l-Injection: Toward Effective Collaborative Filtering Using Uninteresting Items

Jongwuk Lee Won-Seok Hwang Juan Parc Youngnam Lee Sang-Wook Kim Dongwon Lee

(Invited Paper)

Abstract—We develop a novel framework, named as l-injection, to address the sparsity problem of recommender systems. By carefully injecting low values to a selected set of unrated user-item pairs in a user-item matrix, we demonstrate that top-*N* recommendation accuracies of various collaborative filtering (CF) techniques can be significantly and consistently improved. We first adopt the notion of *pre-use preferences* of users toward a vast amount of *unrated* items. Using this notion, we identify *uninteresting* items that have not been rated yet but are likely to receive low ratings from users, and selectively impute them as low values. As our proposed approach is method-agnostic, it can be easily applied to a variety of CF algorithms. Through comprehensive experiments with three real-life datasets (*e.g.*, Movielens, Ciao, and Watcha), we demonstrate that our solution consistently and universally enhances the accuracies of existing CF algorithms (*e.g.*, item-based CF, SVD-based CF, and SVD++) by 2.5 to 5 times on average. Furthermore, our solution improves 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 the experiments are available at: https://goo.gl/KUrmip.

1 Introduction

The goal of recommender systems (RS) is to suggest appealing items (*e.g.*, movies, books, or news articles) to a user by analyzing her prior preferences. As a large number of online applications use RS as a core component, improving the quality of RS becomes a critically important problem to businesses. Among existing solutions in RS, in particular, *collaborative filtering* (*CF*) methods (*e.g.*, [2], [3], [4], [5], [6], [7]) have been shown to be widely effective. Based on the past behavior of users such as explicit user ratings and implicit click logs, CF methods exploit the similarities between users' behavior patterns.

However, when the fraction of known ratings in a rating matrix R is overly small (so-called *data sparsity* problem), CF methods tend to suffer. For an R with m users and n items, if we assume that each user has rated k items on average, the fraction of rated items in R is $\frac{k}{n}$ (= $\frac{m \times k}{m \times n}$). Asymptotically, such a fraction of rated items in R is extremely small (*i.e.*, $k \ll n$). It is common for an e-business to sell millions of items with a very *long tail*, and many users rate very few items (*i.e.*, *cold-start* users). The goal of this work is to mitigate such a data sparsity problem to improve top-*N* recommendation accuracies of CF methods. Our proposal is based on the following hypothesis in CF:

- *This paper builds on and significantly extends our prior work [1].*
- *J. Lee is with the Department of Software, Sungkyunkwan University, Republic of Korea.*
- *W. Hwang, J. Park, Y. Lee, and S. Kim (corresponding author) are with the Department of Computer and Software, Hanyang University, Republic of Korea.*
- *D. Lee is with the College of Information Sciences and Technology, The Pennsylvania State University, PA, USA.*
  E-mail: {jongwuklee@skku.edu, hws23, crystaldia, utopianami, wook}@hanyang.ac.kr, dongwon@psu.edu

TABLE 1
Rating distributions of three real-life datasets.

Dataset	Low ratings (1 or 2)	High ratings (3, 4, or 5)
Movielens	17%	83%
Ciao	10%	90%
Watcha	13%	87%

Hypothesis 1. *Filling some values into empty cells, i.e., unrated items, in a rating matrix R can improve the accuracy of CF methods for top-N recommendation.*

We first argue that ratings in R be often a reflection of the satisfaction of users. Therefore, users tend to rate (high) only the items that they like, and those who are dissatisfied tend *not* to rate items in R. Corroborating this point, Table 1 illustrates severe imbalance between low (*i.e.*, 1 or 2) and high (*i.e.*, 3, 4, or 5) ratings from three real-life datasets that we used in our experiments. Note that only a small fraction (*i.e.*, 10–17%) of ratings are low values. Then, a natural question to raise is: *how can we identify the unknown opinions of those users who were dissatisfied with and did not leave ratings for items*?

To answer this question, note that unrated items in R can be classified into three different types: (1) unrated items whose existence users were not aware of, (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. We note that the unrated items of the third type, called *uninteresting items* (denoted by I^{un}), clearly indicate users' latent *negative* preferences on them. Therefore, it is better *not* to recommend those uninteresting items. In order to identify such uninteresting items, we propose to use a new notion of *pre-use preference*, *i.e.*, an impression of items *before* purchasing and using them. That is, by definition, uninteresting items indicate the items with low pre-use preferences. Unfortunately, the ratings in R do

not indicate pre-use preferences but the preferences *after* using the items, called *post-use preference*.

Based on this novel notion of pre-use preference and uninteresting item, we develop a solution that consists of three steps: (1) infer the pre-use preferences of unrated items by solving the *one-class collaborative filtering* (*OCCF*) problem [8], [9], (2) assign "low" values to uninteresting items in R, yielding an augmented matrix L, and (3) apply existing CF methods to L, instead of R, to recommend top-N appealing items. This simple-yet-novel imputation solution significantly alleviates the data sparsity problem by augmenting R. Extending our prior work [1], in this work, we develop a more general l-injection to infer different user preferences for uninteresting items for users, and show that l-injection mostly outperforms 0-injection in [1].

The proposed l-injection approach can improve the accuracy of top-N recommendation based on two strategies: (1) preventing uninteresting items from being included in the top-N recommendation, and (2) exploiting both uninteresting and rated items to predict the relative preferences of unrated items more accurately. With the first strategy, because users are aware of the existence of uninteresting items but do not like them, such uninteresting items are likely to be false positives if included in top-N recommendation. Therefore, it is effective to exclude uninteresting items from top-N recommendation results. Next, the second strategy can be interpreted using the concept of typical memory-based CF methods. Suppose that a few neighbors of a user u rated an item i high but most neighbors of u considered i uninteresting (thus left i unrated in R). In this case, existing CF methods tend to recommend i to user u. However, if many neighbors of u consider i as an uninteresting item, we should avoid recommending i to u.

To summarize, our main contributions are as follows:

- We introduce a new notion of *uninteresting* items, and classify user preferences into *pre-use* and *post-use* preferences to identify uninteresting items.
- We propose to identify uninteresting items via pre-use preferences by solving the OCCF problem and show its implications and effectiveness.
- We propose *low-value injection* (called *l-injection*) to improve the accuracy of top-N recommendation in existing CF algorithms.
- We evaluate the proposed solution with three real-life datasets, and demonstrate that our solution consistently outperforms baseline CF methods (*e.g.*, item-based CF, SVD-based CF, and SVD++) with respect to accuracy (by 2.5 to 5 times) and running time (by 2.5 to 5 times) on average.

The remainder of this paper is organized as follows. In Section 2, we explain the preliminaries of our approach. In Section 3, we present our approach. In Section 4, we evaluate our approach by comparing it with existing methods via extensive experiments. In Section 5, we review related work. In Section 6, we conclude our work.

2 Preliminaries

In general, CF methods have been studied under two settings: (1) predicting the ratings of unrated items, and (2) recommending top-N unrated appealing items to users. In this paper, we focus on the top-N recommendation setting, which is more practical in real-world applications [10].

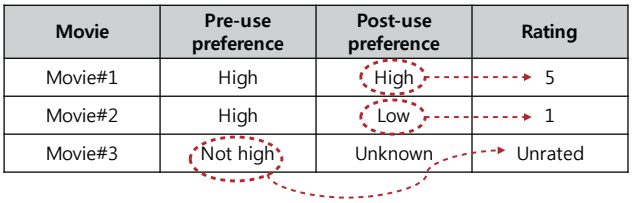


Fig. 1. Pre-use and post-use preferences for three movies.

We first explain some 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} (respectively q_{ui}) indicates the *pre-use* (respectively *post-use*) preference for item i of user u. The pre-use preference p_{ui} is different from a known rating r_{ui} in the rating matrix R, implying post-use preference q_{ui}. In theory, both types of preferences p_{ui} and q_{ui} exist as a user-item pair (u, i) although they are not always available.

Note that it is possible to infer the pre-use preference for item $i \in I$ of user u from its *external* features, (*e.g.*, genre, director, or actors, in the case of a movie). After using i, based on the level of her satisfaction, u then assigns a specific score to i, indicating her post-use preference for i. Therefore, the post-use preference is determined by the *inherent* features that u had not known before using i (*e.g.*, storyline or choreography of a movie). Let us explain both types of preferences using the following example.

Example 1 (Two preference types). Fig. 1 illustrates the pre-use and post-use preferences of a user u for three movies. Initially, user u has a high pre-use preference for Movie #1 and Movie #2. On the other hand, u does not have a high pre-use preference for Movie #3. In this case, Movie #1 and Movie #2 are to be called *interesting* items while Movie #3 an *uninteresting* item. Thus, u would decide to watch only the two movies for which she has high pre-use preferences. After watching the movies, u likes Movie #1 as expected, but does not like Movie #2. Therefore, u assigns a high rating to Movie #1 and a low rating to Movie #2 as her post-use preferences. In contrast, the user does not watch Movie #3 as it is an uninteresting movie to her. The post-use preference for Movie #3 thus remains *unknown*, *i.e.*, missing in the rating matrix.

