Regularizing Matrix Factorization with User and Item Embeddings for Recommendation

Thanh Tran, Kyumin Lee
Worcester Polytechnic Institute, USA
{tdtran,kmlee}@wpi.edu

Yiming Liao, Dongwon Lee
Penn State University, USA
{yiming,dongwon}@psu.edu

ABSTRACT

Following recent successes in exploiting both latent factor and word embedding models in recommendation, we propose a novel *Regularized Multi-Embedding* (RME) based recommendation model that simultaneously encapsulates the following ideas via decomposition: (1) which items a user *likes*, (2) which two users *co-like* the same items, (3) which two items users often *co-liked*, and (4) which two items users often *co-disliked*. In experimental validation, the RME outperforms competing state-of-the-art models in both explicit and implicit feedback datasets, significantly improving Recall@5 by 5.9~7.0%, NDCG@20 by 4.3~5.6%, and MAP@10 by 7.9~8.9%. In addition, under the *cold-start* scenario for users with the lowest number of interactions, against the competing models, the RME outperforms NDCG@5 by 20.2% and 29.4% in MovieLens-10M and MovieLens-20M datasets, respectively. Our datasets and source code are available at: https://github.com/thanhdtran/RME.git.

KEYWORDS

Recommendation; item embeddings; user embeddings; negative sampling; collaborative filtering.

ACM Reference Format:
Thanh Tran, Kyumin Lee and Yiming Liao, Dongwon Lee. 2018. Regularizing Matrix Factorization with User and Item Embeddings for Recommendation. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271730

1 INTRODUCTION

Among popular *Collaborative Filtering* (CF) methods in recommendation [14, 17, 29, 33], in recent years, latent factor models (LFM) using matrix factorization have been widely used. LFM are known to yield relatively high prediction accuracy, are language independent, and allow additional side information to be easily incorporated and decomposed together [1, 35]. However, most of conventional LFM only exploited positive feedback while neglected negative feedback and treated them as missing data [8, 14, 27, 34].

In movie recommender systems, it was observed that many users who enjoyed watching *Thor: The Dark World*, also enjoyed *Thor: Ragnarok*. In this case, *Thor: The Dark World* and *Thor: Ragnarok* can be seen as a pair of co-liked movies. So, if a user preferred *Thor: The Dark World* but never watch *Thor: Ragnarok*, the system can precisely recommend *Thor: Ragnarok* to her (**first observation**). Similarly, if two users A and B liked the same movies, we can assume A and B have the same movie interests. If user A likes a movie that B has never watched, the system can recommend the movie to B (**second observation**). In the same manner, we ask if co-occurred disliked movies can provide any meaningful information. We observed that most users, who rated *Pledge This!* poorly (0.8/5.0 on average), also gave a low rating to *Run for Your Wife* (1.3/5.0 on average). If the disliked co-occurrence pattern was exploited, *Run for Your Wife* would not be recommended to other users who did not enjoy *Pledge This!* (**third observation**). This will help reduce the false positive rate for recommender systems. The same phenomena would have also occurred in other recommendation domains.

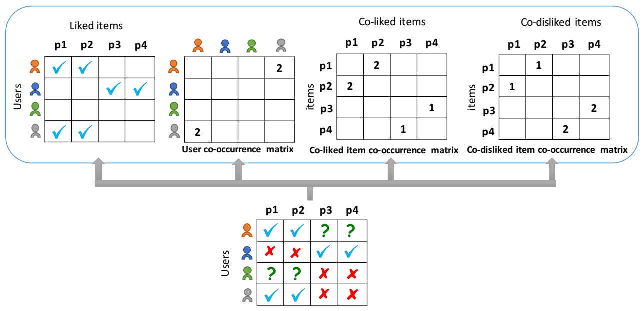

Liked items
p1 p2 p3 p4
Users
User co-occurrence matrix
Co-liked items
items
Co-liked item co-occurrence matrix
Co-disliked items
Co-disliked item co-occurrence matrix

Figure 1: An overview of our RME Model, which jointly decomposes user-item interaction matrix, co-liked item co-occurrence matrix, co-disliked item co-occurrence matrix, and user co-occurrence matrix. (V: liked, X: disliked, and ?: unknown)

The first two observations are similar to the basic assumptions of item CF and user CF where similar scores between items/users are used to infer the next recommended items for users. Unfortunately, only the first two observations have been exploited in conventional CF. While treating the negative-feedback items differently from missing data led to better results [13], to the best of our knowledge, no previous works exploited the **third observation** to enhance the recommender systems' performance.

Therefore, in this paper, we attempt to exploit all three observations in one model to achieve better recommendation results. With the recent success of word embedding techniques in natural language processing, if we consider pairs of co-occurred liked/disliked items or pairs of co-occurred users as pairs of co-occurred words, we can apply word embedding to learn latent representations of items (e.g., item embeddings) and users (e.g. user embeddings). Based on this, we propose a *Regularized Multi-Embedding* based recommendation model (RME), which jointly decomposes (1) a user-item interaction matrix, (2) a user co-occurrence matrix, (3) a co-liked item co-occurrence matrix, and (4) a co-disliked item co-occurrence matrix. The RME model concurrently exploits the co-liked co-occurrence patterns and co-disliked co-occurrence patterns of items to enrich the items' latent factors. It also augments users' latent factors by incorporating user co-occurrence patterns on their preferred items. Figure 1 illustrates an overview of our RME model.

Both liked and disliked items can be explicitly measured by rating scores (e.g., a liked item is $\geq$ 4 star-rating and a disliked item is $\leq$ 2 star-rating) in explicit feedback datasets such as 5-star rating datasets (e.g., a Movie dataset and an Amazon dataset). However, in implicit feedback datasets (e.g., a music listening dataset and a browsing history dataset), users do not explicitly express their preferences. In *implicit feedback datasets*, the song plays and URL clicks could indicate how much users like the items (i.e., positive samples), but inferring the disliked items (i.e., negative samples) is a big challenge due to the nature of implicit feedback. In order to deal with this challenge, we propose an algorithm which infers a user's disliked items in implicit feedback datasets, so that we can build an RME model and recommend items for both explicit and implicit feedback datasets. In this paper, we made the following contributions:

- We proposed a joint RME model, which combined weighted matrix factorization, co-liked item embedding, co-disliked item embedding, and user embedding, for both explicit and implicit feedback datasets.
- We designed a user-oriented EM-like algorithm to draw negative samples (i.e., disliked items) from implicit feedback datasets.
- We conducted comprehensive experiments and showed that the RME model substantially outperformed several baseline models in both explicit and implicit feedback datasets.

2 PRELIMINARIES

Item. Items are objects that users interact with or consume. They can be interpreted in various ways, depending on the context of a dataset. For example, an item is a movie in a movie dataset such as MovieLens, whereas it is a song in TasteProfile.

Liked items and disliked items. In explicit feedback datasets such as MovieLens (a 5-star rating dataset), an item $\geq$ 4 stars is classified to a liked item of the user, and an item $\leq$ 2 stars is classified to a disliked item of the user [5]. In implicit feedback datasets such as TasteProfile, the more a user consumes an item, the more he/she likes it (e.g., larger play count in TasteProfile indicates stronger preference). But, disliked items are not explicitly observable.

Top-N recommendation. In this paper, we focus on top-N recommendation scenario, in which a recommendation model suggests a list of top-N most appealing items to users. We represent the interactions between users and items by a matrix M^{m*n} where m is the number of users and n is the number of items. If a user u likes an item p, M_{up} will be set to 1. From M, we are interested in extracting co-occurrence patterns including liked item co-occurrences, disliked item co-occurrences, and user co-occurrences. Our goal is to exploit those co-occurrence information to learn the latent representations of users and items, then recommend top-N items to the users.

