (*i.e.*, empty in a rating matrix).

Figure 2 further depicts how a user u thinks about an entire set I of items with the *Venn diagram*. Let *interesting items* I_u^{in} denote items with high pre-use preferences while *uninteresting items* I_u^{un} denote u's items with not-high pre-use preferences. The two item sets are disjoint, *i.e.*, $I_u^{in} \cap I_u^{un} = \emptyset$, $I_u^{in} \cup I_u^{un} = I$. (In Section III-B, we will explain how to identify uninteresting items from entire item set I.) We formally define them as follows:

DEFINITION 1 (UNINTERESTING ITEMS). For a user u, a set of uninteresting items I_u^{un} is defined as: (1) $I_u^{un} = I - I_u^{in}$ and (2) $p_{ui} \leq p_{uh}$ $(\forall i \in I_u^{un}, \forall h \in I_u^{in})$.

Among interesting items I_u^{in}, user u buys/uses some items and evaluates them by assigning ratings. A set of items that are likely to get high ratings are called *preferred items*, denoted by I_u^{pre}, which is a subset of interesting items as shown in Figure 2 (*i.e.*, $I_u^{pre} \subseteq I_u^{in}$).

In a real scenario, u would be able to evaluate only a small fraction of interesting items. This item set, denoted by I_u^{eval}, is only a subset of I_u^{in} (*i.e.*, $I_u^{eval} \subseteq I_u^{in}$). For this reason, if we understand the uninteresting items of each user, we can understand the users' taste more accurately.

Based on this viewpoint, our goal is to identify top-N preferred items to user u by considering the uninteresting items for each user. Specifically, user u's pre-use and post-use preferences for item $i \in I_u^{eval}$ are known while both types of preferences for item $j \in I - I_u^{eval}$ are unknown.

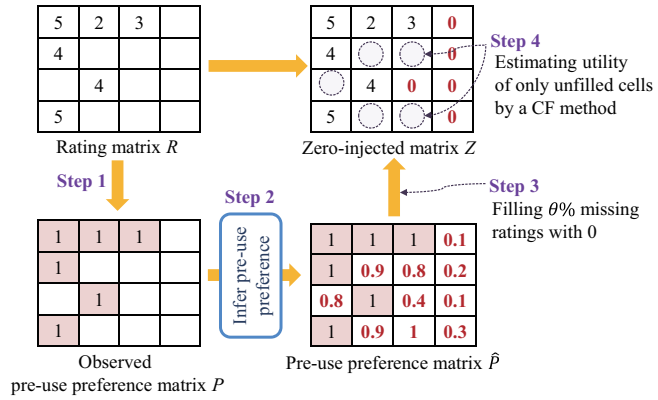

Fig. 3. Overview of our approach.

If u has evaluated item i, its pre-use preference p_{ui} can be considered high. Based on known pre-use preferences, we infer the pre-use preferences of the remaining unknown items j. In addition, its post-use preference q_{ui} can be directly indicated by the score of r_{ui}. We formally define our problem for top-N recommendation as follows:

PROBLEM 1 (TOP-N RECOMMENDATION) For user u, identify top-N items $J = \{j_1, ..., j_N\}$ such that: (1) $J \subseteq I_u^{in} - I_u^{eval}$ and (2) $q_{uj_1} \geq ... \geq q_{uj_N} \geq q_{uy}$ $(\forall y \in I - I_u^{eval} - J)$.

III. PROPOSED APPROACH

Existing CF methods employ user u's preferences only on evaluated (rated) items $i \in I_u^{eval}$. On the other hand, our approach identifies uninteresting items I_u^{un} and exploits them to enrich the rating matrix with additional user ratings. In addition, CF methods equipped with our approach could exclude those uninteresting items explicitly from top-N recommendation. We carefully analyze the benefits of our approach in Section III-D.

The main challenges of our approach are two-fold: (1) *how to identify uninteresting items among unrated items* and (2) *how to exploit uninteresting items thus discerned in CF methods*. To address the first challenge, we need to infer the pre-use preferences for all of the unrated items, and to find those unrated items whose pre-use preferences are not high. For the second challenge, we build the zero-injected matrix whose some entries are set as 0 if their corresponding items are considered *uninteresting*. This augmented matrix can be applicable to *any* CF methods (thus making our approach method-agnostic), which enables those CF methods to benefit from uninteresting items in their predictions.

Figure 3 depicts the overall process of our proposed approach. First, we build a *pre-use preference matrix* $P = (p_{ui})_{m \times n}$ by examining a rating matrix $R = (r_{ui})_{m \times n}$. Specifically, we set the pre-use preference, p_{ui}, of a user u on an item i as 1 when $r_{ui} \in R$ has been already rated (*i.e.*, u must have liked i so that she bought i) (Step 1). The value 1 is the highest pre-use preference because we represent p_{ui} as a real value in [0, 1]. Next, we infer (unknown) pre-use preferences on "unrated" user-item pairs (u, i) (*i.e.*, $p_{uj} = null$) based on other observed pre-use preferences (*i.e.*, $p_{ui} = 1$) and add them in P, which becomes $\hat{P}$ (Step 2). We define uninteresting items for each user based on $\hat{P}$, and build a *zero-injected matrix* $Z = (z_{ui})_{m \times n}$ (Step 3). That is, z_{ui} is set as r_{ui}

when user u has actually evaluated item i and is set as 0 if i is an uninteresting item for u. In our approach, an item j is an uninteresting item for a user v if j's pre-use preference score $\hat{p}_{ui}$ is ranked within the bottom $\theta\%$ in $\hat{P}$. Thus, Z is an augmented matrix that contains u's original ratings as well as "0", inferred and injected by our solution. We then apply a CF method to Z to predict the post-use preferences of empty entries (dotted circles) in Z (Step 4). Finally, we recommend top-*N* items to an active user. In the following subsections, we explain each step in detail.

A. Inferring Pre-Use Preferences

It is straightforward to determine a pre-use preference p_{ui} when a user u has already rated an item i (*i.e.*, $r_{ui} \neq null$) because i must be interesting to u in the beginning, *i.e.*, $I_u^{eval} \subseteq I_u^{in}$. As such, we set the pre-use preference p_{ui} as 1 in this case. When u has not rated i (*i.e.*, $r_{ui} = null$), determining p_{ui} becomes non-trivial. Therefore, our main challenge is to accurately infer pre-use preferences p_{ui} when unrated.

To address our challenge, we propose to borrow the framework of the *one-class collaborative filtering* (OCCF) problem [15], [20]. The OCCF problem occurs when a rating score is *unary* such as clicks, bookmarks, and purchases so that either a cell in a matrix has null value or a single value indicating "yes." Meanwhile, ambiguity arises in the interpretation of unrated items. That is, it is difficult to differentiate *negative* and *positive* examples that co-exist among unrated items [15]. Some unrated items could be positive as the user was not aware of the existence of the items but if she knew she would have liked them. On the other hand, some are negative as the user knew about the items but decided not use them as she did not like them.