Fig. 2 further illustrates the preferences of a user u for an entire set I of items. I_u^{in} denotes *interesting* items with high pre-use preferences while I_u^{un} denotes *uninteresting* items with low pre-use preferences. Two item sets are disjoint, *i.e.*, $I_u^{in} \cap I_u^{un} = \emptyset$, $I_u^{in} \cup I_u^{un} = I$. We formally define uninteresting items as follows:

Definition 1 (Uninteresting items). *For a user u, a set of uninteresting items I_u^{un} is defined as: $I_u^{un} = I - I_u^{in}$. The following inequality for pre-use preferences holds: $\forall i \in I_u^{un}, \forall j \in I_u^{in} : p_{ui} \leq p_{uj}$.*

A user u purchases some items out of interesting items I_u^{in} and rates them. A set of items that are likely to receive

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2698461, IEEE Transactions on Knowledge and Data Engineering

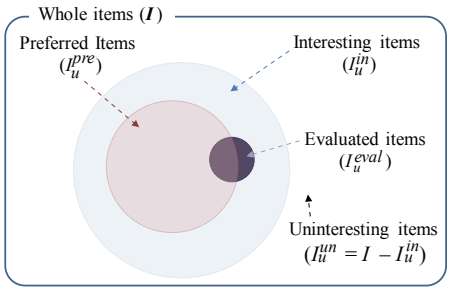


Fig. 2. Venn diagram for the preferences and interestingness of items.

high ratings, (*i.e.*, high post-use preferences) is called *preferred items*, denoted by I_u^{pre}. Therefore, it is a subset of interesting items as shown in Fig. 2 (*i.e.*, $I_u^{pre} \subseteq I_u^{in}$). In a real scenario, u would be able to rate 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 identify the uninteresting items of each user, we can understand the user's taste more accurately.

Based on this viewpoint, our goal is to identify the *top-N preferred items* of a user u by considering the "latent" uninteresting items of u. In particular, 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. If u has evaluated item i, the 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, the post-use preference q_{ui} can be directly indicated by estimating the score of r_{ui}. The top-N recommendation is formally defined as follows:

Problem 1 (Top-N recommendation) *For user u, we aim to identify the top-*N *unrated items $J = \{j_1, ..., j_N\}$ such that: (1) $J \subseteq I_u^{in} - I_u^{eval}$ and (2) $q_{uj_1} \geq ... \geq q_{uj_N} \geq q_{uy}$ ($\forall y \in I - I_u^{eval} - J$).*

3 Proposed Approach

While existing CF methods only employ user preferences on rated items, the proposed approach employs both pre-use and post-use preferences. Specifically, the proposed approach first infers pre-use preferences of unrated items (Section 3.1) and identifies uninteresting items I_u^{un} (Section 3.2). Then, it enriches the rating matrix by exploiting uninteresting items (Section 3.3). The existing CF methods equipped with our approach not only benefit from the enriched matrix and but also exclude the uninteresting items from top-N recommendation. Lastly, we analyze the benefits of the proposed approach on account of improving the accuracy greatly (Section 3.4).

The main challenges of our approach are as follows: (1) *how to identify uninteresting items among unrated items* and (2) *how to exploit uninteresting items discerned in CF methods*. To address the first challenge, we infer pre-use preferences for all unrated items and find the unrated items whose pre-use preferences are low. For the second challenge, we build an augmented matrix where some missing entries are imputed by *low* values if their corresponding items are considered uninteresting. The augmented matrix can be applied to *any* CF method (thus making our approach method-agnostic), which enables existing CF methods to benefit from uninteresting items in their top-N recommendation.

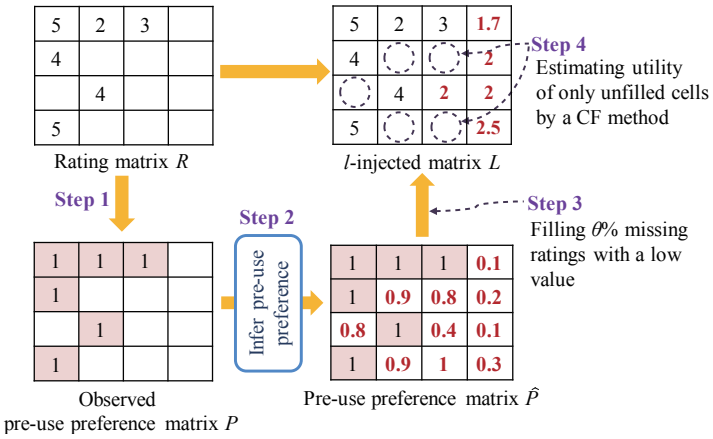


Fig. 3. Overall process of the proposed approach.

Fig. 3 depicts the overall processes of the proposed approach. First, we build a *pre-use preference matrix* $P = (p_{ui})_{m \times n}$ by examining a rating matrix $R = (r_{ui})_{m \times n}$. It is set as one if $r_{ui} \in R$ has been already rated (*i.e.*, u should have liked i if she bought i) (Step 1). It is the highest because p_{ui} is set as a real value in [0, 1]. Next, we infer pre-use preference scores 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). Based on $\hat{P}$, we identify uninteresting items for each user and build a *low-value injected matrix* $L = (l_{ui})_{m \times n}$ (Step 3). That is, if r_{ui} in R is unrated and item i is an uninteresting item for user u, it is imputed by l_{ui}. In the proposed approach, i is determined as the uninteresting item for u if the pre-use preference score $\hat{p}_{ui}$ is ranked in the bottom θ% in $\hat{P}$. The augmented matrix L thus includes both the original ratings for rated items and the imputed ratings for uninteresting items. Lastly, existing CF algorithms are applied to the augmented matrix L (Step 4). We recommend top-N items by predicting the post-use preferences of empty entries (dotted circles). In the following subsections, we explain each step in detail.

3.1 Inferring Pre-Use Preferences

It is straightforward to determine a pre-use preference p_{ui} if a user u has already rated an item i (*i.e.*, $r_{ui} \neq null$). This is because i may have been interesting to u at first consideration, *i.e.*, $I_u^{eval} \subseteq I_u^{in}$. As such, we set the pre-use preference p_{ui} as one. However, when u has not rated i (*i.e.*, $r_{ui} = null$), it is non-trivial to determine p_{ui}. Therefore, it is essential to infer pre-use preferences p_{ui} if r_{ui} is unrated.

To address this challenge, we borrow the framework of the *one-class collaborative filtering* (OCCF) problem [8], [9]. The OCCF problem occurs when a rating score is *unary* such as clicks, bookmarks, and purchases so that a cell $r_{ij} \in R$ has a null value or a single value indicating "yes." The ambiguity arises from the interpretation of unrated items. That is, it is difficult to distinguish *negative* and *positive* examples that co-exist among unrated items [8]. Some unrated

items can be positive because the user is not aware of their existence. On the other hand, some are negative because the user knows about the items but dislikes them. Therefore, she determines not to use them.

This problem setting also happens when we infer pre-use preferences for unrated items. That is, known pre-use preferences for rated items have positive values (*i.e.*, $p_{ui} = 1$) and missing pre-use preferences for unrated items are ambiguous. In Fig. 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 [8] to infer pre-use preferences. In Section 4.2, we also demonstrate that the OCCF method is the most effective to infer users' pre-use preferences.

The basic idea of the OCCF method is to treat all unrated items as negative examples and to assign weights to quantify the relative contribution of these examples. In our situation, the OCCF method assigns 0 to every 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, which was the best performer in [8]. The underlying principle of the user-oriented scheme is essentially that *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 $p_{ui} \in P$ using their corresponding weights. We treat the updated values as the inferred pre-use preference scores.

To update these values, the OCCF method employs the weighted alternating least squares (wALS) method [11] in building a singular value decomposition (SVD) with a rating matrix and its weight matrix. 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 latent features of users and items, respectively. The objective function is represented as follows:

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

where p_{ui} and w_{ui} are the entries in the observed pre-use preference matrix P and its 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 matrix P, the OCCF method first assigns random values to elements in matrix Y, and updates elements in matrix X as in Eq. (2) by optimizing the objective function. $\forall 1 \leq u \leq m$:

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

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

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

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 matrix $\widehat{P}$ represents a pre-use preference score of user u for item i.

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

3.2 Identifying Uninteresting Items

Once pre-use preferences of unrated items are computed, we can identify uninteresting items. Based on the pre-use preference scores inferred by the OCCF method, the uninteresting items of user u 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 example, $I_u^{un}(20)$ 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 unrated entries are injected with low values, leading to a less sparse rating matrix. On the other hand, if θ is set low, we may not be fully utilizing the benefit of uninteresting items as only a small number of unrated entries are injected. The simple use of *relative* cut-off based on the percentile rank works well. (In Section 4.4, we will evaluate the effectiveness of the cut-off method.)