Notations. Table 1 shows key notations used in this paper. Note that all vectors in the paper are column vectors.

Table 1: Notations.

Notation	Description
M	a $m \times n$ user-item interaction matrix.
U	a $m \times k$ latent factor matrix of users.
P	a $n \times k$ latent factor matrix of items.
X	a $n \times n$ SPPMI matrix of liked items-item co-occurrences.
Y	a $n \times n$ SPPMI matrix of disliked item-item co-occurrences.
Z	a $m \times m$ SPPMI matrix of user-user co-occurrences.
α_u	a $k \times 1$ latent factor vector of user u.
β_p	a $k \times 1$ latent factor vector of item p.
γ_i	a $k \times 1$ latent factor vector of co-liked item context i.
$\delta_{i\prime}$	a $k \times 1$ latent factor vector of co-disliked item context $i\prime$.
θ_j	a $k \times 1$ latent factor vector of user context j.
λ	a hyperparameter of regularization terms.
b, d	co-liked and co-disliked item bias.
c, e	co-liked and co-disliked item context bias.
f, g	user bias and user context bias.
w_{up}	a weight for an interaction between user u and her liked item p.
$w_{uj}^{(u)}$	a weight for two users u and j who co-liked same items.
$w_{pi}^{(+p)}$	a weight for two items p and i that are co-liked by users.
$w_{pi}^{(-p)}$	a weight for two items p and i that are co-disliked by users.

3 OUR RME MODEL

First, we review the Weighted Matrix Factorization (WMF), and co-liked item embedding. Then, we propose co-disliked item embedding and user embedding. Finally, we describe our RME model and present how to compute it.

3.1 WMF, Embedding and RME model

Weighted matrix factorization (WMF). WMF is a widely-used collaborative filtering method in recommender systems [14]. Given a sparse user-item matrix $M^{m \times n}$, the basic idea of WMF is to decompose M into a product of 2 low rank matrices $U^{m \times k}$ and $P^{n \times k}$ (i.e., $M = U \times P^T$), where k is the number of dimensions and $k < min(m, n)$. Here, U is interpreted as a latent factor matrix of users, and P is interpreted as a latent factor matrix of items.

We denote $U^T = (\alpha_1, \alpha_2, ..., \alpha_m)$ where $\alpha_u \in R^k$ ($u \in \overline{1, m}$) and α_u represents the latent factor vector of user u. Similarly, we denote $P^T = (\beta_1, \beta_2, ..., \beta_n)$ where $\beta_p \in R^k$ ($p \in \overline{1, n}$) and β_p represents the latent factor vector of item p. The objective of WMF is defined by:

$$\mathcal{L}_{WMF} = \frac{1}{2}\sum_{u,p} w_{up}(M_{up} - \alpha_u^T \beta_p)^2 + \frac{1}{2}\Big(\lambda_\alpha \sum_u ||\alpha_u||^2 + \lambda_\beta \sum_p ||\beta_p||^2\Big) \quad (1)$$

where w_{up} is a hyperparameter to compensate the interaction between user u and item p, and is used to balance between the number of non-zero and zero values in a sparse user-item matrix. The weight w of the interaction between user u and item p (denoted as w_{up}) can be set as $w_{up} = l(1 + \phi M_{up})$ [14, 20] where l is a relative scale and ϕ is a constant. λ_α and λ_β are used to adjust the importance of two quadratic regularization terms $\sum_u ||\alpha_u||^2$ and $\sum_p ||\beta_p||^2$.

Word embedding models. Word embedding models have recently received a lot of attention from the research community. Given a sequence of training words, the embedding models learn a latent representation for each word. For example, *word2vec* [24] is one of popular word embedding methods. Especially, the skip-gram model in word2vec tries to predict surrounding words (i.e., word context) of a given word in the training set.

According to Levy et al. [19], skip-gram model with negative sampling (SGNS) is equivalent to implicitly factorize a word-context matrix, whose cells are the *Pointwise Mutual Information* (PMI) of the respective word and context pairs, shifted by a global constant. Let D as a collection of observed word and context pairs, the PMI between a word i and its word context j is calculated as:

$$PMI(i,j) = log\frac{P(i,j)}{P(i) * P(j)}$$

where $P(i,j)$ is the joint probability that word i and word j appears together within a window size (e.g. $P(i,j) = \frac{\#(i,j)}{|D|}$, where $|D|$ refers to the total number of word and word context pairs in D). Similarly, $P(i)$ is the probability the word i appears in D, and $P(j)$ is the probability word j appears in D (e.g. $P(i) = \frac{\#(i)}{|D|}$ and $P(j) = \frac{\#(j)}{|D|}$). Obviously, $PMI(i,j)$ can be calculated as:

$$PMI(i,j) = log\frac{\#(i,j) * |D|}{\#(i) * \#(j)} \tag{2}$$

By calculating PMI of all word-context pairs in D, we can form a squared $n \times n$ matrix M^{PMI} where n is the total number of distinct words in D. Next, a *Shifted Positive Pointwise Mutual Information* (SPPMI) of two words i and j is calculated as:

$$SPPMI(i,j) = max(PMI(i,j) - log(s), 0) \tag{3}$$

where s is a hyperparameter to control the density of PMI matrix M^{PMI} and s can be interpreted equivalently as a hyperparameter that indicates the number of negative samples in SGNS. When s is large, more values in the matrix M^{PMI} are cleared, leading M^{PMI} to become sparser. When s is small, matrix M^{PMI} becomes denser. Finally, factorizing matrix M^{SPPMI}, where each cell in M^{SPPMI} is transformed by Formula (3), is equivalent to performing SGSN.

Co-liked item embedding (LIE). As mentioned in the previous studies [3, 10, 20], when users liked/consumed items in a sequence, the items sorted by the ascending interaction time order can be inferred as a sequence. Thus, performing co-liked item embeddings to learn latent representations of items is equivalent to perform word embeddings to learn latent representations of words. Therefore, we can apply word embedding methods to learn latent representations of items, and perform a joint learning between embedding models and traditional factorization methods (e.g. WMF).

Given each user's liked item list, we generate co-liked item-item co-occurrence pairs without considering liked time. Particularly, given a certain item in the item sequence, we consider all other items as its contexts. We call this method as a *greedy context generation* method which can be applied to other non-timestamped datasets. After generating item and item context pairs, we construct an item co-occurrence SPPMI matrix and perform SPPMI matrix factorization. In particular, given generated item-item co-occurrence pairs, we construct a SPPMI matrix of items by applying Equation (2) to calculate the pointwise mutual information of each pair, and then by measuring the shifted positive pointwise mutual information of the pair based on Equation (3). Once the SPPMI matrix of co-liked items is constructed, we incorporate it to the traditional matrix factorization method to improve the item latent representations.

Co-disliked item embedding (DIE). As mentioned in the Introduction section, when many users disliked two items p_1 and p_2 together, the two items can form a pair of co-occurred disliked items. If the recommender systems learned this disliked co-occurrence pattern, it would not recommend item p_2 to a user, who disliked p_1. This will help reduce the false positive rate for the recommender systems. Therefore, similar to liked item embeddings, we applied the word embedding technique to exploit the disliked co-occurrence information to enhance the item's latent factors.

User embedding (UE). When two users A and B preferred same items, we can assume the two users share similar interests. Therefore, if user A enjoyed an item p that has not been observed in user B's transactions, we can recommend the item to user B. Similar to liked and disliked item embeddings, we applied the word embedding technique to learn user embeddings that explain the co-occurrence patterns among users.

From the user-item interaction matrix $M^{m \times n}$, where each row represents consumed items of a user (e.g. a list of items that the user rated or backed), we only keep liked items per user in the matrix M'. Then, we construct a $n \times m$ reverse matrix M'^T of M', where each row represents users that liked a certain item. Then, users, who liked the same item, form a sequence, and the sequence of users is interpreted as a sequence of words. From this point, word embedding techniques are applied to the user sequence to enhance latent representations of users.