This problem setting happens when we infer pre-use preferences for unrated items. That is, known pre-use preferences for rated items have a value of 1 (*i.e.*, $p_{ui} = 1$), and missing pre-use preferences for unrated items are ambiguous. In Figure 2, we observe that both unlabeled positive examples ($I_u^{in} - I_u^{eval}$) and negative examples (I_u^{un}) co-exist in the set of items whose pre-use preferences are unknown ($I - I_u^{eval}$). We thus employ the OCCF method [15] to infer pre-use preferences. [1]

The basic idea of the OCCF method is to treat all unrated items as negative examples, and assign weights to quantify the relative contribution of these examples. In our situation, it assigns 0 to p_{ui} whose value is null in P and determines weight w_{ui} by three schemes: *uniform*, *user-oriented*, and *item-oriented* schemes. In this paper, we employ the user-oriented scheme that was the best performer in [15]. The intuition of the user-oriented scheme is essentially *as a user rates more items, she is more likely to dislike unrated items.* That is, it computes the weight w_{ui} in proportion to the number of items rated by u: $w_{ui} = \sum_i p_{ui}$. The OCCF method finally updates the value of p_{ui} whose value is 0 using all entries in P and their corresponding weights. We treat the updated values as the inferred pre-use preference scores.

To update the values, the OCCF method employs the weighted alternating least squares (wALS) method [21] in building an SVD model with a matrix and weights. It infers the preference scores for each user's unrated items via the SVD model. The wALS method decomposes a matrix P into two low-rank matrices X and Y while optimizing an objective function $\pounds(X, Y)$. The matrix P represents observed pre-use preferences in our case, *i.e.*, $P = (p_{ui})_{m \times n}$. The matrices X and Y represent the features of users and items for latent factors, respectively. The objective function is represented as follows:

$$\pounds(X,Y) = \sum_u \big[\sum_i w_{ui} \{ (p_{ui} - X_u Y_i^T)^2 + \lambda(\|X_{u(\cdot)}\|_F^2 + \lambda \|Y_{i(\cdot)}\|_F^2) \} \big] \quad (1)$$

where p_{ui} and w_{ui} are the entries in the observed pre-use preference matrix P and the weight matrix W, respectively. The vector X_u is the u-th row of matrix X, and the vector Y_i is the i-th row of matrix Y. The two vectors represent the features of user u and item i. In addition, $\|\cdot\|_F$ denotes the *Frobenius norm* and λ is a regularization parameter.

In order to factorize the matrix P, the OCCF method first assigns random values to elements in the matrix Y, and updates elements in the matrix X as in Eq. (2) by optimizing the objective function. $\forall 1 \leq u \leq m$:

$$X_{u(\cdot)} = p_{u(\cdot)} \widetilde{w}_{u(\cdot)} Y (Y^T \widetilde{w}_{u(\cdot)} Y + \lambda (\sum_i w_{ui}) L)^{-1} \quad (2)$$

where $\widetilde{w}_{u(\cdot)}$ is a diagonal matrix with elements of $w_{u(\cdot)}$ on the diagonal, and matrix L is an identity matrix. After that, the OCCF method updates elements in the matrix Y while fixing the matrix X as in Eq. (3). $\forall 1 \leq i \leq n$:

$$Y_{i(\cdot)} = p_{(\cdot)i}^T \widetilde{w}_{(\cdot)i} X (X^T \widetilde{w}_{(\cdot)i} X + \lambda (\sum_u w_{ui}) L)^{-1} \quad (3)$$

We optimize the objective function by repeating Eq. (2) and Eq. (3) until matrices X and Y converge to a local optimum. Finally, we approximate matrix $\widehat{P}$ by calculating an inner product of X and Y as in Eq. (4) where an entry $\widehat{p}_{ui}$ in the matrix $\widehat{P}$ represents a pre-use preference score of user u for item i.

$$\widehat{P} \approx P = XY^T \quad (4)$$

B. Identifying Uninteresting Items

Once pre-use preferences of unrated items are inferred, next, we attempt to identify uninteresting items. Based on the pre-use preference scores inferred by the OCCF method, a user u's uninteresting items are defined as follows:

$$I_u^{un}(\theta) = \{ i | \rho(\hat{p}_{ui}) \leq \theta, r_{ui} = null \} \quad (5)$$

where $\rho(\hat{p}_{ui})$ indicates the percentile rank of $\hat{p}_{ui}$ among all user-item pairs whose ratings are missing in R. For instance, $I_u^{un}(20)$ indicates that we assign all unrated items whose percentile ranks of pre-use preference scores are at the bottom 20% as uninteresting items.

In Eq. (5), we do not use an absolute cut-off value for pre-use preference scores because the OCCF method is originally designed for computing users' *relative* preferences. In addition, we adjust the parameter θ to obtain the best accuracy for top-*N* recommendation. If θ is set high, a large number of zero

[1] Note that if there was a method better than the OCCF method, it would only improve our result further.

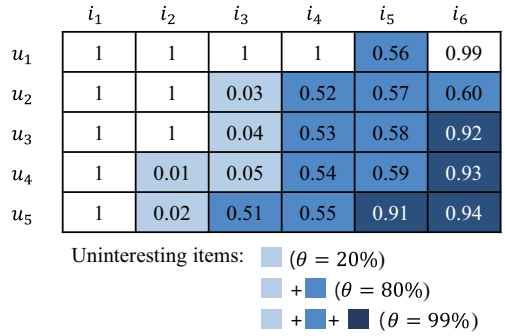

Fig. 4. A Pre-use preference matrix $\hat{P}$.

ratings are injected to Z, leading a less sparse rating matrix at the cost of some zeros being false set. On the hand, if θ is set low, we may not be fully utilizing the benefit of uninteresting items as only a small number of zero ratings are injected. Our simple use of *relative* cut-off based on percentile rank works well, at the end, as we will demonstrate in experiments.

EXAMPLE 2. Figure 4 illustrates a pre-use preference matrix $\hat{P}$, where the cells with 1 and with decimal numbers are originally derived from the rated and unrated items in R, respectively. As a larger θ is set, more items are determined as uninteresting items. For example, when $\theta = 20$, only light-colored cells become uninteresting items. If $\theta = 80$, both light and middle-colored cells become uninteresting items. Finally, when $\theta = 99$, all colored cells become uninteresting items. Note that the number of uninteresting items could be different per user. For instance, when $\theta = 80$, the numbers of uninteresting items for u_1 and u_2 are 1 and 4, respectively.

It is worthwhile to emphasize that our approach identifies uninteresting items more broadly than what a user herself would have recognized. In a real setting, even if asked, users are able to review only a small fraction of (millions of) unrated items to identify truly uninteresting items. In contrast, our approach finds uninteresting items that users have not recognized yet but are likely to consider uninteresting when presented.

C. Zero-Injection

This paper proposes a novel approach to fill a part of missing ratings, named as *zero-injection*, which assigns a zero value to r_{ui} in R if an item i is determined as a user u's uninteresting item. We inject zeros for missing ratings because a user would not be satisfied with her uninteresting items even when recommended.

By augmenting a rating matrix R with zeros, the zero-injection method builds a new matrix that contains zero ratings as well as actual user ratings. We call this augmented matrix as *zero-injected matrix* $Z = (z_{ui})_{m \times n}$, where entry z_{ui} can be represented as follows:

$$z_{ui} = \begin{cases} r_{ui} & \text{if } u \text{ has evaluated } i; \\ 0 & \text{if (1) } u \text{ has not evaluated } i, \text{ and} \\ & \text{(2) } i \text{ is an uninteresting item to } u; \\ null & \text{otherwise} \end{cases}$$

Note that z_{ui} is set as r_{ui} if user u has rated item i. On the other hand, if i has not been rated by u ($r_{ui} = null$) and its pre-use preference p_{ui} is not high, z_{ui} is set as 0. Otherwise, z_{ui} is set as *null*, indicating "unknown." Entry z_{ui} indicates a user u's rating for an item i. When the value of z_{ui} is zero, it implies that user u is very unlikely to love item i even though i is recommended to u. When the value of z_{ui} is the rating given by user u, it indicates that user u is satisfied with item i to the extent that its rating score indicates.