Example 2 (Uninteresting items). Fig. 4 illustrates a pre-use preference matrix $\hat{P}$, where the cells with one and with decimal numbers are originally derived from the rated and unrated items in R, respectively. For a larger θ, more items are considered 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 clear contrast, our approach can find a large number of uninteresting items that users have not recognized yet but are likely to consider uninteresting.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2698461, IEEE Transactions on Knowledge and Data Engineering

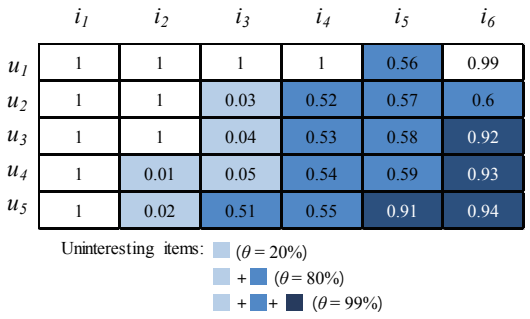


Fig. 4. Identifying uninteresting items from pre-use preference matrix $\hat{P}$.

3.3 l-Injection

We now propose a novel method to impute missing ratings, named as *l-injection*, such that we assign a "low" value to $r_{ui} \in R$ if an item i is determined as uninteresting for a user u. This is because u would not be satisfied with an uninteresting item i even if recommended.

By filling a rating matrix R with low values, we can build a new "denser" matrix that contains low value ratings as well as actual user ratings. We call this augmented matrix an *l-injected matrix* $L = (l_{ui})_{m \times n}$, where entry l_{ui} is defined as follows:

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

where v_{ui} is to be defined below. We now develop various methods to impute missing ratings to uninteresting items. (In Section 4.4, we evaluate the effectiveness of various imputation methods for uninteresting items.) To determine v_{ui}, a simple way is to fill it with zero [1]. This imputation means that a user does not like uninteresting items at all. Alternatively, because uninteresting items are generally less preferred than rated items, we can also fill a "low" value by *under-estimating* the average of known ratings. To calculate the global average for rated items, we define an indicator y_{ui} for the existence of r_{ui} as follows:

$$y_{ui} = \begin{cases} 1 & \text{if } u \text{ has rated } i; \\ 0 & \text{otherwise} \end{cases}$$

When we use the global average of ratings, v_{ui} is defined as:

$$v_{ui} = \frac{\sum_{x \in U, j \in I} y_{xj} r_{xj}}{\sum_{x \in U, j \in I} y_{xj}} \times \delta \tag{6}$$

where $\delta \in [0, 1]$ is used to control the degree of uninterestingness compared to the average. It is also possible to use the average of ratings per user/item. For the user average, v_{ui} is computed by:

$$v_{ui} = \frac{\sum_{j \in I} y_{uj} r_{uj}}{\sum_{j \in I} y_{uj}} \times \delta \tag{7}$$

For the item average, v_{ui} is computed by:

$$v_{ui} = \frac{\sum_{x \in U} y_{xi} r_{xi}}{\sum_{x \in U} y_{xi}} \times \delta \tag{8}$$

Note that the proposed approach works regardless of the choice of underlying CF methods as it simply replaces the original rating matrix R by the l-injected matrix L. The proposed approach is *orthogonal to existing CF methods*, which is one of our key strengths. It is also possible to develop other imputation methods to reflect the characteristic of uninteresting items. Because our intention is to evaluate the effectiveness of using uninteresting items for top-N recommendation, we leave more sophisticated modeling for l-injection as our future work.

The proposed approach can improve existing CF methods with three aspects. First, when CF methods are applied, uninteresting items are *excluded* from the recommendation list. While existing CF methods consider *all* items whose ratings are missing as the candidates for top-N recommendation, we essentially avoid uninteresting items from top-N recommendation. That is, the l-injected matrix can prevent uninteresting items from top-N recommendation. Second, the l-injected matrix includes a higher number of ratings (including ratings with low values) for uninteresting items than the original rating matrix. The CF algorithms equipped with an l-injected matrix are able to understand users' preferences more accurately. Lastly, because the number of candidates for top-N recommendation essentially reduces, the computational cost can also decrease in top-N recommendation.

We further discuss a key difference from existing work. Similar to an l-injected matrix, PureSVD [7] also assigns zero to missing ratings. However, PureSVD has no regard for identifying uninteresting items and simply fills zero values to "all" missing ratings. That is, PureSVD simply regards all unrated items as uninteresting items. In addition, PureSVD considers the items with missing ratings as candidates for top-N recommendation. In clear contrast, the l-injection selectively fills the uninteresting items I_u^{un} with low values, to help understand user preferences more precisely, and exclude uninteresting items from top-N recommendation. In Section 4, we demonstrate the superiority of our approach over PureSVD.

3.4 Why Does the l-Injected Matrix Help?

We argue that an l-injected matrix helps improve the accuracy of any CF method. To present the ground for our argument, we discuss the effect of our approach when applied to two popular CF methods: *item-based collaborative filtering method (ICF)* [4] and *SVD-based method (SVD)* [5].

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

$$\hat{l}_{ui} = \frac{\sum_{j \in S_i} \{l_{uj} * sim(i, j)\}}{\sum_{j \in S_i} sim(i, j)} \tag{9}$$

where S_i is a set of (up to) k items which have most similar rating patterns to the items for which item i of user u is known. If there are less than k items evaluated by u, S_i includes that number of items only instead of k. In addition, let $sim(i, j)$ denote the similarity between items i and j in terms of users' rating patterns. In this paper, we adopt Pearson's correlation coefficient as the well-known similarity measure [2], [12].

Example 3 (l-injected matrix). Fig. 5 illustrates the difference between a rating matrix R and its corresponding l-injected matrix L. We observe that, unlike R, L has extra

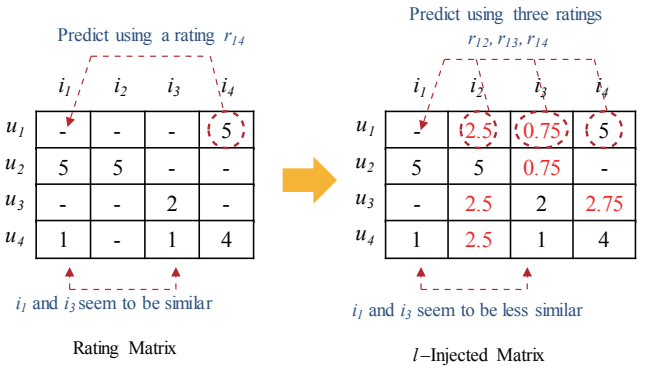


Fig. 5. Comparison of an original rating matrix and its corresponding l-injected matrix.

low-value ratings (in red color) 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 because u_1 has not rated i_2 and i_3 (*i.e.*, r_{12} and r_{13} are nulls). In contrast, owing to our l-injection, ICF can consider more ratings, *i.e.*, l_{12}, and l_{13} in the l-injected matrix, which can help improve the accuracy of ICF.

The l-injected matrix can help find 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, because it is based on only a single user's opinion, the similarity can be inaccurate. With L, the two items are regarded much less similar because u_2 rated the two items very differently. As such, an l-injected matrix is useful to compute the similarity between users more accurately because it enables CF to reflect more users' opinions. In particular, we note that the l-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.

Next, we explain how an l-injected matrix makes existing SVD-based methods [5] more accurate. Given L, it is factorized 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. Each item i is involved with an f-dimensional vector y_i. The rating prediction for $\hat{l}_{ui}$ is computed by:

$$\hat{l}_{ui} = x_u y_i^T \tag{10}$$

With the original rating matrix R in Fig. 5, SVD cannot recognize that u_2 is related to u_1 or u_3 because they have no common items rated. In contrast, when using L, it can successfully observe 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. In addition, it can overlook the relationships between items such as i_2 and i_3 with R while it can find the relationship between i_2 and i_3 by using L.

4 Experiments

In this section, we evaluate the accuracy and efficiency of the proposed approach. We first validate the accuracy of the OCCF method [8], which is used to infer pre-use preferences of items. We then compare several methods to inject low values for uninteresting items and examine the sensitivity of a parameter θ meaning the ratio of uninteresting items out of all unrated items. Finally, we compare the proposed methods equipped with an l-injected matrix against existing CF methods.

TABLE 2
Statistics of three real-life datasets

	MovieLens	Ciao	Watcha
Density	6.30%	1.38%	3.77%
Min. # of ratings of users	20	1	16
Max. # of ratings of users	737	319	513
Avg. # of ratings per user	106.04	18.72	72.66
# of total ratings	100,000	18,648	101,073

4.1 Experimental Setup

We employ three real-life datasets: (1) *MovieLens* 100K [6] includes 943 users, 1,682 items, and 100,000 ratings; (2) *Ciao* [13] consists of 996 users, 1359 items, and 18,648 ratings; (3) *Watcha* consists of 1,391 users, 1,927 items, and 101,073 ratings. All ratings take integer values ranging from 1 (worst) to 5 (best). As the latest dataset, the Watcha dataset is privately released from a Korean movie recommendation company (http://watcha.net). Because the datasets have clear differences for density, the number of ratings, and rating distributions, it is effective for cross-checking the accuracy of the proposed algorithms. Table 2 reports statistics of three datasets. If the dataset is not explicitly mentioned in experiments, we use the "MovieLens" dataset by default, which has been widely used.