Our RME model. It is a joint learning model combining WMF, co-liked item embedding, co-disliked item embedding, and user embedding. It minimizes the following objective function:

$$\begin{aligned}
\mathcal{L} = & \overbrace{\frac{1}{2}\sum_{u,p} w_{up}(M_{up} - \alpha_u^T\beta_p)^2}^{\mathcal{L}_1} \text{ (WMF)} \\
& + \overbrace{\frac{1}{2}\sum_{X_{pi}\neq 0} w_{pi}^{(+p)}(X_{pi} - \beta_p^T\gamma_i - b_p - c_i)^2}^{\mathcal{L}_2} \text{ (LIE)} \\
& + \overbrace{\frac{1}{2}\sum_{Y_{pi'}\neq 0} w_{pi'}^{(-p)}(Y_{pi'} - \beta_p^T\delta_{i'} - d_p - e_{i'})^2}^{\mathcal{L}_3} \text{ (DIE)} \\
& + \overbrace{\frac{1}{2}\sum_{Z_{uj}\neq 0} w_{uj}^{(u)}(Z_{uj} - \alpha_u^T\theta_j - f_u - g_j)^2}^{\mathcal{L}_4} \text{ (UE)} \\
& + \frac{1}{2}\lambda\Big(\sum_u ||\alpha_u||^2 + \sum_p ||\beta_p||^2 + \sum_i ||\gamma_i||^2 + \sum_{i'} ||\delta_{i'}||^2 + \sum_j ||\theta_j||^2\Big)
\end{aligned} \tag{4}$$

where the item's latent representation β_p is shared among WMF, co-liked item embedding and co-disliked item embedding. The user's latent representation α_u is shared between WMF and user embedding. X and Y are SPPMI matrices, constructed by co-liked item-item co-occurrence patterns and disliked item-item co-occurrence patterns, respectively. γ and δ are $k \times 1$ latent representation vectors of co-liked item context and co-disliked item context, respectively. Z is a SPPMI matrix constructed by user-user co-occurrence patterns. θ is a $k \times 1$ latent representation vector of a user context. $w^{(+p)}$, $w^{(-p)}$ and $w^{(u)}$ are hyperparameters to compensate for item/user co-occurrences in X, Y and Z when performing decomposition. b is liked item bias, and c is co-liked item context bias. d is disliked item bias, and e is co-disliked item-context bias. f and g are user bias and user context bias, respectively. Incorporating bias terms were originally introduced in [16]. A liked item bias b_p and a co-liked item context bias c_i mean that when the two items p_i and p_j are co-liked by users, each item may have a little bit higher/lower preference compared to the average preference. The similar explanation is applied to the other biases. The last line shows regularization terms along with a hyperparameter λ to control their effects.

3.2 Optimization

We can use the stochastic gradient descent to optimize the Equation (4). However, it is not stable and sensitive to parameters [37]. Therefore, we adopt vector-wise ALS algorithm [37, 39] that alternatively optimize each model's parameter in parallel while fixing the other parameters until the model gets converged. Specifically, we calculate the partial derivatives of the model's objective function with regard to the model parameters (i.e., $\{\alpha_{1:m}, \beta_{1:n}, \gamma_{1:n}, \delta_{1:n}, b_{1:n}, c_{1:n}, d_{1:n}, e_{1:n}, \theta_{1:m}, f_{1:m}, g_{1:m}\}$). Then we set them to zero and obtain updating rules. Details are given as follows:

From the objective function in Equation (4), while taking partial derivatives of $\mathcal{L}$ with regard to each user's latent representation vector α_u, we observe that only $\mathcal{L}_1$, $\mathcal{L}_4$ and the L2 user regularization $\frac{1}{2}\lambda \sum_u ||\alpha_u||^2$ contain α_u. Therefore, we obtain:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \alpha_u} &= \frac{\partial \mathcal{L}_1}{\partial \alpha_u} + \frac{\partial \mathcal{L}_4}{\partial \alpha_u} + \frac{\partial \lambda \sum_u ||\alpha_u||^2}{2\partial \alpha_u} \\ &= -\sum_{u,p} w_{up}(M_{up} - \alpha_u^T \beta_p)\beta_p^T - \sum_{u,j} w_{uj}^{(u)}(Z_{uj} - \alpha_u^T\theta_j - f_u - g_j)\theta_j^T + \lambda \alpha_u^T\end{aligned}$$

Fixing item latent vectors β, user context latent vectors θ, user bias d and user context bias e, and solving $\frac{\partial \mathcal{L}}{\partial \alpha_u} = 0$, we obtain the updating rule of α_u as follows:

$$\begin{aligned}\alpha_u =& \Big[\sum_p w_{up}\beta_p\beta_p^T + \sum_{j|Z_{uj}\neq 0} w_{uj}^{(u)}\theta_j\theta_j^T + \lambda I_K\Big]^{-1} \\ & \Big[\sum_p w_{up}M_{up}\beta_p + \sum_{j|Z_{uj}\neq 0} w_{uj}^{(u)}(Z_{uj} - f_u - g_j)\theta_j\Big] \\ & ,\forall 1 \leq u \leq m, 1 \leq p \leq n, 1 \leq j \leq m\end{aligned} \tag{5}$$

Similarly, taking partial derivatives of $\mathcal{L}$ with respect to each item latent vector β_p needs to consider only $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ and item regularization $\frac{1}{2}\lambda \sum_p ||\beta_p||^2$. By fixing other parameters and solving $\frac{\partial \mathcal{L}}{\partial \beta_p} = 0$, we obtain:

$$\begin{aligned}\beta_p =& \Big[\sum_u w_{up}\alpha_u\alpha_u^T + \sum_{i|X_{pi}\neq 0} w_{pi}^{(+p)}\gamma_i\gamma_i^T + \sum_{i'|Y_{pi'}\neq 0} w_{pi'}^{(-p)}\delta_{i'}\delta_{i'}^T + \lambda I_K\Big]^{-1} \\ & \Big[\sum_u w_{up}M_{up}\alpha_u + \sum_{i|X_{pi}\neq 0} w_{pi}^{(+p)}(X_{pi} - b_p - c_i)\gamma_i + \\ & \sum_{i'|Y_{pi'}\neq 0} w_{pi'}^{(-p)}(Y_{pi'} - d_p - e_{i'})\delta_{i'}\Big] \\ & ,\forall 1 \leq u \leq m, 1 \leq p \leq n, 1 \leq i, i' \leq n\end{aligned} \tag{6}$$

In the same manner, we obtain the update rules of item contexts γ, δ, and user context θ alternatively as follows:

$$\begin{aligned}\gamma_i =& \Big[\sum_{p|X_{ip}\neq 0} w_{ip}^{(+p)}\beta_p\beta_p^T + \lambda I_K\Big]^{-1}\Big[\sum_{p|X_{ip}\neq 0} w_{ip}^{(+p)}(X_{ip} - b_p - c_i)\beta_p\Big] \\ \delta_{i'} =& \Big[\sum_{p|Y_{i'p}\neq 0} w_{i'p}^{(-p)}\beta_p\beta_p^T + \lambda I_K\Big]^{-1}\Big[\sum_{p|Y_{i'p}\neq 0} w_{i'p}^{(-p)}(Y_{i'p} - d_p - e_{i'})\beta_p\Big] \\ \theta_j =& \Big[\sum_{u|Z_{ju}\neq 0} w_{ju}^{(u)}\alpha_u\alpha_u^T + \lambda I_K\Big]^{-1}\Big[\sum_{u|Z_{ju}\neq 0} w_{ju}^{(u)}(Z_{ju} - d_u - e_j)\alpha_u\Big] \\ & ,\forall 1 \leq u \leq m, 1 \leq p \leq n, 1 \leq i, i' \leq n\end{aligned} \tag{7}$$