Note that our proposed approach works regardless of the choice of underlying CF methods as we can simply replace the original rating matrix R by the zero-injected matrix Z. In other words, our approach is orthogonal to existing CF methods, which is one of strengths in our approach. Businesses and recommender systems can choose any appropriate CF methods suiting their particular settings, yet still use our proposed idea for improving accuracy and running time.

Our proposed approach can improve the CF methods in two aspects. When CF methods are applied, the zero-assigned items for a user are *excluded* from her recommendation list (as ratings are zero). We note that existing CF methods consider *all* items whose ratings are missing as recommendation candidates. Therefore, the zero-injected matrix prevents a user's uninteresting items[2] from recommendation. In addition, the zero-injected matrix provides a much more number of ratings (including ratings with zero values) than the original rating matrix. Thus, the CF methods equipped with our approach are able to understand relative users' preferences more accurately when they are applied to the zero-injected matrix. Moreover, as the number of recommendation candidates reduces, the computational cost also decreases because the CF methods have only to compute the relative preferences for a small number of items.

Similar to our approach, PureSVD [4] also assigns zero values to missing ratings. Unlike our approach, however, PureSVD has no regard for identifying users' uninteresting items, and fills zero to "all" missing ratings (items $i \in I - I_u^{eval}$ for each user in Figure 2). PureSVD regards user's favorite items as the items with low preferences ($I_u^{in} \subseteq I - I_u^{eval}$ in Figure 2). Unlike this approach, our zero-injection selectively fills only the items in I_u^{un} of low pre-use preference scores with zeros, recognizing a user u's taste more precisely. Moreover, PureSVD cannot explicitly prevent uninteresting items from being recommended. In experiments, we demonstrate the superiority of our approach over PureSVD.

D. Why Does Zero-Injected Matrix Help?

We argue that the zero-injected matrix helps improve the accuracy of any CF methods. To present the ground for our argument, we discuss the effect of our approach when applied to two popular CF methods: the *item-based collaborative filtering (ICF)* [19] and the *SVD-based method (SVD)* [24].

ICF predicts a rating $\hat{z}_{ui}$ for a target item i of a user u by referencing her ratings on those items similar to the item i as follows:

$$\hat{z}_{ui} = \frac{\sum_{j \in S_i} \{z_{uj} * sim(i,j)\}}{\sum_{j \in S_i} sim(i,j)} \quad (6)$$

where S_i is a set of (up to) k items that have users' rating patterns most similar to that of i and u's rating is known

[2]We note that the user is highly unlikely to use those items.

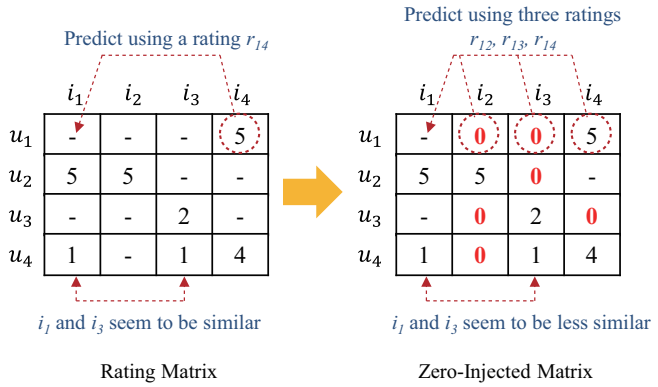

Fig. 5. Rating and zero-injected matrices in CF.

for. If there are less than k users who have evaluated i, S_i includes this number of items only instead of k. In addition, $sim(i, j)$ denotes the similarity between items i and j in terms of users' rating patterns. This paper adopts Pearson's correlation coefficient as a similarity metric, which is widely used in recommendation field [1], [3].

Figure 5 illustrates the difference between a rating matrix R and its corresponding zero-injected matrix Z built by our approach. We observe that, unlike R, Z has extra zero ratings (in boldface) implying users' uninteresting items.

Suppose that ICF predicts the rating r_{11} of a user u_1 for an item i_1 when its parameter k is set as 3. With the rating matrix, it refers to only r_{14} for prediction[3] because u_1 has not rated i_2 and i_3 (*i.e.*, r_{12} and r_{13} are nulls). Therefore, its predicted rating would be inaccurate due to the sparsity problem. On the other hand, thanks to zero-injection, ICF considers more ratings, z_{12} and z_{13}, in addition to z_{14} with Z. This is because item set S_i now contains a more number of ratings, including two uninteresting items.

In addition, our approach helps find the items that are truly similar. With R, ICF may conclude that i_1 and i_3 are highly similar because u_4 gives 1 to both of them. However, the similarity can be inaccurate because it is based on only a single user's opinion. With Z, the two items are regarded less similar because u_2 rated two items as quite different. As such, the zero-injected matrix is useful to compute a more accurate similarity because it enables CF to reflect more users' opinions. In particular, we note that the zero-injection makes it possible for ICF to successfully find truly similar users who have a set of "uninteresting" items in common, which has been overlooked in existing CF methods.

In Figure 5, we only show the accuracy improvement owing to a more number of items in S_i. Moreover, even when the number of items in S_i is the same, the accuracy could increase because S_i with our zero-injected matrix contains those items more similar to item i than S_i obtained with R.

Next, we also explain how the zero-injected matrix makes existing SVD-based methods [24] more accurate. Given Z, SVD factorizes it into an inner product of two low-rank matrices X and Y with a dimension f. That is, one low-rank matrix is an m-by-f *user-factor* matrix and the other is an n-by-f *item-factor* matrix. Each user u is thus associated with an f-dimensional vector $x_u \in \mathbb{Z}^f$. Each item i is involved with an f-dimensional vector $y_i \in \mathbb{Z}^f$. The rating prediction for $\hat{z}_{ui}$ is computed by Eq. 7:

$$\hat{z}_{ui} = x_u y_i^T \tag{7}$$

With the original rating matrix R in Figure 5, SVD cannot recognize that u_2 is related to u_1 or u_3 because they have no common items rated. In contrast, using Z, it now successfully observes the relationship between those users, *i.e.*, both u_2 and u_1 are not interested in i_3, and have different opinions on i_2. Similarly, it also misses items' relationships such as i_2 and i_3 with R while it is able to find the relationships with Z. Therefore, SVD can build a better model representing more latent relationships among users and items with our zero-injected matrix, which helps improve the overall accuracy of top-N recommendation.

IV. Evaluation

In this section, we evaluate the effectiveness and efficiency of our proposed approach with a real-life dataset and an open-source implementation of existing CF methods. Specifically, we first validate the accuracy of the OCCF method [15] to infer users' pre-preferences in comparison with other methods and also verify our assumption that users are unlikely to use uninteresting items with low pre-use preferences. In addition, we perform the sensitivity analysis on parameter θ that is crucial to determine uninteresting items. Finally, we show the accuracy and running time of the modified CF methods equipped with our approach compared with the original ones.

A. Experimental Set-Up

We use the MovieLens 100K dataset [16], widely used for evaluating recommender systems [15], [22], [13], [23]. This dataset consists of 943 users, 1,682 items, and 100,000 ratings. The ratings are integer values from 1 (*i.e.*, worst) to 5 (*i.e.*, best). The minimum number of ratings per user is 20.

For evaluating the accuracy of top-N recommendation, we vary the value of N from 5 to 20 in an increment of 5 (default value = 5). Only the items with the rating score of 5 are considered as *relevant*, *i.e.*, ground truth, as correctly predicting items with high ratings has more business ramifications (than predicting items with low ratings), as argued in [22].