For top-N recommendation, we vary the value of N from 5 to 20 in an increment of 5 (default N = 5). It is also possible to recommend more items by extending N. Because users are usually interested a few items, we focus on evaluating small N. (When extending N, it is found that the tendency of accuracy is still consistent.) We only consider the items with 5 (best) as *relevant* items, *i.e.*, *ground truth*, because it is most effective for top-N recommendation, as discussed in [14]. That is, correctly predicting items with the highest ratings leads to positive business ramifications.

We adopt four metrics to measure the accuracy of top-N recommendation, namely, *precision*, *recall*, *normalized discounted cumulative gain* (*nDCG*), and *mean reciprocal rank* (*MRR*). For a user u, precision $P_u@N$ and recall $R_u@N$ are computed by $\frac{|Rel_u \cap Rec_u|}{|Rec_u|}$ and $\frac{|Rel_u \cap Rec_u|}{|Rel_u|}$, respectively. Let Rec_u denote a set of N items recommended to u, and Rel_u denote a set of items considered relevant. nDCG is used 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 one. Otherwise, y_k is set as zero. 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 $IDCG_u@N$ is an *ideal* $DCG_u@N$ where y_k is set as one for every item $i_k \in Rec_u$. MRR is computed as the average inverse rankings of every item $i_k \in Rec_u$. For user u, MRR_u is computed by $\frac{1}{|Rel_u|} \sum_{i=1}^{|Rel_u|} \frac{1}{rank_i}$. We performed four cross-validations for all experimental results.

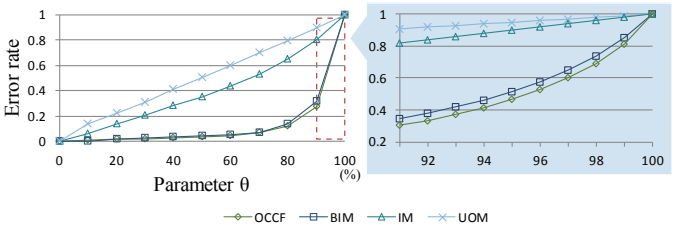


Fig. 6. Comparison on error rates of four inference methods.

TABLE 3
Accuracy (*i.e.*, P@5) of ICF and SVD equipped with our approach using four inference methods.

CF method	Parameter Θ	Inference Method			
		OCCF	BIM	IM	UOM
ICF	30%	**0.209**	0.145	0.113	0.057
	60%	**0.210**	0.155	0.124	0.016
	90%	**0.216**	0.160	0.128	0.002
Average		**0.212**	0.154	0.122	0.025
SVD	30%	**0.205**	0.144	0.121	0.066
	60%	**0.215**	0.153	0.150	0.034
	90%	**0.235**	0.194	0.147	0.013
Average		**0.218**	0.164	0.139	0.038

To evaluate the accuracy of the proposed approach, we first augment a rating matrix R to an l-injected matrix L, and feed L as input to existing CF methods such as item-based CF (ICF) [4] and SVD-based CF (SVD) [5]. We adopted the ICF and SVD algorithms implemented in the open-source MyMediaLite [15] with their default parameter settings. All datasets and codes that we used are available at: https://goo.gl/KUrmip.

The goal of our empirical study is to answer the following key questions through comprehensive evaluation.

- **Q1**: Is the OCCF method (most) effective to infer users' pre-use preferences?
- **Q2**: Are users not satisfied with uninteresting items that our approach infers?
- **Q3**: How does the accuracy of our approach change over injected values and the ratios of uninteresting items? How mich sensitive is it to them?
- **Q4**: How much does our approach enhance the accuracy of existing CF methods? Is our hypothesis valid?
- **Q5**: How much running time does our approach spend compared to existing CF methods?

4.2 Inference of Pre-Use Preferences

We validate how effective the OCCF method is against three candidates: *user-oriented method* (*UOM*), *binary item-based method* (*BIM*), and *item-based method* (*IM*). First, UOM determines pre-use preference scores of users to be inversely proportional to the number of rated items. Second, BIM uses the observed (binary) pre-use preference matrix P and infers pre-use preference scores by using item-based CF [4]. Third, IM indicates item-based CF, which produces pre-use preference scores from original rating matrix R. For the OCCF method, we followed the same parameter settings for wALS as in [8].

We define an *error rate* to quantify the ratio of misclassified items out of all rated items. It is calculated as: $err_u^\theta = \frac{|I_u^{un}(\theta) \cap I_u^{test}|}{|I_u^{test}|}$. I_u^{test} is a set of items *rated* by u in a test set, and $I_u^{un}(\theta)$ is a set of uninteresting items (*i.e.*, ranked in the bottom θ% according to the inferred pre-use preference scores). The lower the error rate, the better is the inference method.

Fig. 6 depicts the result for comparing error rates of four inference methods. As θ increases, the error rates of all methods increase as well. The error rates of UOM and IM increase more rapidly than those of the OCCF method and BIM. In contrast, the OCCF method and BIM show relatively small error rates until θ reaches 90%, implying that their accuracies are 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 the four methods, the OCCF method shows the best error rates regardless of θ.

We further examine if pre-use preferences inferred by the OCCF method indeed yield the best accuracy. We first build an l-injected matrix L using pre-use preferences, and apply L to two CF methods: *item-based CF* (*ICF*) and *SVD-based CF* (*SVD*). We vary θ as 30%, 60%, and 90%. Because all accuracy metrics show similar tendencies, we only report the results for P@5. (More detailed analysis on the effect of θ will be given in Section 4.4.) Table 3 shows the P@5 of ICF and SVD with the l-injected matrix. As shown in Fig. 6, the OCCF method consistently shows the best accuracy in all cases (19%–26% higher than BIM, the second best one) while UOM shows the worst accuracy.

As for **Q1**, we conclude that the OCCF method is most effective for inferring pre-use preferences. Therefore, we adopt the OCCF method to infer pre-use preferences in subsequent evaluation.

4.3 User Satisfaction for Uninteresting Items

Our main assumption is that a user would not be satisfied with her uninteresting items. To justify this assumption, we examine how much users are satisfied with items in proportion to their pre-use preference scores. Recall that an item is likely to be selected as an uninteresting item if its pre-use preference score is relatively low.

We first hide the ratings of items in the test set and compute the pre-use preference scores for all unrated items by using the ratings of items in the training set as done in Section 4.2. We then partition all 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 actually rated in the test set. That is, $I_u^{un}(\beta_j, \beta_{j+1})$ 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 $\forall i \in I_u^{un}(\beta_j, \beta_{j+1})$).

Fig. 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 Fig. 7). This result indicates that users hardly ever rate the items whose pre-use preference scores are low. Therefore, our assumption holds.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2698461, IEEE Transactions on Knowledge and Data Engineering

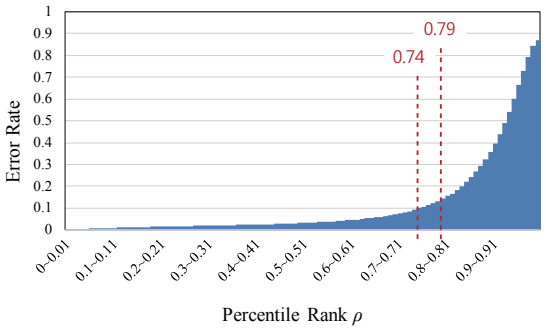


Fig. 7. The change of error rates for pre-use preference scores.

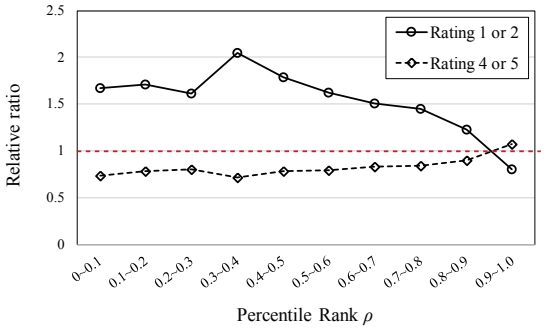


Fig. 8. Distribution of pre-use preference scores for rated items.

Next, we analyze rated items according to their pre-use preference scores. We divide user-item pairs into 10 bins according to their percentile rank ρ of pre-use preference scores. For rated items in 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 fair comparison, we note two important observations: (1) 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; (2) 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 have them in the last 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:

$$ratio_u(j,s) = \frac{|I_u^{test}(s) \cap I_u^{un}(\beta_j, \beta_{j+1})|}{|I_u^{test} \cap I_u^{un}(\beta_j, \beta_{j+1})|} \div \frac{|I_u^{eval}(s)|}{|I_u^{eval}|} \quad (11)$$

where I_u^{test} is a set of items rated 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}. In addition, I_u^{eval} indicates a set of items rated by u among all items, and $I_u^{eval}(s)$ is a set of items rated by u as s in I_u^{eval}. The fraction before the division sign in Eq. 11 means the ratio of items rated as s by u to all rated items by u in a bin. The fraction after the division sign in Eq. 11 means the ratio of items rated as s by u to the whole items rated by u and is used for normalization. A higher relative ratio indicates that more items rated as s exist in a bin.

Fig. 8 shows the relative ratios of rated items. When ρ is smaller than 0.3, the relative ratio of items rated as 1 or 2 stagnates. 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. 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. That is, the items whose ρ is smaller than 0.9 are more likely to be rated as 1 or 2 than 4 or 5. Meanwhile, the items whose ρ is higher than 0.9 are more likely to be rated as 4 or 5. Users are less likely to be satisfied with the items whose ρ is less than 0.9.