The item biases and item context biases b, c, d, e, as well as the user and user context biases f, g are updated alternatively using the following update rules:

$$\begin{aligned}b_p =& \frac{1}{|i : X_{pi} \neq 0|}\sum_{i:X_{pi}\neq 0}(X_{pi} - \beta_p^T\gamma_i - c_i) \\ c_i =& \frac{1}{|p : X_{ip} \neq 0|}\sum_{p:X_{ip}\neq 0}(X_{ip} - \beta_p^T\gamma_i - b_p) \\ d_p =& \frac{1}{|i' : Y_{pi'} \neq 0|}\sum_{i':Y_{pi'}\neq 0}(Y_{pi'} - \beta_p^T\delta_{i'} - e_{i'}) \\ e_{i'} =& \frac{1}{|p : Y_{i'p} \neq 0|}\sum_{p:Y_{i'p}\neq 0}(Y_{i'p} - \beta_p^T\delta_{i'} - d_p) \\ f_u =& \frac{1}{|j : Z_{uj} \neq 0|}\sum_{j:Z_{uj}\neq 0}(Z_{uj} - \alpha_u^T\theta_j - g_j) \\ g_j =& \frac{1}{|u : Z_{ju} \neq 0|}\sum_{u:Z_{ju}\neq 0}(Z_{ju} - \alpha_u^T\theta_j - f_u)\end{aligned} \tag{8}$$

In short, the pseudocode of our proposed RME model is presented in Algorithm 1.

Algorithm 1 RME algorithm

```
Require: M, $\lambda$
 1: Build SPPMI matrices of liked item $X$, disliked item $Y$ and user co-occurrences $Z$ using Eq. (2) and Eq. (3)
 2: Initialize $U$ (or $\alpha_{1:m}$), $P$ (or $\beta_{1:n}$), $\gamma_{1:n}$, $\delta_{1:n}$, $\theta_{1:m}$.
 3: Initialize $b_{1:n}$, $c_{1:n}$, $d_{1:n}$, $e_{1:n}$, $f_{1:m}$, $g_{1:m}$.
 4: repeat
 5:     For each user u, update $\alpha_u$ by Eq. (5) ($1 \leq u \leq m$).
 6:     For each item p, update $\beta_p$ by Eq. (6) ($1 \leq p \leq n$).
 7:     Alternatively update each item context $\gamma_i$, $\delta_{i'}$ and user context $\theta_j$ by Eq. (7) ($1 \leq i, i' \leq n; 1 \leq j \leq m$).
 8:     Alternatively update each bias $b_p$, $c_i$, $d_p$, $e_{i'}$, $f_u$, $g_j$ by Eq. (8) ($1 \leq p, i, i' \leq n; 1 \leq u, j \leq m$).
 9: until convergence
10: return $U$, $P$
```

3.3 Complexity Analysis

In this section, we briefly provide time complexity analysis of our model. Let $\Omega_M = \{(u, p) \mid M_{up} \neq 0\}$, $\Omega_X = \{(p, i) \mid X_{pi} \neq 0\}$, $\Omega_Y = \{(p, i') \mid Y_{pi'} \neq 0\}$, $\Omega_Z = \{(u, j) \mid Z_{uj} \neq 0\}$. Constructing SPPMI matrices X, Y and Z take $O(|\Omega_X|^2)$, $O(|\Omega_Y|^2)$ and $O(|\Omega_Z|^2)$, respectively. However, the SPPMI matrices are calculated once and are constructed in parallel using batch processing, so they are not costly. For learning RME model, computing α takes $O((|\Omega_M| + |\Omega_Z|)k^2 + k^3)$ time, and computing β takes $O((|\Omega_M| + |\Omega_X| + |\Omega_Y|)k^2 + k^3)$ time. Also, it takes $O(|\Omega_X|k^2 + k^3)$ for computing co-liked item context γ, and so do other latent contexts δ, θ. It takes $O(|\Omega_Z|k)$ time to compute all user bias f and so do the other biases. Thus, the time complexity for RME is $O(\eta(2(|\Omega_M| + |\Omega_X| + |\Omega_Y| + |\Omega_Z|)k^2 + (2m+3n)k^3))$, where η is the number of iterations. Since $k << min(m, n)$ and M, X, Y, Z are often sparse, which mean $(|\Omega_M| + |\Omega_X| + |\Omega_Y| + |\Omega_Z|)$ is small, the time complexity of RME is shortened as $O(\eta(m + \frac{3}{2}n)k^3)$, which scales linearly to the conventional ALS algorithm for collaborative filtering [37].

4 INFERRING DISLIKED ITEMS IN IMPLICIT FEEDBACK DATASETS

Unlike explicit feedback datasets, there is a lack of substantial evidence, on which items the users disliked in implicit feedback datasets. Since our model exploits co-disliked item co-occurrences patterns among items, the implicit feedback datasets challenge our model. To deal with it, we can simply assume that missing values are equally likely to be negative feedback, then sample some negative instances from missing values with uniform weights [12, 27, 32, 34]. However, assigning uniform weight is suboptimal because the missing values are a mixture of negative and unknown feedbacks. A recent work suggests to sample negative instances by assigning non-uniform weights based on item popularity [13]. The idea is that popular items are highly aware by users, so if they are not observed in a user's transactions, it assumes that the user dislikes them. However, this sampling method is also not optimal because same unobserved popular items can be sampled across multiple users. This approach does not reflect user's personalized interests.

Instead, we follow the previous works [22, 25, 38], and propose a user-oriented EM-like algorithm to draw negative samples (i.e., inferred disliked items) for users in implicit feedback datasets. Our approach is described as follows:

First, we assume that an item with a low ranking score of being liked will have a higher probability to be drawn as a negative sample of a user. Given r_u is the ranked list of all items of the user u, the prior probabilities of items to be drawn as negative samples are calculated by using a softmax function as follows:

$$Pr_i^{(u)} = \frac{\exp(-r_u[i])}{\sum_{j=1}^{n} \exp(-r_u[j])} \tag{9}$$

After negative samples are drawn for each user, we built the RME model by using Algorithm 1. The pseudocode of the RME model for implicit feedback datasets is presented in Algorithm 2.

In Algorithm 2, since each user may prefer a different number of items, we define a hyper-parameter τ as a negative sample drawing ratio to control how many negative samples we will sample for each user. In line 6, $count(u)$ returns the number of observed items of a user u. Then, the number of drawn negative samples for the user u is calculated and assigned to ns. If a user prefers 10 items and $\tau = 0.8$, the algorithm will sample 8 disliked items. We note that sampling with replacement is used such that different items are drawn independently. The value of τ is selected using the validation data. In line 8, we set the ranking of observed items to $+\infty$ to avoid drawing the observed items as negative samples. In line 12, we build the RME model based on the negative samples drawn in the Expectation step, and temporally store newly learned user latent matrix, item latent matrix and corresponding NDCG to $U\_tmp, P\_tmp, ndcg$ variables, respectively (NDCG is a measure to evaluate recommender systems, which will be mentioned in Experiment section). If we obtain a better $ndcg$ comparing with the previous NDCG $prev\_ndcg$ (line 13), we will update $U, P, prev\_ndcg$ with new values (line 14). Overall, at the end of the Expectation step, we obtain the disliked items

Algorithm 2 RME model for implicit feedback datasets using user-oriented EM-like algorithm to draw negative samples

```
Require: M, negative sample drawing ratio $\tau$
 1: $max\_iter = 10, prev\_ndcg = 0, iter = 0$
 2: Initialize Step: $U, P = WMF(M)$
 3: repeat
 4:     $iter$ += 1
 5:                                              ▷ Expectation Step
 6:     for $u \in [1, m]$ do:
 7:         $ns = \tau * count$(u)
 8:         Compute ranked item list: $r_u = P.\alpha_u$
 9:         Assign observed items with ranking of $+\infty$.
10:         Measure prior probabilities of items to be drawn as
            negative samples by Eq. (9) then randomly draw $ns$ negative
            samples with those prior probabilities.
11:     end for
12:                          ▷ Maximization Step with early stopping
13:     $U\_tmp, P\_tmp, ndcg$ = RME(train_data, vad_data)
14:     if $ndcg > prev\_ndcg$ then
15:         $U, P, prev\_ndcg = U\_tmp, P\_tmp, ndcg$
16:     else
17:         break                                ▷ Early stopping
18:     end if
19: until $iter < max\_iter$
20: return $U, P$
```

for each user. Then, in the Maximization step, we build our RME model to re-learn user and item latent representations U and P. The process is repeated until getting converged or the early stopping condition (line 13 to 17) is satisfied.