We adopt four metrics to measure the accuracy such as *precision*, *recall*, *normalized discounted cumulative gain* (nDCG), and *mean reciprocal rank* (MRR). For a user u, precision $P_u@N$ and recall $R_u@N$ can be computed by $\frac{|Rel_u \cap Rec_u|}{|Rec_u|}$ and $\frac{|Rel_u \cap Rec_u|}{|Rel_u|}$, respectively, where Rec_u denotes a set of N items that each method recommends to u, and Rel_u denotes a set of items considered relevant (*i.e.*, ground truth). We also use nDCG to reflect ranked positions of items in Rec_u. Let y_k represent a binary variable for the k-th item i_k in Rec_u. If $i_k \in Rel_u$, y_k is set as 1. Otherwise, y_k is set as 0. Then, $nDCG_u@N$ is computed by $\frac{DCG_u@N}{IDCG_u@N}$, where $DCG_u@N = \sum_{k=1}^{N} \frac{2^{y_k}-1}{\log_2(k+1)}$, and $IDCG_u@N$ means an *ideal* $DCG_u@N$ where y_k is set as 1 for every item $i_k \in Rec_u$. MRR shows the average inversed rankings of every item $i_k \in Rec_u$. MRR_u can be computed by $\frac{1}{|Rel_u|} \sum_{i=1}^{|Rel_u|} \frac{1}{rank_i}$. All measurements are averaged using 5-cross validation.

[3] Otherwise, the items whose similarity to i_1 is less than i_2 and i_3 need to be included in S_i.

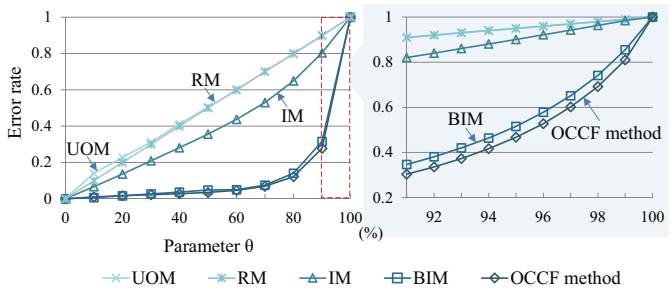

Fig. 6. Error rate comparison of five inference methods.

To show the effectiveness of our approach, we first augment a rating matrix R to a zero-injected matrix Z using our proposed method, and feed Z as input to existing CF methods such as an item-based CF (ICF) [19] and SVD-based CF (SVD) [24], well known methods in memory-based and model-based approaches, respectively. Further, we use the ICF and SVD methods implemented in the open-source MyMediaLite [6] using their default parameter settings in our experiments. All datasets and codes that we used in experiments are available at: https://goo.gl/KUrmip.

Our empirical study is to answer the following questions via comprehensive experiments.

- $Q1$: Is the OCCF method (most) effective to infer users' pre-use preferences?
- $Q2$: Are users really not satisfied with uninteresting items that our approach determines?
- $Q3$: How does the accuracy of our approach vary over different θ? How sensitive is θ?
- $Q4$: How much does our approach improve existing CF methods with respect to accuracy? Is Hypothesis 1 valid (with respect to accuracy)?
- $Q5$: How much time our approach spends compared to existing CF methods? Is Hypothesis 1 valid (with respect to running time)?

B. Q1: Inference of Pre-Use Preferences

As the gist of our approach to identify uninteresting items depends on the effectiveness of the OCCF method to infer users' pre-use preferences, first, we test how effective the OCCF method is, against four other candidates ($Q1$): user-oriented method (UOM), binary item-based method (BIM), item-based method (IM), and random method (RM). UOM borrows the idea from the user-oriented scheme that determines the weights in the OCCF method (mentioned in Section III-A). UOM determines user's pre-use preference scores to be inversely proportional to the number of her rated items. BIM uses the "observed" (binary) pre-use preference matrix P (introduced in Section III-A) and infers the pre-use preference scores exploiting the item-based CF [19]. IM is identical to the original item-based CF, which produces pre-use preference scores based on the original rating matrix R. RM is a baseline that does not infer pre-use preference scores but randomly selects uninteresting items among unrated items. We also employ the same parameter settings as in [15] for wALS in the OCCF method.

TABLE II. ACCURACY (*i.e.*, P@5) OF CF METHODS EQUIPPED WITH OUR APPROACH WITH FIVE INFERENCE METHODS.

CF method	Parameter Θ	Inference Method				
		OCCF	BIM	IM	UOM	RM
ICF	30%	**0.199**	0.155	0.123	0.062	0.085
	60%	**0.199**	0.165	0.130	0.021	0.035
	90%	**0.201**	0.154	0.134	0.004	0.009
Average		**0.200**	0.158	0.129	0.029	0.043
SVD	30%	**0.177**	0.137	0.110	0.062	0.067
	60%	**0.189**	0.150	0.144	0.034	0.039
	90%	**0.207**	0.192	0.144	0.011	0.008
Average		**0.191**	0.160	0.133	0.036	0.038

As the accuracy measure for $Q1$, we especially define an *error rate*, which captures how many rated items are selected as uninteresting items (*i.e.*, mis-classified) for each user by an inference method. The idea behind this notion is that as user's pre-use preferences should have been relatively high for her *rated items*, an inference method that does a less number of mis-classification is deemed as a better solution. A user u's error rate is defined as: $err_u^\theta = \frac{|I_u^{un}(\theta) \cap I_u^{test}|}{|I_u^{test}|}$, where I_u^{test} is a set of items *rated* by u in a test set, and $I_u^{un}(\theta)$ is a set of items determined as uninteresting (*i.e.*, ranked in the bottom $\theta\%$ according to the inferred pre-use preference scores) by the particular inference method. The lower a user's error rate gets, the better an inference method is.

Figure 6 shows the changes of error rates of five inference methods, averaged over all users with varying θ. In general, as θ increases, error rates of five methods increase as well. In particular, the error rates of UOM, IM, and RM increase more rapidly than those of the OCCF method and BIM. The OCCF method and BIM have relatively small error rates until θ reaches to 90%, implying that their accuracy is fairly good when θ is smaller than 90%. Above 90%, their error rates grow rapidly. This is because at this point there are only a relatively small number of unrated items left among which a substantial amount of interesting items exist. Among five methods, the OCCF method shows the best error rates, regardless of θ. Therefore, we conclude that it is the most effective in correctly inferring pre-use preferences.

So far, we validate that pre-use preferences inferred by the OCCF method are most accurate. Next, we examine if pre-use preferences inferred by the OCCF method indeed yield the best accuracy when used in real CF methods, *i.e.*, the end-to-end performance. We build the zero-injected matrix Z using pre-use preferences inferred by five inference methods, and apply Z to two CF methods–the *item-based CF* (*ICF*) and *SVD-based CF* (*SVD*). In this experiment, we vary θ as 30%, 60%, and 90%. As all accuracy metrics of Section IV-A show similar tendencies, we only report the results for P@5 here (More detailed analysis on the effect of θ will be given in Section IV-D).