As for **Q2**, we conclude that users are less satisfied with the items whose pre-use preferences are low. In addition, it is found that users tend to be unsatisfied with most of the items in R (*e.g.*, 90%), implying that most of items can be uninteresting to users.

4.4 Effect of l-Injection

We compare several methods of imputing low values for uninteresting items. The baseline method is to simply inject *zero* for uninteresting items, *i.e.*, $v_{ri} = 0$, which is proposed in our preliminary work [1]. Alternatively, we utilize the global average of ratings and the average of ratings per user/item. Because uninteresting items are unlikely to be preferred, their ratings should be set relatively low. We inject a value by under-estimating the averages, *i.e.*, $v_{ui} = average \times \delta \in [0.25, 0.50, 0.75, 1.00]$.

Tables 4 and 5 report the accuracies of ICF [4] and SVD [5] with l-injection matrix L. To build l-injected matrix L, various imputation methods can be employed. The gray color indicates the best accuracy over varying δ and θ. In both algorithms, the imputation methods of using averages outperform that of using zero. When $\delta = 0.5$, it shows the best performance regardless of θ. Meanwhile, when $\delta < 0.5$ or $\delta > 0.5$, the accuracies of ICF and SVD tend to decrease. This implies that users may rate uninteresting items as relatively low values, but they would not extremely dislike them if rated.

Next, we conduct the sensitivity test to evaluate the effect of θ. Fig. 9 shows the accuracy of top-N ($N = 5$) recommendation with ICF and SVD over varying θ. We increase θ in an increment of 10% for the range of $10-90\%$, while increasing 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. The result with $\theta = 0\%$ indicates the accuracy of original ICF and SVD methods without using our approach. Meanwhile, when setting θ up to 99.7%, we only leave top-N items whose pre-use preference scores are the highest for each user. In this case, *all* remaining items are thus selected as a top-N recommendation list (*i.e.*, top-5) to each user without employing CF methods.

In Fig. 9, we observe that the results of precision, recall, nDCG, and AUC show similar patterns. The accuracies of all CF methods increase as θ increases up to around 95%. Moreover, they grow quite rapidly until θ reaches 10%. All results clearly show that our idea of using l-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. Similarly, when $\theta = 95\%$, our approach improves the precision of SVD by *3.4 times*.

The accuracy changes considerably when θ is less than 10% or more than 90% while it changes much less when θ is

TABLE 4
Accuracy (*i.e.*, P@5) of ICF equipped with our approach over varying imputation methods for an l-injected matrix.

l-Value		Parameter (θ)			
		0	30	60	90
0		0.039	0.199	0.199	0.201
User_avg	δ=0.25		0.208	0.204	0.207
Item_avg			0.208	0.207	0.210
All_avg			0.205	0.206	0.206
User_avg	δ=0.5		0.209	**0.210**	0.214
Item_avg			0.209	**0.210**	**0.216**
All_avg			**0.210**	0.209	0.214
User_avg	δ=0.75		0.196	0.200	0.212
Item_avg			0.179	0.187	0.206
All_avg			0.202	0.204	0.214
User_avg	δ=1.00		0.139	0.144	0.179
Item_avg			0.003	0.056	0.136
All_avg			0.171	0.177	0.203
5			0.000	0.001	0.018

TABLE 5
Accuracy (*i.e.*, P@5) of SVD equipped with our approach over varying imputation methods for an l-injected matrix.

l-Value		Parameter (θ)			
		0	30	60	90
0		0.063	0.177	0.198	0.207
User_avg	δ=0.25		0.191	0.206	0.222
Item_avg			0.192	0.191	0.223
All_avg			0.192	0.207	0.221
User_avg	δ=0.5		0.204	0.216	**0.235**
Item_avg			**0.205**	0.215	**0.235**
All_avg			0.204	**0.217**	0.234
User_avg	δ=0.75		0.204	0.215	0.228
Item_avg			0.193	0.205	0.219
All_avg			0.203	0.218	0.227
User_avg	δ=1.00		0.138	0.159	0.187
Item_avg			0.018	0.077	0.141
All_avg			0.161	0.176	0.200
5			0.013	0.013	0.031

between 10% and 90%. This phenomenon can be interpreted as follows: (1) as θ increases up to 10%, more user-item pairs (*i.e.*, highly likely to be uninteresting) are filled and are also *correctly* excluded from top-N recommendation; (2) when θ ranges between 10% and 90%, the accuracy changes less because filling unrated user-item pairs in the case of (1) has already alleviated most of the data sparsity problem. Filling more ratings no longer gives useful information to CF although user-item pairs whose θ is between 10% and 90% are highly likely to be uninteresting (See Section 4.3); (3) when θ is larger than 90%, the accuracy decreases significantly. As θ reaches 99.7%, more user-item pairs with *high* pre-use preference scores (*i.e.*, could be interesting to users) are incorrectly filled by mistake giving inaccurate and less useful information to CF methods. In addition, they could be *incorrectly* excluded from top-N recommendation.

As for **Q3**, we conclude that the injection of using averages is useful. In addition, it is found that a very high accuracy can be achieved even if θ is set within a large interval of $10\% \leq \theta \leq 95\%$, indicating that our approach is *parameter-insensitive* (with respect to θ).

4.5 Accuracy of Modified CF Algorithms

We apply our approach to four existing CF methods (*i.e.*, ICF [4], SVD [5], *SVD++* [16], [17], and *pureSVD* [7]). SVD++ utilizes both ratings and binary ratings (just indicating whether a user evaluates an item or not). PureSVD fills *all* missing ratings with zeros, and then produces top-N recommendation based on the SVD model. Based on the findings in Section 4.4, we set the parameter θ as 90% in our approach.

Tables 6–8 report the accuracy of all CF methods *with* and *without* our approach in three real-life datasets. We denote a CF method equipped with our approach as *name_LI* (*i.e.*, l-injection) such as ICF_LI, SVD_LI, SVD++_LI, and PureSVD_LI. The numbers in bold indicate the highest accuracy among all CF methods with and without our approach. We also compare our approaches with OCCF [8] and MNAR [14], exploiting unrated items.

Among existing CF methods, PureSVD has the best accuracy while ICF shows the worst accuracy. Both PureSVD and SVD++ are known to provide a better accuracy than SVD and ICF. It is found that our experimental results are consistent with [7]. We also observe that our approach *dramatically improves* the accuracies of all 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. This idea is similar to to an l-injection in the sense that some unrated items are not interested to the users. That is, SVD_LI performs the best, followed by ICF_LI among the CF methods equipped with our proposed approach. Our approach improves both SVD and ICF considerably because they regard all unrated items as unknown ones. For this reason, SVD_LI and ICF_LI adopt additional information correctly by regarding low-value ratings as uninteresting ones and null values as unknown ones. Because OCCF and MNAR also consider unrated items, they show a better accuracy than existing CF methods. Because the SVD equipped with l-injection considers uninteresting items more effectively, it shows better than OCCF and MNAR. These results are consistent with three datasets.

However, our approach does not improve SVD++ and PureSVD effectively, because they originally have a positive view on rated items and a negative view on unrated items. SVD++ builds an SVD model by considering whether a user rates an item. Meanwhile, SVD++_LI injects uninteresting items as low values. That is, while SVD++ considers all unrated items as uninteresting items, SVD++_LI has

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2698461, IEEE Transactions on Knowledge and Data Engineering