Time Complexity: In order to construct RME model for implicit feedback datasets, we need to re-learn RME model, which includes re-building 3 SPPMI matrices in the maximization step in $\eta\prime$ iterations to get converged. Thus, it takes $O(\eta\prime((|\Omega_X|^2 + |\Omega_Y|^2 + |\Omega_Z|^2) + \eta(m + \frac{3}{2}n)k^3))$ time where $\eta\prime$ is small.

5 EXPERIMENTS

5.1 Experimental Settings

Datasets: To measure the performance of our RME model, we evaluate the model on 3 real-world datasets:

- MovieLens-10M [29]: is an explicit feedback dataset. It consists of 69,878 users and 10,677 movies with 10m ratings. Following the k-cores preprocessing [11, 12], we only kept users, who rated at least 5 movies, and movies, which were rated by at least 5 users. This led to 58,057 users and 7,223 items (density= 0.978%).
- MovieLens-20M: is an explicit feedback dataset. It consists of 138,000 users, 27,000 movies, and 20 millions of ratings. We filtered with the same condition as for MovieLens-10M. This led to 111,146 users and 9,888 items (density= 0.745%).
- TasteProfile: is an implicit feedback dataset containing a song's play count by a user [1]. The play counts are user's implicit preference and are binarized. Similar to [20], we first

[1]http://the.echonest.com/

Table 2: **Performance of the baselines, our RME model, and its two variants. The improvement of our model over the baselines and its variants were significant with *p-value* < 0.05 in the three datasets under the non-directional two-sample t-test.**

Method	**MovieLens-10M**			**MovieLens-20M**			**TasteProfile**		
	Recall@5	NDCG@20	MAP@10	Recall@5	NDCG@20	MAP@10	Recall@5	NDCG@20	MAP@10
Item-KNN	0.0137	0.0338	0.0397	0.0131	0.0345	0.0402	0.0793	0.0685	0.0904
Item2vec	0.1020	0.1001	0.0502	0.1066	0.1019	0.0539	0.1455	0.1593	0.0727
WMF	0.1280	0.1245	0.0655	0.1348	0.1290	0.0720	0.1745	0.1853	0.0931
Cofactor	0.1460	0.1381	0.0772	0.1480	0.1387	0.0804	0.1771	0.1873	0.0950
U_RME	0.1516	0.1412	0.0818	0.1524	0.1425	0.0847	0.1825	0.1899	0.0997
I_RME	0.1511	0.1422	0.0817	0.1530	0.1412	0.0838	0.1826	0.1915	0.0996
RME	**0.1562**	**0.1458**	**0.0841**	**0.1570**	**0.1461**	**0.0869**	**0.1876**	**0.1954**	**0.1025**

subsampled the dataset to 250k users and 25k items. Then we kept only users, who listened to at least 20 songs, and songs, which were listened by at least 50 users. As a result, 221,011 users and 22,713 songs were remained (density= 0.291%).

Baselines: To illustrate the effectiveness of our RME model, we compare it with the following baselines:

- WMF [14]: It is a weighted matrix factorization with *l2*-norm regularization.
- Item-KNN [7]: This is an item neighborhood-based collaborative filtering method.
- Item2Vec [3]: This method used Skip-gram with negative sampling [24] to learn item embeddings, then adopted a similarity score between item embeddings to generate user's recommendation lists.
- Cofactor [20]: This is a method that combines WMF and co-liked item embedding.

We note that we do not compare our models with user collaborative filtering method (i.e. User-KNN) because it is not applicable to run the method on the large datasets. However, [31] reported that User-KNN had worse performance than Item-KNN, especially when there are many items but few ratings in a dataset.

Our models: We not only compare the baselines with our RME, but also two variants of our model such as U_RME and I_RME to show the effectiveness of incorporating all of the user embeddings, liked-item embeddings and disliked-item embeddings:

- U_RME (i.e., RME - DIE): This is a variant of our model, considering only WMF, user embeddings, and liked-item embeddings.
- I_RME (i.e., RME - UE): This is another variant of our model, considering only WMF, liked-item embeddings, and disliked-item embeddings.
- RME: This is our proposed RME model.

Evaluation metrics. We used three well-known ranking-based metrics – Recall@N, normalized discounted cumulative gain (NDCG@N), and mean average precision (MAP@N). Recall@N considers all items in top N items equally, whereas NDCG@N and MAP@N apply an increasing discount of log_2 to items at lower ranks.

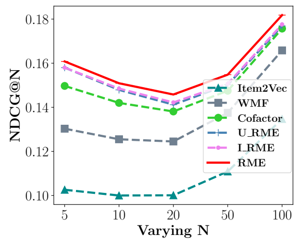
Training, validation and test sets. We follow 70/10/20 proportions for splitting the original dataset into training/validation/test sets [21]. MovieLens-10M and MovieLens-20M datasets contain timestamp values of user-movie interactions. To create training, validation and testing sets for these datasets, we sorted all user-item interaction pairs in the ascending interaction time order in each of MovieLens-10M and MovieLens-20M datasets. The first 80% was used for training and validation, and the rest 20% data was used as a test set. Out of 80% data extracted for training and validation, we randomly took 10% for the validation set. To measure the statistical significance of RME over the baselines, we repeated the splitting process five times (i.e., generating five pairs of training and validation sets). Since TasteProfile dataset did not contain timestamp information of user-song interactions, we randomly split the TasteProfile dataset into training/validation/test sets five times with 70/10/20 proportions. Averaged results are reported in the following subsection.

Stopping criteria and Hyperparameters. To decide when to stop training a model, we measured the model's $NDCG@100$ by using the validation set. We stopped training the model when there was no further improvement. Then, we applied the best model to the test set to evaluate its performance. This method was applied to the baselines and RME.

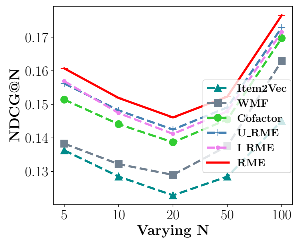
All hyper-parameters were tuned on the validation set by a grid search. We used the same hyper-parameter setting in all models. The grid search of the regularization weight λ was performed in {0.001, 0.005, 0.01, 0.05, ..., 10}. The size of latent dimensions was in a range of {30, 40, 50, ..., 100}. We set weights $w^{(+p)} = w^{(-p)} = w^{(u)} = w$ for all user-user and item-item co-occurrence pairs. When building our RME model for TasteProfile dataset, we do a grid search for the negative sample drawing ratio τ in {0.2, 0.4, 0.6, 0.8, 1.0}.