Table II shows the precision@5 scores of ICF and SVD using a zero-injected matrix, inferred by five different inference methods. Similar to Figure 6, the OCCF method shows the best accuracy in all cases (19%–26% higher than BIM, the second best one) while UOM shows the worst accuracy (even lower than RM). Therefore, as the answer to $Q1$, we conclude that the OCCF method is the most effective for inferring pre-use

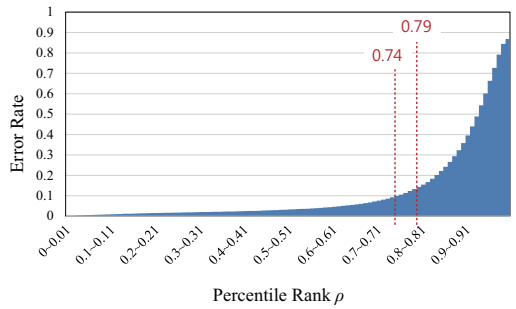

Fig. 7. Users' error rates according to items' pre-use preference scores.

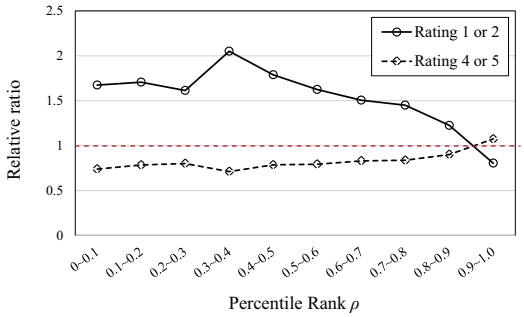

Fig. 8. Distribution of users' pre-use preference scores for rated items.

preferences, and, in subsequent evaluation, employ only the OCCF method for inferring pre-use preferences.

C. Q2: User's Satisfaction on Uninteresting Items

One of main assumptions of our proposal is that a user would not be satisfied with her uninteresting items. If our assumption does hold in practice, then a user is unlikely to purchase or use the uninteresting items recommended to her, and even if she does, she is more likely to assign low ratings to these items. To validate this assumption, we examine whether users are going to be satisfied with items in proportion to their pre-use preference scores. Recall that an item is likely to be selected as uninteresting if its pre-use preference score is relatively low.

First, we observe what portion of items are rated according their pre-use preference scores. For this observation, we first hide some ratings (*i.e.*, a test set), and compute the pre-use preference scores for all unrated user-item pairs by using the remaining ratings (*i.e.*, a training set). Next, we divide the unrated user-item pairs into 100 bins according to their percentile rank ρ of pre-use preference scores. We calculate the error rates for the j-th subset, $I_u^{un}(\beta_j, \beta_{j+1})$, instead of $I_u^{un}(\theta)$ to show the ratio of those user-item pairs that are really rated in the test set. $I_u^{un}(\beta_j, \beta_{j+1})$ includes u's unrated items whose rank ρ is between β_j and β_{j+1} (*i.e.*, $\beta_j \leq \rho(\hat{p}_{ui}) < \beta_{j+1}$ for $\forall i \in I_u^{un}(\beta_j, \beta_{j+1})$).

Figure 7 depicts the distribution of error rates over ρ. As ρ increases, the error rate increases rapidly. Moreover, the test set verifies that users have evaluated only a few items whose ρ is low. For example, among all the rated items, 90% items (resp. 95% items) have ρ higher than 74% (resp. 79%) (marked in Figure 7). This result indicates that users hardly ever use the items whose pre-use preference scores are not high, thereby supporting out assumption.

Next, we investigate users' given rating scores according to their pre-use preference scores. For this analysis, we build training and test sets, and make the OCCF method infer pre-use preferences for unrated user-item pairs by using the training set. After that, we divide those user-item pairs into 10 bins according to their percentile rank ρ of pre-use preference scores. By referring to the test set, we compare the number of items rated as 1 or 2 and that of items rated as 4 or 5 in each bin. For a fair comparison, we need to note two important observations: users leave 4 or 5 ratings (*i.e.*, 55% of all ratings) much more often than 1 or 2 ratings (*i.e.*, 17% of all ratings) in the MovieLens dataset; In addition, the numbers of "rated" items in the test set differ significantly depending on the bins. For example, 0.1% of user-item pairs have ratings in the first bin while 86.8% of pairs does in the 99-th bin. Considering these inequalities, we compute the *relative ratio* of items *rated as 1 or 2* (resp. *4 or 5*) for each bin as follows:

$$ratio_u(j,s) = \frac{|I_u^{test}(s) \cap I_u^{un}(\beta_j, \beta_{j+1})|}{|I_u^{test} \cap I_u^{un}(\beta_j, \beta_{j+1})|} \div \frac{|I_u^{eval}(s)|}{|I_u^{eval}|} \quad (8)$$

where I_u^{test} is a set of items evaluated by a user u in the test set, and $I_u^{test}(s)$ indicates a set of items rated as s in I_u^{test}. I_u^{eval} indicates a set of items rated by u among all items (*i.e.*, the test as well as the training set). $I_u^{eval}(s)$ presents a set of items rated as s in I_u^{eval}. The fraction before the division sign in Eq. 8 means the ratio of items rated as s to all rated items in a bin. In addition, the fraction after the division sign in Eq. 8 (*i.e.*, the ratio of items rated as s to the whole rated items) is necessary for normalization. A higher relative ratio indicates that more items rated as s exist in a bin.

Figure 8 shows the distribution of relative ratios. When ρ is smaller than 0.3, the relative ratio of items rated as 1 or 2 stagnates as ρ gets higher. This is because there are only a few rated items whose ρ is less than 0.3. When ρ is higher than 0.3, the relative ratio of items rated as 1 or 2 decreases as ρ gets higher. When ρ is smaller than 0.9, the relative ratio of items rated as 4 or 5 is smaller than 1. Only when ρ is in the range of 0.9 and 1, the relative ratio is higher than 1. In short, the items whose ρ is smaller than 0.9 are likely to be rated as 1 or 2 rather than 4 or 5. On the other hand, the items whose ρ is higher than 0.9 are more likely to be rated as 4 or 5. Therefore, we know that users are less likely to be satisfied with the items whose ρ is less than 0.9.

Based on the results of two experiments above, as the answer to Q2, we conclude that users are rarely satisfied with the items whose pre-use preferences are not high. In addition, we found that users tend to be unsatisfied with most of items in R (*e.g.*, 90%), suggesting that there are many items uninteresting to a user.

D. Q3: Effect of Parameter θ

As the parameter θ controls the amount of values imputated with zero-injection, it greatly affects the accuracy in recommendation. To verify the effect of θ, therefore, we conduct the sensitivity test. We first build different zero-injected matrices with varying θ and apply them to two CF methods, ICF [19] and SVD [24].