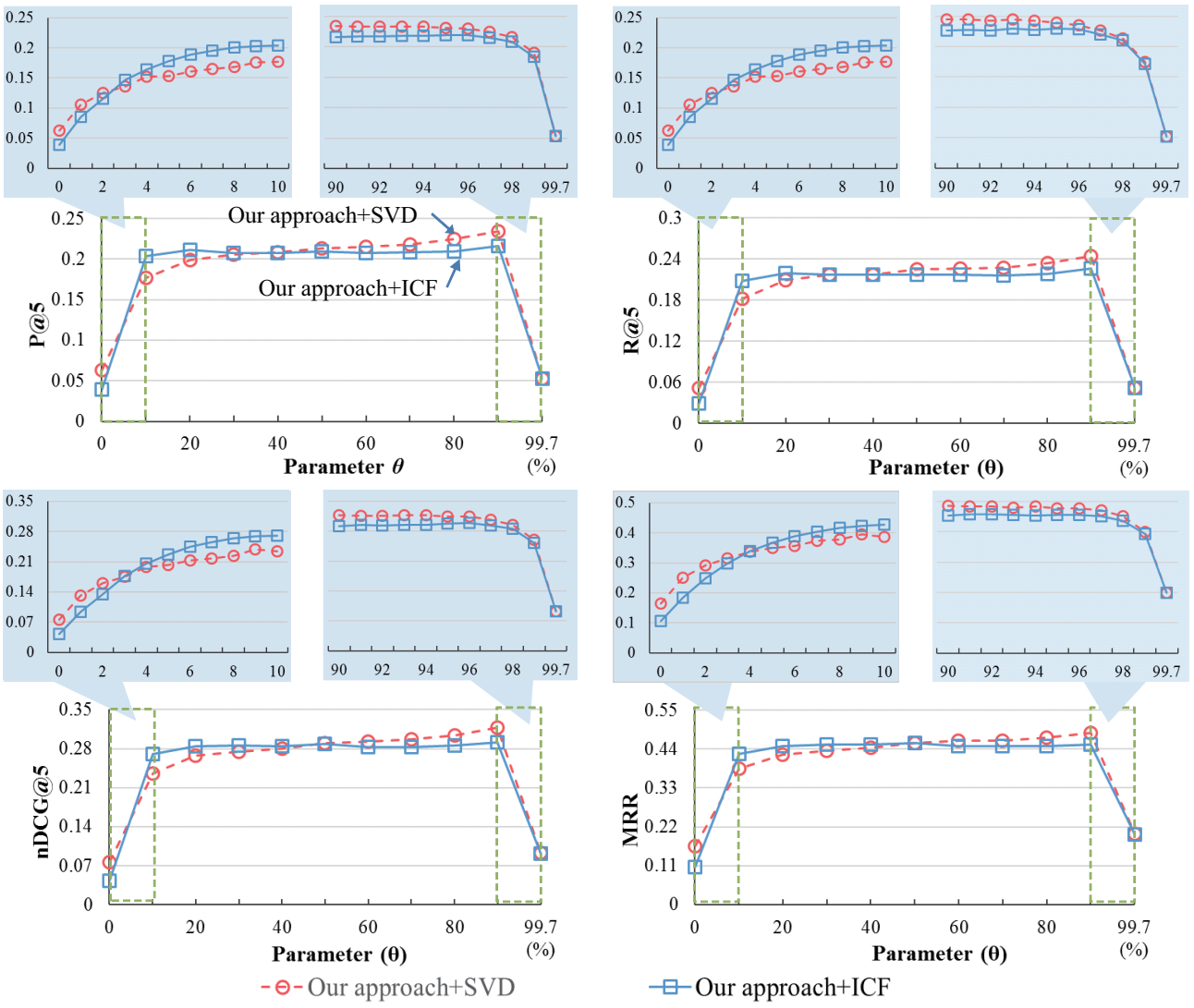


Fig. 9. Accuracy of ICF and SVD equipped with our proposed approach over varying parameter θ.

already injected uninteresting items as rated items with low values. This means that the unrated items between SVD++ and SVD++_LI have a conflict. Some interesting items among unrated items can be regarded as uninteresting items. SVD++ with the l-injected matrix does not effectively capture the negative view for unrated items. Similarly, because PureSVD simply fills zeros even for the items that *could be interesting* to users, it could affect the accuracy adversely. For PureSVD, we impute the same values as the l-injected matrix to unrated items. Similar to SVD++, it may not distinguish between uninteresting items and unrated items. Because we have another strategy of excluding uninteresting items from top-N recommendation, it can achieve a better accuracy than PureSVD.

We further evaluate the accuracy of CF methods for extreme cases, *e.g.*, *long-tail* items and *cold-start* users. Because top-N recommendation can be biased to item popularity [7], [18], it is difficult to achieve high accuracy for long-tail items, *i.e.*, unpopular items with a few ratings. In addition, the cold-start users [19] mean the users who have rated only a few items. For the MovieLens dataset, we define an item to be an long-tail item if its number of ratings is less than 100. Because the items are unpopular, users tend to prefer such long-tail items to be recommended [20]. We also define a user to be a cold-start user if her number of rated items is less than 10, 15, and 20. Compared to the average users who have rated more than 100 items in Table 2, it is much more difficult to infer hidden preferences of cold-start users. (These definitions for long-tail items and cold-start users are consistent with these of existing work [7], [18], [19], [20].)

Table 9 reports the accuracy of CF methods for long-tail items. For all metrics, it is found that the CF methods equipped with our approach can consistently improve the accuracy of top-N recommendation. In addition, it is also found that using our approach can help improve the accuracy of existing CF methods regardless of N. For $NDCG@5$, we can achieve the highest improvement of three to six times.

Table 10 reports the accuracy of CF methods for cold-start users. It is found that our approach improves the accuracy of top-N recommendation. When the number of ratings per user is 10, the improvement gap is smallest. However, as the number of ratings per user increases, it increases as well. This is because the proposed approach can

TABLE 6
Accuracy of four CF methods equipped with our approach in the MovieLens dataset ($\theta = 90\%$).

Metric		ICF			SVD			SVD++			PureSVD			OCCF	MNAR
		Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain		
Precision	@5	0.039	**0.216**	**453.3%**	0.063	**0.235**	272.3%	0.076	**0.174**	129.0%	0.169	**0.227**	34.1%	0.192	0.213
	@10	0.041	**0.169**	**311.2%**	0.056	**0.180**	221.6%	0.069	**0.141**	105.0%	0.142	**0.177**	24.8%	0.151	0.169
	@15	0.040	**0.144**	**259.4%**	0.053	**0.152**	187.2%	0.063	**0.123**	94.6%	0.122	**0.150**	22.6%	0.129	0.143
	@20	0.039	**0.127**	**226.3%**	0.048	**0.133**	177.5%	0.058	**0.109**	88.2%	0.109	**0.130**	19.9%	0.115	0.124
Recall	@5	0.030	**0.226**	**653.8%**	0.052	**0.244**	370.0%	0.063	**0.181**	186.8%	0.177	**0.237**	34.0%	0.202	0.224
	@10	0.059	**0.324**	**449.8%**	0.089	**0.344**	286.6%	0.109	**0.271**	148.9%	0.280	**0.341**	21.7%	0.299	0.327
	@15	0.085	**0.398**	**368.6%**	0.121	**0.415**	242.9%	0.150	**0.339**	126.0%	0.346	**0.412**	19.0%	0.367	0.398
	@20	0.111	**0.454**	**309.4%**	0.144	**0.469**	225.7%	0.184	**0.396**	115.0%	0.397	**0.462**	16.6%	0.421	0.448
nDCG	@5	0.043	**0.292**	**578.3%**	0.076	**0.318**	318.4%	0.087	**0.232**	166.8%	0.217	**0.312**	43.3%	0.254	0.286
	@10	0.053	**0.307**	**478.8%**	0.084	**0.331**	293.8%	0.099	**0.250**	152.3%	0.243	**0.327**	34.8%	0.272	0.305
	@15	0.062	**0.326**	**425.0%**	0.094	**0.348**	270.1%	0.110	**0.268**	143.7%	0.261	**0.344**	32.1%	0.290	0.323
	@20	0.071	**0.342**	**382.1%**	0.101	**0.363**	259.8%	0.121	**0.285**	135.2%	0.276	**0.359**	29.8%	0.306	0.337
MRR		0.106	**0.453**	**327.8%**	0.165	**0.485**	194.1%	0.181	**0.384**	112.3%	0.393	**0.487**	24.1%	0.453	0.445

TABLE 7
Accuracy of four CF methods equipped with our approach in the Ciao dataset ($\theta = 90\%$).

Metric		ICF			SVD			SVD++			PureSVD			OCCF	MNAR
		Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain		
Precision	@5	0.007	**0.030**	**317.1%**	0.005	**0.031**	521.0%	0.014	**0.017**	24.7%	0.009	**0.024**	150.4%	0.028	0.026
	@10	0.007	**0.023**	**250.8%**	0.004	**0.024**	468.0%	0.012	**0.014**	17.4%	0.006	**0.018**	197.0%	0.023	0.022
	@15	0.006	**0.020**	**250.2%**	0.004	**0.021**	415.3%	0.011	**0.012**	12.6%	0.005	**0.015**	205.9%	0.019	0.019
	@20	0.005	**0.018**	**232.5%**	0.004	**0.019**	373.9%	0.010	**0.011**	16.3%	0.004	**0.014**	226.0%	0.017	0.017
Recall	@5	0.017	**0.062**	**264.0%**	0.010	**0.062**	548.1%	0.028	**0.038**	34.0%	0.021	**0.048**	130.0%	0.056	0.056
	@10	0.030	**0.095**	**218.7%**	0.018	**0.097**	433.0%	0.050	**0.061**	23.2%	0.025	**0.073**	185.7%	0.093	0.092
	@15	0.038	**0.125**	**228.3%**	0.026	**0.124**	382.3%	0.067	**0.081**	20.0%	0.032	**0.096**	196.8%	0.121	0.119
	@20	0.047	**0.146**	**208.6%**	0.032	**0.147**	358.3%	0.082	**0.100**	21.7%	0.036	**0.117**	221.3%	0.143	0.142
nDCG	@5	0.013	**0.052**	**295.0%**	0.008	**0.054**	602.3%	0.022	**0.032**	46.5%	0.020	**0.041**	103.8%	0.048	0.049
	@10	0.018	**0.064**	**254.8%**	0.011	**0.067**	523.1%	0.029	**0.040**	36.3%	0.021	**0.049**	129.8%	0.061	0.062
	@15	0.021	**0.073**	**254.8%**	0.013	**0.075**	476.9%	0.035	**0.046**	31.5%	0.024	**0.057**	139.4%	0.070	0.071
	@20	0.023	**0.080**	**241.4%**	0.015	**0.082**	447.5%	0.039	**0.052**	31.5%	0.025	**0.063**	152.1%	0.077	0.078
MRR		0.032	**0.092**	**187.4%**	0.022	**0.099**	360.5%	0.047	**0.062**	30.6%	0.029	**0.076**	161.9%	0.069	0.091

infer hidden user preferences more accurately as the number of rated items increases. For instance, when the number of ratings per user is 20, our approach is shown to improve the accuracy of the original ICF by four times.