5.2 Experimental Results

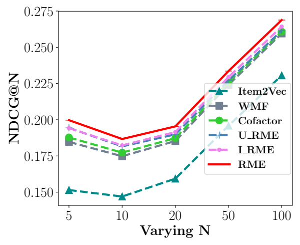
RQ1: Performance of the baselines and RME. Table 2 presents recommendation results of RME and compared models at Recall@5, NDCG@20, and MAP@10. First, we compared RME with the baselines. We observed that RME outperformed all baselines in the three datasets, improving the Recall by 6.3%, NDCG by 5.1%, and MAP by 8.3% on average over the best baseline (*p-value* < 0.001). Second, we compared two variants of RME model with the baselines. We see that both U_RME and I_RME performed better than the baselines. Adding user embeddings improved the Recall by 3.0~3.5%, NDCG by 1.4~2.2%, and MAP by 4.2~5.8% (*p-value* < 0.001), while adding disliked item embeddings improved the Recall by 3.1~3.8%, NDCG by 2.2~3.0%, and MAP by 4.9~6.0%. Third, we compare RME with its two variants. RME also achieved the best result, improving Recall by 2.6~3.0%, NDCG by 2.0~2.5%, and MAP by 2.6~2.8% (*p-value* $<$ 0.05). We further evaluated NDCG@N of our model when varying



(a) NDCG@N on MovieLens-10M.

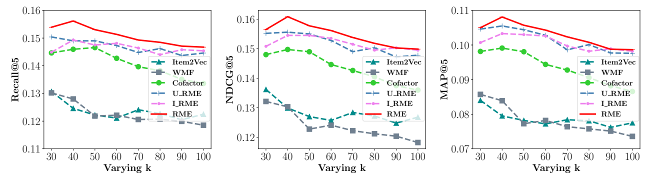


(b) NDCG@N on MovieLens-20M.

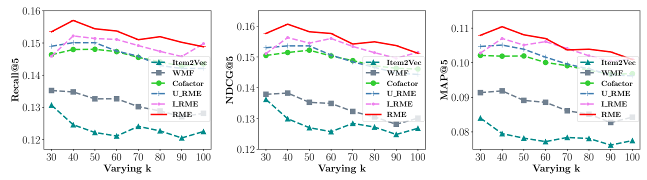


(c) NDCG@N on TasteProfile.

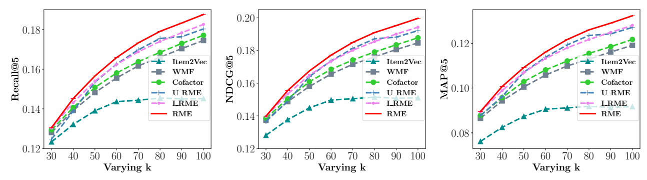
Figure 2: Performance of all models when varying top N.



(a) Recall@5, NDCG@5, MAP@5 on MovieLens-10M. Fix $\lambda = 1$, and vary k.

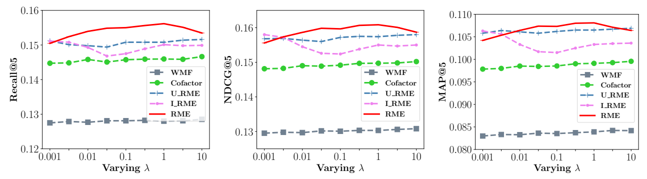


(b) Recall@5, NDCG@5, MAP@5 on MovieLens-20M. Fix $\lambda = 0.5$, and vary k.

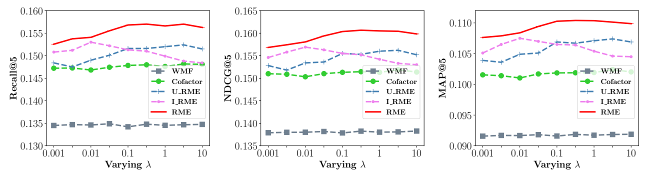


(c) Recall@5, NDCG@5, MAP@5 on TasteProfile. Fix $\lambda = 10$, $\tau = 0.2$, and vary k.

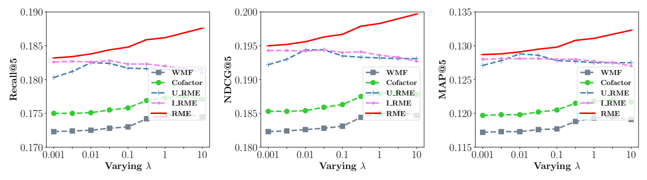
Figure 3: Performance of models when varying the latent dimension size k with fixing the value of λ.



(a) Recall@5, NDCG@5, MAP@5 on MovieLens-10M. Fix $k = 40$, and vary λ.



(b) Recall@5, NDCG@5, MAP@5 on MovieLens-20M. Fix $k = 40$, and vary λ.



(c) Recall@5, NDCG@5, MAP@5 on TasteProfile. Fix $k = 100$, $\tau = 0.2$, and vary λ.

Figure 4: Performance of models when varying λ with fixing the latent dimension size k. Item2Vec did not contain regularization, so we excluded it.

top N in range {5, 10, 20, 50, 100}. Figure 2 shows our result (we excluded Item-KNN in the figure and following figures since it performed extremely worst). Our model still performed the best. On average, it improved NDCG@N by 6.2% comparing to the baselines, and by 3.3% comparing to its variants. These experimental results show that both co-disliked item embedding and user embedding positively contributed to RME, and also confirm our observations addressed in Section 1 are correct.

The experimental results in TasteProfile in Table 2 showed that inferring disliked items in Algorithm 2 worked well since RME model incorporating co-disliked item embedding outperformed the baselines. To further confirm the effectiveness of the algorithm, we also applied it to MovieLens-10M and MovieLens-20M datasets after removing the explicit disliking information, pretending them as implicit feedback datasets. In the datasets without disliking information, RME under Algorithm 2 still outperformed the best baseline with 4.2%, 4.6% and 7.2% improvements on average in Recall, NDCG and MAP, respectively (*p-value* < 0.001). Its performance was slightly lower than the original RME (based on explicit disliking information) at 0.3%, 0.7% and 1.3% on average in Recall, NDCG, and MAP, respectively. The experimental results confirmed the effectiveness of Algorithm 2. We note that Algorithm 2 got converged in up to 4 iterations for all three datasets by the early stopping condition. Due to the space limitation, we do not include figures which show the loss over iterations.

RQ2:Parameter sensitivity analysis: We analyze the effects of the parameters in RME model in order to answer the following research questions: (*RQ2-1*:) How does RME work when varying the latent dimension size k?; (*RQ2-2*:) How does RME model change with varying λ?; (*RQ2-3*:) How sensitive is the RME model on an implicit feedback dataset (e.g. TasteProfile) when varying negative sample drawing ratio τ?; and (*RQ2-4*:) Can RME achieve better performance with a dynamic setting of regularization hyper-parameters?

Regarding *RQ2-1*, Figure 3 shows the sensitivity of all compared models when fixing λ and varying the latent dimension size k in {30, 40, 50, 60, 70, 80, 90, 100}. It is clearly observed that our

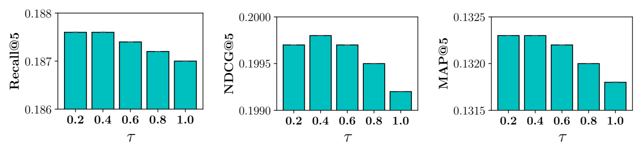


Figure 5: Performance of RME in TasteProfile when varying negative sample drawing ratio τ with fixing k = 100, λ = 10.

model outperforms the baselines in all datasets. In MovieLens-10M and MovieLens-20M datasets, all six models downgrade the performance when the latent dimension size k is over 60. In the TasteProfile dataset, when increasing k, although all models gain a higher performance, our model tends to achieve much higher performance.