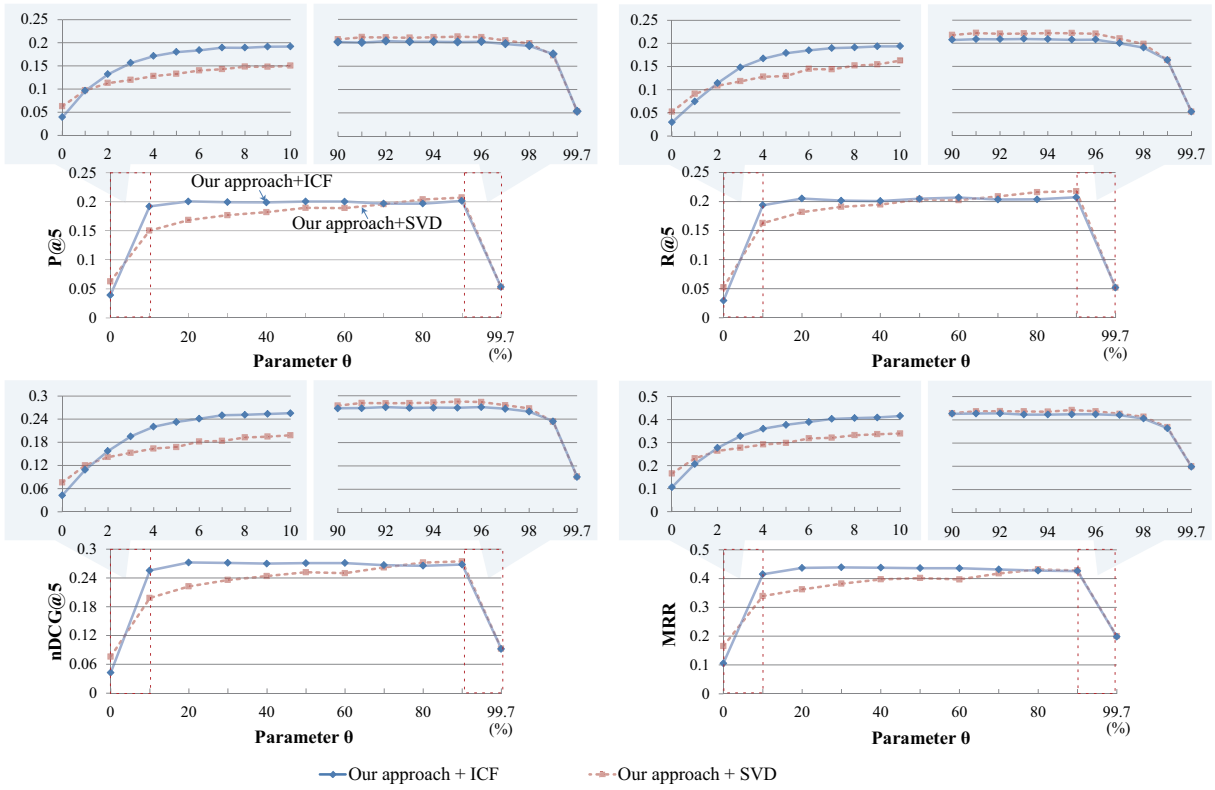

Fig. 9. Accuracy of ICF and SVD equipped with our proposed approach with varying parameter θ.

Figure 9 shows the accuracy of top-N ($N = 5$) recommendation with ICF and SVD with varying θ. We increase θ in an increment of 10% for the range of 10−90%, while increase θ in an increment of 1% for two extreme ranges, 0−10% and 90−99.7%. Note that we do not report the result with $\theta = 100\%$ because CF methods with our approach recommend *nothing* in this case. For this reason, we set θ up to 99.7% in order to leave only 5 items whose pre-use preference scores are the highest for each user. In this case, these 5 remaining items are thus *all* recommended (as top-5) to each user without requiring further CF methods. In summary, the result with $\theta = 0\%$ indicates the accuracy of original ICF and SVD methods without using our approach, while the result with $\theta = 99.7\%$ indicates the accuracy of the OCCF method without using ICF or SVD.

In Figure 9, we observe that the results of precision, recall, nDCG, and MRR show similar patterns. All these accuracy values of CF methods increase as θ increases up to around 95%. Moreover, accuracies in general grow rapidly until θ reaches 10%. All results clearly show that our idea of using zero-injection *dramatically improves* the accuracy of two original CF methods. ICF using our approach with $\theta = 96\%$ shows the best precision, *5.2 times higher* than ICF without our approach. When $\theta = 95\%$, similarly, our approach improves the precision of SVD by *3.4 times*.

As mentioned earlier, the OCCF method can be used to produce top-N recommendation without using ICF and SVD (*i.e.*, θ = 99.7%). The method, however, shows accuracy much lower than ICF and SVD equipped with our approach. This implies that the OCCF method is quite effective in finding uninteresting items. However, it is not that effective in recommending the final items because it ignores rating scores.

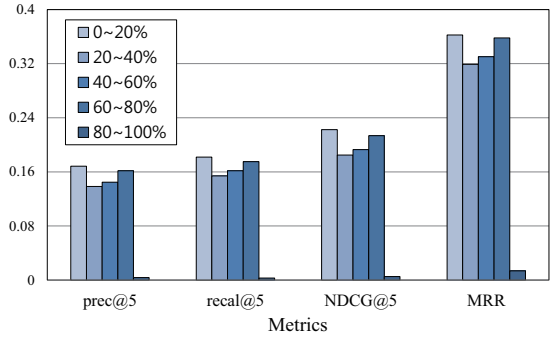

Fig. 10. Accuracy of SVD with five uninteresting item sets defined by percentile rank ρ.

The accuracy changes greatly when θ is less than 10% or more than 90% while it changes little when θ is between 10% and 90%. We can interpret this phenomenon as follows: (1) as θ increases up to 10%, more user-item pairs (*i.e.*, highly likely to be uninteresting) are filled with zeros (giving *accurate* and *useful* information to CF methods) and are also *correctly* excluded from top-N recommendation. (2) When θ ranges between 10% and 90%, accuracy changes little because filling unrated user-item pairs in the case of (1) has already alleviated most of the data sparsity problem. Filling more zero ratings no longer gives useful information to CF methods although user-item pairs whose θ is between 10% and 90% are highly likely to be uninteresting (See Section IV-C). (3) When θ is larger than 90%, the accuracy decreases significantly. As θ reaches 99.7%, more user-item pairs having *high* pre-use preferences (*i.e.*, could be interesting to users) are incorrectly

filled with zeros (giving inaccurate and less useful information to CF methods) and could be *incorrectly* excluded from top-*N* recommendation.

To understand this phenomenon more clearly, we define five sets of uninteresting items according to the percentile rank ρ of pre-use preference scores. Figure 10 shows the accuracy of SVD with the five sets of uninteresting items including those items whose ρ is $0-20$, $20-40$, $40-60$, $60-80$, and $80-99.7\%$, respectively. In the results, all accuracy metrics show similar tendencies. The four results of $0-20$, $20-40$, $40-60$, $60-80\%$ produce good accuracies, while that of $80-99.7\%$ shows significantly worse accuracies. This implies that, among the items whose ρ is $0-80\%$, there are only a few interesting items (errors) as we explained above. However, among the items whose ρ is $80-99.7\%$, there are many interesting items. This phenomenon is also observed in Section IV-C.

As to $Q3$, finally, we conclude that accuracies increase greatly when θ is less than 10% while it decreases significantly when θ is more than 90%. In addition, accuracies remain quite high and changes little for a large interval of $10\% \leq \theta \leq 95\%$, indicating that we can obtain a high accuracy even if we arbitrarily set θ within that interval, making our approach *parameter-insensitive* (with respect to θ).

E. Q4: Accuracy of CF Methods with Our Approach

We apply our approach to four existing CF methods (*i.e.*, ICF [19], SVD [24], *SVD++* [2], [10], and *pureSVD* [4]), and validate its effectiveness in improving accuracy. SVD++ utilizes both ratings and binary ratings (whether a user evaluates an item or not). PureSVD fills *all* missing ratings with zeros, and then produces recommendation based on the SVD model. Based on the findings in Section IV-D, we set the parameter θ as 90%.

Table III shows the accuracy of all CF methods with and without our approach. We denote a CF method equipped with our approach as *name_ZI* (*i.e.*, zero-injection) such as ICF_ZI, SVD_ZI, SVD++_ZI, and PureSVD_ZI. The numbers in boldface indicate the highest accuracy among all CF methods with and without our approach.