As for **Q4**, we conclude that our approach improves the accuracy of existing CF methods by 2.5 to 5 times. This improvement is significant in comparison with the results obtained by other state-of-the-art methods [7], [10]. In addition, our approach improves the accuracy of existing CF methods consistently even for the extreme cases of long-tail items and cold-start users.

4.6 Running Time of Modified CF Algorithms

Finally, we compare the running times of CF methods with and without our approach. We note the weaknesses of our approach incur at the pre-computation stage (offline) while the strengths of our approach are at the recommendation stage (online). The CF methods build a model or compute similarities of item/user pairs during the pre-computation stage, and compute top-N items during the recommendation stage. Our approach can reduce the recommendation time because it significantly reduces the number of candidate items by excluding uninteresting items from top-N recommendation. Meanwhile, our approach may require more pre-computation time because it has to infer pre-use preference scores for all unrated items.

We examine the trade-off between the running time of the pre-computation stage and the accuracy of the recommendation stage. Fig. 10 shows both recommendation and pre-computation times of SVD_LI, SVD, SVD++, and PureSVD. The recommendation time includes the running times for predicting users' ratings and providing the items to users, and the pre-computation time does those for building a model with rating matrix R (SVD, SVD++, and PureSVD) and l-injected matrix L (SVD_LI). In Fig. 10(a), the recommendation time of SVD_LI decreases rapidly as θ increases because *fewer* candidate items remain as θ increases. In addition, SVD_LI requires a shorter time at the recommendation stage. It takes about 1.54 seconds when

TABLE 8
Accuracy of four CF methods equipped with our approach in the Watcha dataset ($\theta = 90\%$).

Metric		ICF			SVD			SVD++			PureSVD			OCCF	MNAR
		Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain		
Precision	@5	0.008	**0.083**	**895.2%**	0.022	**0.085**	284.1%	0.026	**0.061**	136.6%	0.038	**0.082**	116.7%	0.061	0.084
	@10	0.008	**0.069**	**717.4%**	0.020	**0.069**	254.1%	0.022	**0.053**	139.4%	0.029	**0.067**	128.8%	0.053	0.067
	@15	0.009	**0.060**	**597.2%**	0.017	**0.061**	248.7%	0.020	**0.047**	128.9%	0.025	**0.058**	134.9%	0.047	0.058
	@20	0.008	**0.054**	**537.7%**	0.016	**0.055**	235.3%	0.019	**0.043**	123.9%	0.022	**0.051**	136.6%	0.043	0.052
Recall	@5	0.009	**0.127**	**1331.8%**	0.028	**0.131**	361.1%	0.032	**0.092**	189.4%	0.054	**0.123**	128.2%	0.091	0.123
	@10	0.019	**0.203**	**978.3%**	0.050	**0.208**	313.3%	0.054	**0.156**	191.5%	0.080	**0.194**	142.6%	0.156	0.197
	@15	0.029	**0.261**	**808.7%**	0.067	**0.266**	296.0%	0.074	**0.204**	177.4%	0.097	**0.249**	157.5%	0.208	0.250
	@20	0.037	**0.310**	**740.9%**	0.082	**0.316**	285.8%	0.091	**0.245**	168.4%	0.110	**0.291**	164.1%	0.253	0.296
nDCG	@5	0.009	**0.125**	**1216.4%**	0.029	**0.127**	330.7%	0.037	**0.091**	143.4%	0.055	**0.121**	120.6%	0.089	0.124
	@10	0.013	**0.151**	**1058.3%**	0.037	**0.153**	315.7%	0.044	**0.112**	157.6%	0.063	**0.145**	128.2%	0.111	0.148
	@15	0.017	**0.170**	**923.1%**	0.042	**0.173**	307.3%	0.051	**0.129**	154.1%	0.069	**0.163**	134.8%	0.129	0.165
	@20	0.020	**0.186**	**851.6%**	0.048	**0.189**	298.2%	0.057	**0.142**	150.9%	0.074	**0.177**	138.4%	0.143	0.180
MRR		0.033	**0.233**	**611.3%**	0.076	**0.234**	207.8%	0.095	**0.185**	94.8%	0.164	**0.226**	37.7%	0.202	0.231

TABLE 9
Accuracy of CF methods for long-tail items.

Metric		ICF			SVD		
		Orginal	Ours	Gain	Orginal	Ours	Gain
Precision	@5	0.016	**0.105**	**545.3%**	0.031	**0.115**	272.0%
	@10	0.018	**0.087**	**393.3%**	0.027	**0.092**	237.1%
	@15	0.018	**0.075**	**319.1%**	0.025	**0.078**	208.1%
	@20	0.018	**0.066**	**267.9%**	0.024	**0.069**	190.9%
Recall	@5	0.022	**0.184**	**730.2%**	0.045	**0.199**	339.9%
	@10	0.023	**0.143**	**520.5%**	0.038	**0.149**	295.9%
	@15	0.035	**0.182**	**416.9%**	0.052	**0.186**	254.3%
	@20	0.047	**0.207**	**341.4%**	0.065	**0.212**	227.6%
nDCG	@5	0.017	**0.135**	**673.1%**	0.037	**0.148**	304.7%
	@10	0.022	**0.143**	**557.0%**	0.039	**0.153**	289.8%
	@15	0.026	**0.152**	**485.2%**	0.043	**0.160**	270.5%
	@20	0.030	**0.159**	**426.9%**	0.047	**0.167**	256.3%
MRR		0.057	**0.255**	**343.6%**	0.088	**0.273**	209.7%

TABLE 10
Accuracy of CF methods for cold-start users.

#Ratings	ICF			SVD		
	Orginal	Ours	Gain	Orginal	Ours	Gain
10	0.025	**0.073**	**192.0%**	0.037	**0.076**	105.4%
15	0.024	**0.090**	**275.0%**	0.045	**0.092**	104.4%
20	0.020	**0.101**	**405.0%**	0.047	**0.101**	114.9%

it has the highest accuracy (θ=90%), which is 17% shorter than that of SVD. In Fig. 10(b), SVD_LI takes more time for pre-processing with larger θ, and is slower than SVD and PureSVD. This is because SVD_LI builds two models, one built based on the pre-use preference matrix and the other built on l-injected matrix while both SVD and PureSVD build only a single model. However, SVD_LI requires less pre-computation time than SVD++.

Fig. 11 shows both recommendation and pre-computation times of ICF_LI and ICF. The pre-computation time indicates the time for computing the similarities of all pairs of items. In Fig. 11(a), when θ is smaller than 20%, the recommendation time of ICF_LI 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 (as explained in Section 3.4) while a fewer number of items remain. Compared with ICF, ICF_LI requires less recommendation time when θ is larger than 70%. As we know that the accuracy of ICF_LI gets higher as θ increases, a user would be satisfied with ICF_LI when θ is set larger than 70% in terms of both accuracy and recommendation time. Fig. 11(b) shows the pre-computation time for computing similarities between items [4]. The pre-computation time of ICF_LI increases with θ. This is because more numbers of items are needed to compute similarities of a pair of items. Therefore, ICF_LI requires more pre-computation time than ICF does.

As to **Q5**, we conclude that our approach reduces the recommendation times of SVD and ICF with $\theta > 70\%$ while it needs more pre-computation time. Considering that online recommendation time is more crucial than offline pre-computation time, our approach can improve existing CF methods for online recommendation time.

4.7 Summary of Experimental Results

Based on the experimental results, we answer the key questions **Q1–Q5**). For **Q1**, the OCCF method is the most effective for inferring pre-use preference scores, compared to other methods: UOM, IM, and BIM. For **Q2**, users are unlikely to be satisfied with uninteresting items with low pre-use preference scores. For **Q3**, l-injection using average values is more effective than 0-injections for most existing algorithms. For **Q4**, our approach equipped with an l-injected matrix can improve the accuracy of existing CF methods by 2.5 to 5 times. For **Q5**, our approach can also save the running time of the recommendation using SVD and ICF while it needs more pre-computation time.

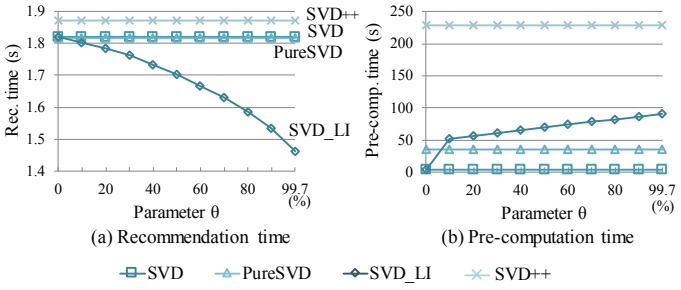


Fig. 10. Running time of SVD variants.