In a *RQ2-2* experiment, we exclude Item2Vec because this model does not contain the regularization term. We fix $k = 40$ in MovieLens-10M and MovieLens-20M. In TasteProfile dataset, we fix k=100, τ=0.2. We vary lambda in range {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10}. Then, we report the average results of Recall@5, NDCG@5, and MAP@5. As shown in Figure 4, the performance of our model is better than the baselines. In MovieLens-10M and MovieLens-20M dataset, RME increases its performance when increasing λ up to 1, then its performance goes down when λ is increasing more. In TasteProfile, RME tends to gain a higher performance and more outperformed the baselines when λ is increasing.

To understand the sensitivity of our model when varying negative sample drawing ratio τ in the implicit feedback dataset – TasteProfile (RQ2-3), we vary τ in {0.2, 0.4, 0.6, 0.8, 1.0}, and fix $k = 100$ and $\lambda = 10$. Figure 5 shows that when τ increases, our model degrades with a small amount (e.g. around -0.3% in Recall@5 and NDCG@5, and -0.4% in MAP@5). In NDCG@5, our model gains the best result when $\tau = 0.4$. We note that our worst case (when $\tau = 1.0$) is still better than the best baseline presented in Table 2. This shows that the sensitivity of our model with regard to the negative sample drawing ratio τ is small/limited.

In our previous experiments, we used a static setting of regularization hyper-parameters by setting $\lambda_\alpha = \lambda_\beta = \lambda_\gamma = \lambda_\delta = \lambda_\theta = \lambda$. To explore if a dynamic setting of those regularization hyper-parameters could lead to better results for RME model (RQ2-4), we set $\lambda_\alpha = \lambda_\beta = \lambda_1$, $\lambda_\gamma = \lambda_\delta = \lambda_\theta = \lambda_2$. Then we both vary λ_1 and λ_2 in {100, 50, 10, 5, 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001} while fixing the latent dimension size k. Next, we report the NDCG@5 for all 3 datasets. As shown in Figure 6, our model even get a higher performance with the dynamic setting. For example, it gains NDCG@5 = 0.1613 when $\lambda_1 = 100$ and $\lambda_2 = 0.005$ in MovieLens-10M dataset. Similarly, NDCG@5 = 0.1639 when $\lambda_1 = 0.5$, $\lambda_2 = 1$ in MovieLens-20M dataset. NDCG@5 = 0.2014 when $\lambda_1 = 100$, $\lambda_2 = 10$ in TasteProfile dataset. The dynamic setting produced 0.3~2% higher results than the static setting presented in Table 2.

So far, we compared the performance of our model and the baselines while varying values of hyper-parameters. We showed that our model outperformed the baselines in all cases, indicating that our model was less sensitive with regard to the hyper-parameters. We also showed that our model produced better results under the dynamic setting.

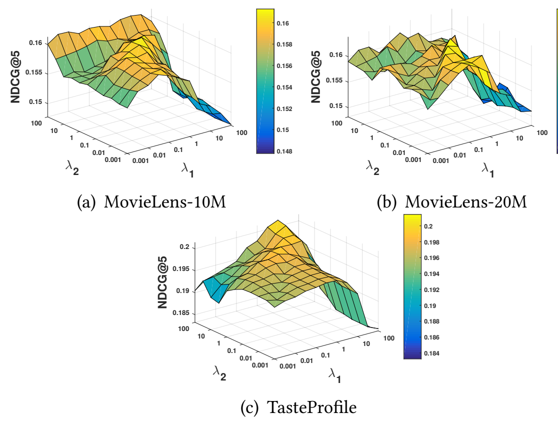
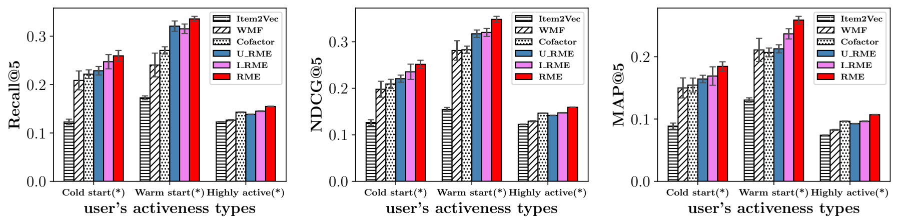


Figure 6: Performance of RME under a dynamic setting of regularization hyper-parameters. Set $\lambda_\alpha = \lambda_\beta = \lambda_1$, and $\lambda_{\gamma^{(+)}} = \lambda_{\gamma^{(-)}} = \lambda_\theta = \lambda_2$.

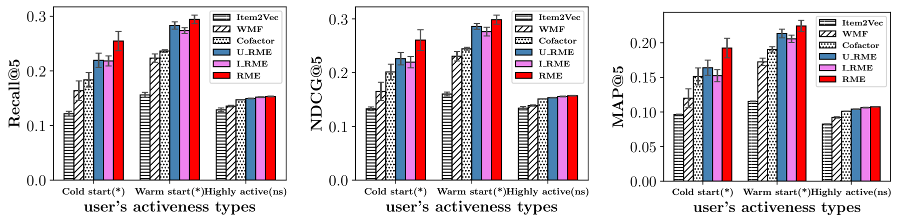
RQ3: Performance of models for different types of users. We sorted users by the ascending order of their activity level in terms of the number of liked items. Then we categorized them into three groups: (1) *cold-start users* who were in the first 20% of the sorted user list (i.e., their activity level is the lowest); (2) *warm-start users* who were in between 20% and 80% of the sorted user list; (3) *highly active users* who were in the last 20% of the sorted user list (i.e., the most active users). Then, we measured the performance of all the compared models for each of the user groups.

Figure 7 shows the performance of all the compared models in MovieLens-10M, MovieLens-20M and TasteProfile datasets. In MovieLens-10M (Figure 7(a)), our model significantly outperformed the baselines and the two variants in all three user groups, improving Recall@5 by 4.7~6.7%, NDCG@5 by 6.8~8.8%, and MAP@5 by 9.2~11.0% over the best compared method. In MovieLens-20M dataset (Figure 7(b)), our model significantly outperformed the baselines and its variants in 2 groups: *cold-start users* and *warm-start users*. It improved Recall@5 by 16.1%, 4.0%, 0.8%, NDCG@5 by 15.3%, 4.4%, 0.9%, MAP@5 by 17.3%, 5.1%, 1.1% in *cold-start users*, *warm-start users* and *highly-active users*, respectively. Specially, in both MovieLens-10M and MovieLens-20M datasets, our model on average much improved the baselines in *cold-start users* with Recall@5, NDCG@5 and MAP@5 by 27.9%, 24.8% and 23.3%, respectively. It shows the benefit of incorporating disliked item embeddings and user embeddings. In TasteProfile dataset (Figure 7(c)), our model significantly improved baselines in *highly-active users* group, improving Recall@5 by 6.8%, NDCG@5 by 7.4%, and MAP@5 by 10.0% comparing to the best state-of-the-art method, while improving Recall@5 by 5.0%, NDCG@5 by 4.9%, and MAP@5 by 5.7% comparing to its best variant. However, in *cold-start users* and *warm-start users* group, RME got an equal performance comparing with the baselines (i.e., the difference between our model and other methods are not significant).

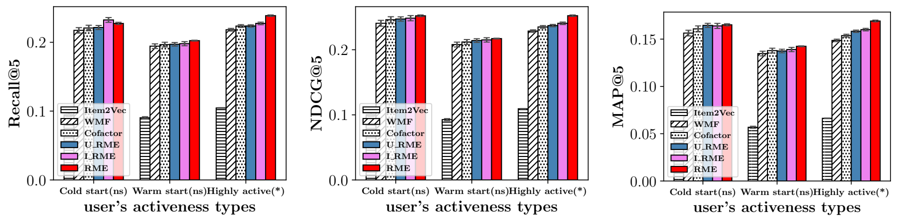
RQ4: Joint learning vs separate learning. What if we conduct learning separately for each part of our model? Will the separate learning model perform better than our joint learning model? To answer the questions, we built a separate learning model as follows: first, we learned latent representations of items by jointly



(a) Dataset: MovieLens-10M. RME outperformed baselines in all three groups (*p-value* < 0.05).