Among *existing* CF methods, PureSVD has the best accuracy while ICF shows the worst accuracy. In literature, both PureSVD and SVD++ are known to provide a better accuracy than SVD and ICF. We confirmed that our finding is consistent with [4]. We observe that our approach *dramatically improves* the accuracy of all the existing CF methods. For example, our approach improves $P@5$ of ICF, SVD, and SVD++ by *5, 3.3, and 2.5 times*, respectively. When our approach is applied to PureSVD, its improvement is the smallest. The reason is that PureSVD already assigns zeros to *all* missing ratings, which is an idea similar to the zero-injection in our approach. However, PureSVD fills zeros even for the items that *could be interesting* to users, which could affect the accuracy adversely. Furthermore, our approach employs another strategy of excluding uninteresting items from top-*N* recommendation, that contributes to the slight accuracy improvement over PureSVD.

Among the CF methods equipped with our proposed approach, SVD_ZI performs the best, followed by ICF_ZI. Our approach improves both SVD and ICF greatly because they consider all unrated items as unknown ones. For this reason, SVD_ZI and ICF_ZI adopt additional information correctly by regarding zero ratings as uninteresting ones and null values as unknown ones. Meanwhile, our approach cannot improve SVD++ and PureSVD as much as SVD and ICF because SVD++ and PureSVD originally have a positive view on rated items and a negative view on unrated items. SVD++ builds a SVD model by considering whether a user rates an item, but SVD++_ZI cannot distinguish the uninteresting items and rated items because all of them have rating values. PureSVD_ZI fails to discriminate zero ratings and null values in our zero-injected matrix. Specifically, since it assigns zero ratings again to all unrated items in our zero-injected matrix, in view of rating scores, unrated interesting items (determined by our approach) are also regarded as uninteresting.

We note that SVD_ZI (the most accurate CF method equipped with our approach) has an accuracy about *2 times higher* than PureSVD on average, the best one among existing CF methods, found in our experiments and also reported in [4]. Furthermore, PureSVD_ZI, the least accurate CF method equipped with our approach, still outperform all existing CF methods without using our approach.

In summary, our approach significantly improves the accuracy of all four CF methods used in our experiments via zero-injection while the degrees of improvement vary. As the answer to $Q4$, therefore, we conclude that our approach improves the accuracy of existing CF methods by *2.5 to 5 times on average, which is significant in comparison with the results obtained by other state-of-the-art methods reported in literature* [4], [7].

F. Q5: Running Time of CF Methods with Our Approach

In this section, we compare the execution times of CF methods with and without our approach. Our approach has both strengths and weaknesses in terms of execution times. We note the weaknesses happen at the pre-computation stage (offline) while the strengths happen at the recommendation stage (online). The CF methods build a model or computes similarities of item pairs during the pre-computation stage; they compute relative preferences and find top-*N* items during the recommendation stage.

Our approach reduces the recommendation time because it significantly reduces the number of candidate items whose relative preferences need to be predicted. Meanwhile, our approach may require more pre-computation time because it has to infer pre-use preference scores for all missing ratings by exploiting the OCCF method. In this section, therefore, we study the trade-off of our approach by examining the running time when our approach is applied to both ICF and SVD.

Figure 11 shows both recommendation and pre-computation times of SVD_ZI, SVD, SVD++, and pureSVD. Specifically, the recommendation time indicates those for predicting users' ratings and providing the items to users; the pre-computation time indicates the time for building a SVD model with a rating matrix R (SVD, SVD++, and PureSVD) and a zero-injected matrix Z (SVD_ZI). In Figure 11(a), the recommendation time of SVD_ZI decreases rapidly as θ increases because there remain *fewer* candidate items as θ increases. In addition, SVD_ZI takes a shorter time at the

TABLE III. ACCURACY OF FOUR CF METHODS EQUIPPED WITH OUR APPROACH ($\theta = 90\%$).

Metric		ICF			SVD			SVD++			PureSVD		
		Original	Ours	Gain	Original	Ours	Gain	Original	Ours	Gain	Original	Ours	Gain
Precision	@5	0.039	**0.201**	**413.8%**	0.063	**0.207**	229.7%	0.076	**0.193**	153.3%	0.100	**0.106**	6.7%
	@10	0.041	**0.161**	**292.6%**	0.056	**0.166**	196.9%	0.069	**0.154**	123.9%	0.082	**0.089**	9.1%
	@15	0.040	**0.137**	**243.7%**	0.053	**0.142**	169.9%	0.063	**0.134**	112.0%	0.071	**0.078**	11.3%
	@20	0.039	**0.121**	**211.7%**	0.048	**0.125**	159.1%	0.058	**0.118**	102.3%	0.063	**0.071**	13.7%
Recall	@5	0.030	**0.207**	**600.3%**	0.052	**0.218**	316.0%	0.063	**0.194**	209.6%	0.112	**0.120**	6.9%
	@10	0.059	**0.305**	**412.7%**	0.089	**0.325**	265.9%	0.109	**0.288**	163.1%	0.175	**0.191**	9.3%
	@15	0.085	**0.375**	**341.4%**	0.121	**0.394**	226.3%	0.150	**0.361**	141.2%	0.220	**0.245**	11.4%
	@20	0.111	**0.428**	**285.4%**	0.144	**0.450**	213.5%	0.184	**0.415**	125.3%	0.254	**0.293**	15.4%
nDCG	@5	0.043	**0.268**	**527.9%**	0.076	**0.274**	260.7%	0.087	**0.256**	196.0%	0.135	**0.143**	6.0%
	@10	0.053	**0.285**	**436.0%**	0.084	**0.297**	252.0%	0.099	**0.272**	175.6%	0.151	**0.162**	7.6%
	@15	0.062	**0.303**	**390.7%**	0.094	**0.315**	234.8%	0.110	**0.291**	163.7%	0.164	**0.178**	8.9%
	@20	0.071	**0.319**	**351.7%**	0.101	**0.332**	227.3%	0.121	**0.306**	153.5%	0.174	**0.193**	10.9%
MRR		0.106	**0.426**	**303.0%**	0.165	**0.428**	159.2%	0.181	**0.416**	129.3%	0.262	**0.274**	4.7%

recommendation stage. It takes about 1.54 seconds when it has the highest accuracy (θ=90%), which is 17% shorter than that of SVD.

In Figure 11(b), SVD_ZI takes more time to pre-process items with larger θ, and is slower than SVD and PureSVD. This is because SVD_ZI builds two models, one built based on the pre-use preference matrix P and the other based on the zero-injected matrix Z while both SVD and PureSVD build only a single model. SVD_ZI, however, needs less pre-computation time than SVD++ that has a more complicated process for building a model.

Figure 12 shows both recommendation and pre-computation times of ICF_ZI and ICF. The pre-computation time indicates the time for computing the similarities of all pairs of items. In Figure 12(a), when θ is smaller than 20%, the recommendation time of ICF_ZI increases as θ increases. This is because a more number of ratings are used for predicting a rating. It decreases linearly as θ increases when θ is higher than 20%. This is because the number of ratings used for prediction is fixed as k (explained in Section III-D) from this point while there remain a fewer items whose rating is null as θ is set larger. Compared with ICF, ICF_ZI requires less recommendation time when θ is larger than 70%. As we know that the accuracy of ICF_ZI gets higher as θ increases, a user would be satisfied with ICF_ZI when θ is set larger than 70% in terms of both accuracy and recommendation time.

Figure 12(b) shows the pre-computation time for computing similarities between items [19]. The pre-computation time of ICF_ZI increases as θ increases because a more number of ratings need to be compared to compute similarities of a pair of items. Therefore, ICF_ZI requires more pre-computation time than ICF does.