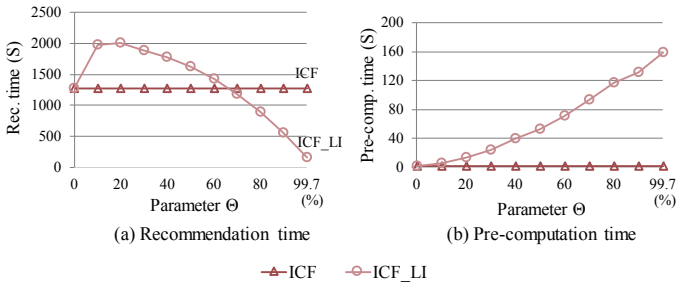


Fig. 11. Running time for ICF variants.

5 Related Work

In general, CF methods are categorized into two approaches: *memory-based* and *model-based* [2]. First, memory-based methods [4], [6], [12] predict the ratings of a user using the similarity of her neighborhoods, and recommend the items with high ratings. Second, model-based methods [3], [5] 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* [21]. This is because the number of unrated items is significantly more than that of rated items. To address this problem, some existing work attempted to infer users' ratings on unrated items based on additional information such as clicks [22] and bookmarks [23]. However, these works require an overhead of collecting extra data, which itself may have another data sparsity problem. Compared to [22], [23], our proposal does not require any extra data and solely works on account of an existing rating matrix.

In addition, to improve the accuracy of top-N recommendation, other works leverage both ratings and the fact whether a user evaluates an item or not. For instance, SVD++ [16], [17] builds an extended SVD model exploiting both information. The conditional restricted Boltzmann machine (RBM) [24] and constrained probabilistic matrix factorization (PMF) [25] also account for both information in learning their models. However, these approaches are based on a simple assumption such that a user would dislike *all* unrated items. On the other hand, we strive to discern a subset of unrated items that users truly dislike. Therefore our proposal yields improvements in accuracy compared to existing methods.

Finally, several CF methods (*e.g.*, [7], [14]) have been 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, PureSVD [7] fills all missing ratings with zeros, and then makes prediction using both known ratings and zero ratings. Steck [14] 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 affecting an overall accuracy in recommendation. Our preliminary work [1] infers uninteresting items and builds 0-injected matrix. Because the 0-injected matrix includes the ratings inferred from uninteresting items, it can infer latent user preferences more accurately. However, because 0-injection simply considers all uninteresting items as zero, it may neglect to the characteristics of users or items. In contrast, l-injection not only maximizes the impact of filling missing ratings but also considers the characteristics of users and items, by imputing uninteresting items with low pre-use preferences.

6 Conclusions

In this paper, we proposed a novel approach, l-injection, for *uninteresting* items by using a new notion of *pre-use preferences*. This approach not only significantly alleviates the data sparsity problem but also effectively prevents those uninteresting items from being recommended. Because the proposed approach is method-agnostic, it can be easily applied to a wide variety of existing CF methods. Through comprehensive experiments, we successfully demonstrated that the 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 improves the running time of those CF methods by 1.2 to 2.3 times when its setting produces the best accuracy.

Acknowledgements

This work was in part supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2014R1A2A1A10054151 and No. 2015R1C1A1A01055442).

References

[1] W. Hwang, J. Parc, S. Kim, J. Lee, and D. Lee, "Told you i didn't like it: Exploiting uninteresting items for effective collaborative filtering," in *Proc. of IEEE ICDE*, 2016, pp. 349–360.

[2] 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*, vol. 17, no. 6, pp. 734–749, 2005.

[3] Y. Koren *et al.*, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] B. Sarwar *et al.*, "Item-based collaboration filtering recommendation algorithms," in *Proc. of IEEE WWW*, 2001, pp. 285–295.

[5] S. Zhang *et al.*, "Using singular value decomposition approximation for collaborative filtering," in *Proc. of IEEE CEC*, 2005, pp. 257–264.

[6] P. Resnick *et al.*, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proc. of ACM CSCW*, 1994, pp. 175–186.

[7] P. Cremonesi *et al.*, "Performance of recommender algorithms on top-n recommendation tasks," in *Proc. of ACM RecSys*, 2010, pp. 39–46.

[8] R. Pan *et al.*, "One-class collaborative filtering," in *Proc. of IEEE ICDM*, 2008, pp. 502–511.

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

[10] J. Ha *et al.*, "Top-n recommendation through belief propagation," in *Proc. of ACM CIKM*, 2012, pp. 2343–2346.

[11] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proc. of AAAI ICML*, 2003, pp. 720–727.

[12] J. Breese *et al.*, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. of UAI*, 1998, pp. 43–52.

[13] J. Tang, H. Gao, and H. Liu, "mtrust: discerning multi-faceted trust in a connected world," in *"Proc. of WSDM", pages = 93–102, year = 2012*.

[14] H. Steck, "Training and testing of recommender systems on data missing not at random," in *Proc. of ACM KDD*, 2010, pp. 713–722.

[15] Z. Gantner *et al.*, "Mymedialite: A free recommender system library," in *Proc. of ACM RecSys*, 2011, pp. 305–308.

[16] R. Bell and Y. Koren, "Lessons from the netflix prize challenge," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 75–79, 2007.

[17] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proc. of ACM KDD*, 2008, pp. 426–434.

[18] H. Steck, "Item popularity and recommendation accuracy," in *Proc. of ACM RecSys*, 2011, pp. 125–132.

[19] H. Ma *et al.*, "Effective missing data prediction for collaborative filtering," in *Proc. of ACM SIGIR*, 2007, pp. 39–46.

[20] J. Lee, D. Lee, Y. Lee, W. Hwang, and S. Kim, "Improving the accuracy of top-n recommendation using a preference model," *Inf. Sci.*, vol. 348, pp. 290–304, 2016.

[21] Z. Huang *et al.*, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM TOIS*, vol. 22, no. 1, pp. 116–142, 2004.

[22] J. Liu *et al.*, "Personalized news recommendation based on click behavior," in *Proc. of ACM IUI*, 2010, pp. 31–40.

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

[24] R. Salakhutdinov *et al.*, "Restricted boltzmann machines for collaborative filtering," in *Proc. of ACM ICML*, 2007, pp. 791–798.

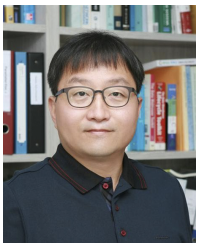
[25] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. of NIPS*, 2007, pp. 1257–1264.

Jongwuk Lee is currently an assistant professor at Department of Software, Sungkyunkwan University, Korea. Before that, he was an assistant professor at Hankuk University of Foreign Studies, Korea. In 2012, he received his Ph.D. in Computer Science and Engineering at Pohang University of Science and Technology, Korea. His research interests include data mining, databases, recommender systems, information retrieval, and Web mining.

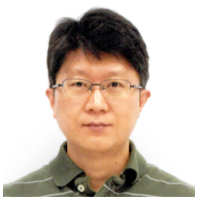
Won-Seok Hwang is currently a researcher at National Security Research Institute. In 2016, he received his Ph.D. degree in Electronics and Computer Engineering from Hanyang University, Korea. His research interests include recommendation, data mining, machine learning, and network security.

Juan Parc acquired his B.S. and M.S. degrees in computer science from Hanyang University in 2013 and 2015, respectively. He is now working as a research engineer for Intelligence Lab in LG Electronics, Korea. His areas of interest include recommender systems, databases, and data mining.

Youngnam Lee is currently a researcher for development for event prediction such as cardiac arrest and sepsis based on deep learning at VUNO. In 2017, he received his M.S. degrees in Computer Software from Hanyang University, Korea. His research interests include recommender systems, machine learning, and deep learning.



Sang-Wook Kim received the B.S. degree in Computer Engineering from Seoul National University at 1989, and earned the M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) at 1991 and 1994, respectively. From 1995 to 2003, he served as an Associate Professor at Kangwon National University. In 2003, he joined Hanyang University, Seoul, Korea, where he currently is a Professor at the Department of Computer Science and Engineering and the director of the Brain-Korea-21-Plus research program. He is also leading a National Research Lab (NRL) Project funded by National Research Foundation from 2015. From 2009 to 2010, Dr. Kim visited Computer Science Department at Carnegie Mellon University as a Visiting Professor. From 1999 to 2000, he worked with the IBM T. J. Watson Research Center, USA as a Post-Doc. He also visited the Computer Science Department of Stanford University as a Visiting Researcher in 1991. He is an author of over 200 papers in refereed international journals and international conference proceedings. His research interests include databases, data mining, multimedia information retrieval, social network analysis, recommendation, and web data analysis. He is a member of the ACM and the IEEE.



Dongwon Lee is currently a program director at National Science Foundation (NSF), USA, co-managing cybersecurity education programs such as SFS and SaTC-EDU with the yearly budget of $50M. He is also a faculty member at Penn State University, USA, currently on leave. He has published over 140 articles in competitive venues, researching broadly in Data Science, in particular, on the management of and mining in data in diverse forms including structured records, text, multimedia, social media, and Web. He received his B.S., M.S., and Ph.D. degrees in Computer Science from Korea University (1993), Columbia University (1995), and UCLA (2002), respectively. From 1995 to 1997, he has worked at AT&T Bell Labs., NJ. Further details of his research can be found at: http://pike.psu.edu/