(b) Dataset: MovieLens-20M. RME outperformed baselines in *cold-start* and *highly-active* user groups (*p-value* < 0.05).



(c) Dataset: TasteProfile. RME outperformed baselines in *highly-active* user group (*p-value* < 0.05).

Figure 7: Performance of models for three user groups. Non-directional two-sample t-test was performed. * indicates significant (*p-value* < 0.05), and *ns* indicates not significant. The error bars are the average of standard errors in the 5 folds.

decomposing two SPPMI matrices $X^{(+)}$ and $X^{(-)}$ of liked item-item co-occurrences and disliked item-item co-occurrences, respectively. Then, we learned user's latent representations by minimizing the objective function in Equation (4), where the latent representations of items and item contexts were already learned and fixed. Next, we compared our joint learning model (i.e., RME) with the separate learning model in MovieLens-10M, MovieLens-20M, and TasteProfile datasets. Our experimental results show that our joint learning model outperformed the separate learning model by significantly improving Recall@5, NDCG@5 and MAP@5 at least 12.1%, 13.5% and 17.1%, respectively (p-value < 0.001).

6 RELATED WORK

Latent factor models (LFM): Some of the first works in recommendation focused on explicit feedback datasets (name some: [15, 30, 31]). Our proposed method worked well for both explicit and implicit feedback settings with almost equal performances.

In implicit feedback datasets, which have been trending recently due to the difficulty of collecting users' explicit feedback, properly treating/modeling missing data is a difficult problem [4, 21, 27]. Even though missing values are a mixture of negative feedback and unknown feedback, many works treated all missing data as negative instances [8, 14, 27, 34], or sampled missing data as negative instances with uniform weights [28]. This is suboptimal because treating negative instances and missing data differently can further improve recommenders' performance [13]. [25] proposed a bagging of ALS learners [14] to sample negative instances. He et al. [13] assumed that unobserved popular items have a higher chance of being negative instances. In our work, we attempted to non-uniformly sample negative instances in implicit feedback datasets and treated them as additional information to enhance our model performance. Specifically, we (i) designed an EM-like algorithm with a softmax function to draw personalized negative instances for each user; (ii) employed a word embedding technique to exploit the co-occurrence patterns among disliked items, further enriching their latent representations.

LFM with auxiliary information: In latent factor models, additional sources of information were incorporated to improve collaborative filter-based recommender systems (e.g., user reviews, item categories, and article information [2, 10, 23, 35]). However, we only used an user-item-preference matrix without requiring additional side information. Adding the side information into our model would potentially further improve its performance. But, it is not a scope of our work in this paper.

LFM with item embeddings: [36] incorporated message embedding for retweet prediction. Cao et al. [6] co-factorized the user-item interaction matrix, user-list interaction matrix, and item-list co-occurrences to recommend songs and lists of songs for users. [20] learned liked item embeddings with an equivalent matrix factorization method of skip-gram negative sampling (SGNS), and performed joint learning with matrix factorization. [3] exploited item embeddings using the SGNS method for item collaborative filtering. So far, the closest techniques to ours [3, 20] only considered liked item embeddings, but we proposed a joint learning model that not only considered LFM using matrix factorization with liked item embeddings, but also user embeddings and disliked item embeddings. Since integrating co-disliked item embedding is non-trivial for implicit feedback datasets, we also proposed an EM-like algorithm for extracting personalized negative instances for each user.

Word embeddings: Word embedding models [24, 26] represent each word as a vector of real numbers called word embeddings. In [19], the authors proposed an implicit matrix factorization that was equivalent to word2vec [24]. To extend word2vec, researchers proposed models that mapped paragraphs or documents to vectors [9, 18]. In our work, we applied word embedding techniques to learn latent representations of users and items.

7 CONCLUSION

In this paper, we proposed to exploit different co-occurrence information: co-disliked item-item co-occurrences and user-user co-occurrences, which were extracted from the user-item interaction matrix. We proposed a joint model combining WMF, co-liked embedding, co-disliked embedding and user embedding, following the recent success of word embedding techniques. Through comprehensive experiments, we successfully demonstrated that our model outperformed all baselines, significantly improving NDCG@20 by 5.6% in MovieLens-10M dataset, by 5.3% in MovieLens-20M dataset, and by 4.3% in TasteProfile dataset. We also analyzed how our model worked on different types of users in terms of their interaction activity levels. We observed that our model significantly improved NDCG@5 by 20.2% in MovieLens-10M, by 29.4% in MovieLens-20M for the *cold-start users* group. In the future extension of our model, we are interested in selecting contexts for users/items by setting a timestamp-based window size for timestamped datasets. In addition, we are also interested in incorporating co-disliked patterns among users (i.e., co-disliked user embeddings) into our model.

8 ACKNOWLEDGMENT

This work was supported in part by NSF grants CNS-1755536, CNS-1422215, DGE-1663343, CNS-1742702, DGE-1820609, Google Faculty Research Award, Microsoft Azure Research Award, and Nvidia GPU grant. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In *SIGKDD*. 19–28.

[2] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *RecSys*. 147–154.

[3] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *MLSP Workshop*. 1–6.

[4] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*. 1341–1350.

[5] Marcel Blattner, Yi-Cheng Zhang, and Sergei Maslov. 2007. Exploring an opinion network for taste prediction: An empirical study. *Physica A: Statistical Mechanics and its Applications* (2007), 753–758.

[6] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. 2017. Embedding Factorization Models for Jointly Recommending Items and User Generated Lists. In *SIGIR*. 585–594.

[7] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *TOIS* (2004), 143–177.

[8] Robin Devooght, Nicolas Kourtellis, and Amin Mantrach. 2015. Dynamic matrix factorization with priors on unknown values. In *SIGKDD*. 189–198.

[9] Nemanja Djuric, Hao Wu, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. 2015. Hierarchical neural language models for joint representation of streaming documents and their content. In *WWW*. 248–255.

[10] Elie Guàrdia-Sebaoun, Vincent Guigue, and Patrick Gallinari. 2015. Latent trajectory modeling: A light and efficient way to introduce time in recommender systems. In *RecSys*. 281–284.

[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[13] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. 549–558.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.

[15] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*. 426–434.

[16] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *SIGKDD*. 447–456.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009).

[18] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. 1188–1196.

[19] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*. 2177–2185.

[20] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *RecSys*. 59–66.

[21] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *WWW*. 951–961.

[22] Bing Liu, Wee Sun Lee, Philip S Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *ICML*. 387–394.

[23] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. 165–172.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[25] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *ICDM*. 502–511.

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.

[27] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. 2010. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *RecSys*. 71–78.

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[29] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW*. 175–186.

[30] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*. 791–798.

[31] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[32] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *SIGKDD*. 713–722.

[33] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Adv. Artificial Intellegence* (2009).

[34] Maksims Volkovs and Guang Wei Yu. 2015. Effective latent models for binary feedback in recommender systems. In *SIGIR*. 313–322.

[35] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*. 448–456.

[36] Can Wang, Qiudan Li, Lei Wang, and Daniel Dajun Zeng. 2017. Incorporating message embedding into co-factor matrix factorization for retweeting prediction. In *IJCNN*. 1265–1272.

[37] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. 2014. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems* (2014), 793–819.

[38] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR*. 785–788.

[39] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Applications in Management*. 337–348.