In summary, our approach reduces the recommendation times of SVD and ICF with $\theta > 70\%$ while it needs more pre-computation time. Considering that online recommendation

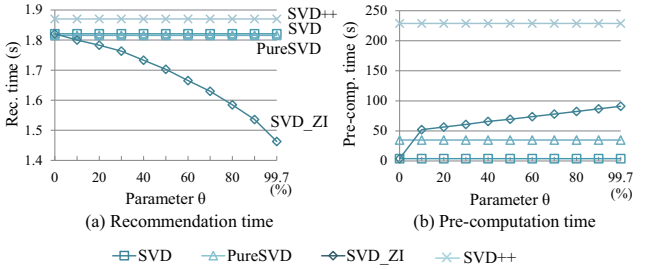

Fig. 11. Execution time of SVD variants.

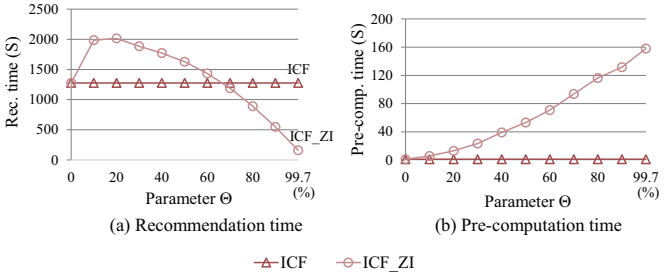

Fig. 12. Execution time for ICF variants.

time is more crucial than offline pre-computation time, we strongly believe that our approach improves those CF methods in terms of running times. As the answer to $Q5$, therefore, according to Figures 9, 11, and 12, the CF methods equipped with our approach produce *more accurate* results in a *shorter time* when θ is set higher than 70%.

V. RELATED WORK

In general, CF methods are categorized into two types: *memory-based* and *model-based* [1]. First, memory-based methods [3], [16], [19] predict the ratings of a user using the similarity of her neighborhoods, and recommend the items with high ratings. Second, model-based methods [11], [24] build a model capturing a users' ratings on items, and then predict her unknown ratings based on the learned model.

Most CF methods, despite their wide adoption in practice,

suffer from low accuracy if most users rate only a few items (thus producing a very sparse rating matrix), called the *data sparsity problem* [9]. This is because the number of unrated items is significantly more than that of rated items. To address this problem, some existing work attempts to infer users' ratings on unrated items based on additional information such as clicks [12] and bookmarks [14]. While reporting improvements, however, these works require an overhead of collecting extra data, which itself may have another data sparsity problem. Compared to [12], [14], our proposal does not require any extra data and solely work using an existing rating matrix.

In addition, to improve the accuracy of recommendation, other works attempted to leverage both ratings and the fact whether a user evaluates an item or not. For instance, SVD++ [2], [10] builds an extended SVD model exploiting both information. The conditional restricted Boltzmann machine (RBM) [17] and constrained probabilistic matrix factorization (PMF) [18] also account for both of information in learning their models. However, these approaches have a rather simplistic assumption such that a user would dislike all unrated items. On the other hand, we strive to discern a sebset of unrated items that users truly dislike. As such, as demonstrated in experiments, our proposal yields several orders of magnitude improvements in accuracy, compared to these appraoches (*e.g.*, SVD++).

Finally, several CF methods (*e.g.*, [4], [22]) have proposed to fill missing ratings with a particular value in order to improve the accuracy. They also simply assume that a user would dislike all unrated items. Based on this assumption, for instance, PureSVD [4] fills all missing ratings with zeros, and then makes prediction using both known ratings and zero ratings. Steck [22] assigns a low value to all missing ratings, and then makes recommendation by learning a multinomial mixture model. By filling *all* missing ratings with low values, however, this approach could mistakenly assign low values to the items that users might like, thereby hurting an overall accuracy in recommendation. While the idea of "zero-injection" in our approach is analogous to these works, our approach selectively applies the zero-injection to only uninteresting items with low pre-use preferences, maximizing the impact of filling missing ratings.

VI. Conclusion

While existing CF methods mainly focus on using only rated items in a rating matrix, we observe that some unrated items could be also used to predict users' ratings if they are successfully recognized as *uninteresting* items. Based on this observation, we proposed a novel approach to unearth such uninteresting items by using a new notion of *pre-use preferences* in the borrowed OCCF framework and assigns zero ratings to those items. This approach not only significantly augments a rating matrix with many zeros, which alleviates the data sparsity problem, but also prevents those uninteresting items from being recommended. Our approach is method-agnostic and thus can be easily applied to a wide variety of known CF methods. Through comprehensive experiments, we successfully demonstrated that our proposed approach is effective and practical, dramatically improving the accuracies of existing CF methods (*e.g.*, item-based CF, SVD-based CF, and SVD++) by 2.5 to 5 times. Furthermore, our approach reduces the running time of those CF methods by 1.2 to 2.3 times when its setting produces the best accuracy.

Acknowledgement

This work was in part supported by National Research Foundation of Korea (NRF-2014R1A2A1A10054151 and No. 2015R1C1A1A01055442), the Institute for Information & Communications Technology Promotion (IITP) grant (No. R2212150007000l002), and by NSF CNS-1422215 and Samsung 2015 GRO-175998 awards.

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] R. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[3] J. Breese et al. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, pages 43–52, 1998.

[4] P. Cremonesi et al. Performance of recommender algorithms on top-n recommendation tasks. In *Proc. of ACM RecSys*, pages 39–46, 2010.

[5] N. Dalvi et al. Para'normal' activity: On the distribution of average ratings. In *Proc. of AAAI ICWSM*, pages 110–119, 2013.

[6] Z. Gantner et al. Mymedialite: A free recommender system library. In *Proc. of ACM RecSys*, pages 305–308, 2011.

[7] J. Ha et al. Top-n recommendation through belief propagation. In *Proc. of ACM CIKM*, pages 2343–2346, 2012.

[8] N. Hu et al. Overcoming the j-shaped distribution of product reviews. *Communications of the ACM*, 52(10):144–147, 2009.

[9] Z. Huang et al. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM TOIS*, 22(1):116–142, 2004.

[10] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. of ACM KDD*, pages 426–434, 2008.

[11] Y. Koren et al. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[12] J. Liu et al. Personalized news recommendation based on click behavior. In *Proc. of ACM IUI*, pages 31–40, 2010.

[13] T. T. Nguyen et al. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proc. of IEEE WWW*, pages 677–686, 2014.

[14] S. Niwa et al. Web page recommender system based on folksonomy mining for ITNG '06 submissions. In *Proc. of IEEE ITNG*, pages 383–393, 2006.

[15] R. Pan et al. One-class collaborative filtering. In *Proc. of IEEE ICDM*, pages 502–511, 2008.

[16] P. Resnick et al. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. of ACM CSCW*, pages 175–186, 1994.

[17] R. Salakhutdinov et al. Restricted boltzmann machines for collaborative filtering. In *Proc. of ACM ICML*, pages 791–798, 2007.

[18] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proc. of NIPS*, pages 1257–1264, 2007.

[19] B. Sarwar et al. Item-based collaboration filtering recommendation algorithms. In *Proc. of IEEE WWW*, pages 285–295, 2001.

[20] V. Sindhwani et al. A family of non-negative matrix factorization for one-class collaborative filtering. In *Proc. of ACM RecSys*, 2009.

[21] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proc. of AAAI ICML*, pages 720–727, 2003.

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

[23] H. Yin et al. Modeling location-based user rating profiles for personalized recommendation. *TKDD*, 9(3):19, 2015.

[24] S. Zhang et al. Using singular value decomposition approximation for collaborative filtering. In *Proc. of IEEE CEC*, pages 257–264, 2005.