Collaborative Similarity Embedding for Recommender Systems

Chih-Ming Chen*
National Chengchi University
Taipei, Taiwan
104761501@nccu.edu.tw

Chuan-Ju Wang
Academia Sinica
Taipei, Taiwan
cjwang@citi.sinica.edu.tw

Ming-Feng Tsai†
National Chengchi University
Taipei, Taiwan
mftsai@nccu.edu.tw

Yi-Hsuan Yang
Academia Sinica
Taipei, Taiwan
yang@citi.sinica.edu.tw

ABSTRACT

We present collaborative similarity embedding (CSE), a unified framework that exploits comprehensive collaborative relations available in a user-item bipartite graph for representation learning and recommendation. In the proposed framework, we differentiate two types of proximity relations: direct proximity and k-th order neighborhood proximity. While learning from the former exploits direct user-item associations observable from the graph, learning from the latter makes use of implicit associations such as user-user similarities and item-item similarities, which can provide valuable information especially when the graph is sparse. Moreover, for improving scalability and flexibility, we propose a sampling technique that is specifically designed to capture the two types of proximity relations. Extensive experiments on eight benchmark datasets show that CSE yields significantly better performance than state-of-the-art recommendation methods.

ACM Reference Format:
Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative Similarity Embedding for Recommender Systems. In *Proceedings of ACM conference (WWW'19).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The task of recommender systems is to produce a list of recommendation results that match user preferences given their past behavior. Collaborative filtering (CF), a common yet powerful approach, generates user recommendations by taking advantage of the collective wisdom from all users [14]. Many CF-based recommendation algorithms have been shown to work well across various domains and been used in many real-world applications [27].

The core idea of model-based CF algorithms is to learn low-dimensional representations of users and items from either explicit user-item associations such as user-item ratings or implicit feedback such as playcounts and dwell time. This can be done by training a rating-based model with matrix completion to learn from observed user-item associations (either explicit or implicit feedback) to predict associations that are unobserved [1, 5, 12, 13, 15, 16, 18, 20, 21, 26, 36]. In addition to this rating-based approach, ranking-based methods have been proposed based on optimizing ranking loss; the ranking-based methods [21, 25, 33–35] have been found more suitable for implicit feedback. However, many existing model-based CF algorithms leverage only the user-item associations available in a given user-item bipartite graph. Thus, when the available user-item associations are sparse, these algorithms may not work well.

It has been noted that it is possible to mine from a user-item bipartite graph other types of collaborative relations, such as user-user similarities and item-item similarities, since users and items can be indirectly connected in the graph. Moreover, by taking random walks on the graph, it is possible to exploit higher-order proximity among users and items. Using item-item similarities in the learning process has been firstly studied by Liang *et al.* [18], who propose to jointly decompose the user-item interaction matrix and the item-item co-occurrence matrix with shared item latent factors. Hsieh *et al.* [12] propose to learn a joint metric space to encode both user preferences and user-user and item-item similarities. A recent work presented by Yu *et al.* [35] shows that jointly modeling user-item, user-user, and item-item relations outperforms competing methods that consider only user-item relations. In [7, 22], the higher-order proximity has been shown useful in graph embedding methods. In general, exploiting additional collaborative relations shows promise in learning better representations of vertexes in an information graph.

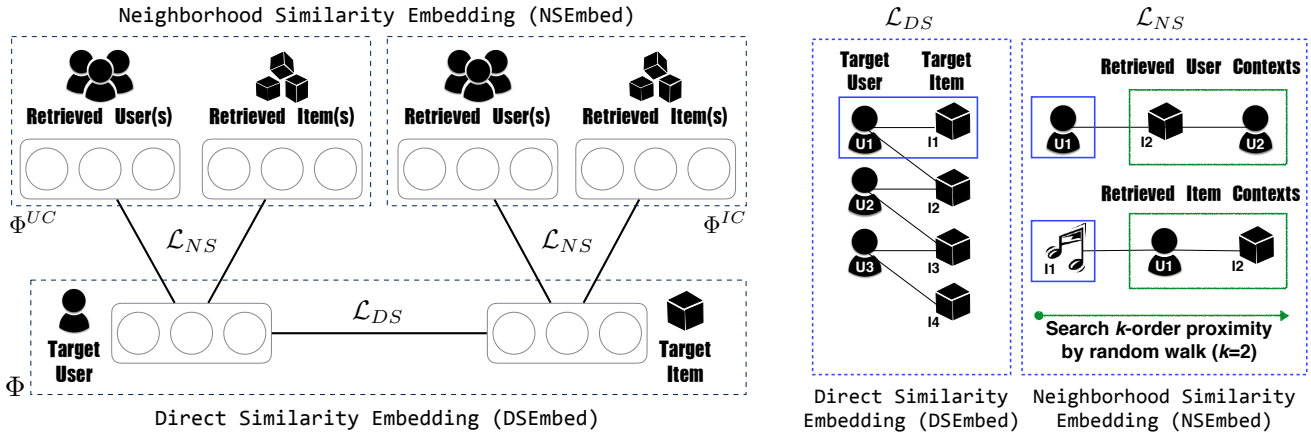
We note that these prior arts [7, 12, 18, 35] share the same core idea: using some specific methods to sample auxiliary information from a graph to augment the data for representation learning. However, there is a lack of a unified and efficient model that generalizes the underlying computation and aims at recommendation problems. For example, Liang *et al.* [18] consider only the item-item similarities but no other collaborative relations; Yu *et al.* [35] consider only ranking-based loss functions but not rating-based ones. Higher-order proximity is exploited in [6, 7, 22], which however deal with the general graph embedding problem not the recommendation one. Moreover, the model presented by [12] fails to manage large-scale user-item associations [29].

To address this discrepancy, in this paper we present *collaborative similarity embedding* (CSE), a unified representation learning

*Social Networks and Human-Centered Computing, Taiwan International Graduate Program, Institute of Information Science, Academia Sinica, Taiwan
†MOST Joint Research Center for AI Technology and All Vista Healthcare, Taiwan

WWW'19, May 2019, San Francisco

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4



(a) The bottom part depicts the direct similarity embedding module for user-item associations, whereas the upper left (right) part corresponds to modeling user-user (item-item, respectively) similarity with the neighborhood similarity embedding module. An optimization step includes a sampled pair of a user and an item in DSEmbed and multiple high-order relation pairs in NSEmbed.

(b) The left part shows that a target user-item pair (U1-I1) can be directly sampled from the observed edges for DSEmbed; the right part shows that for U1 (or I1), a 2-step random walk is applied to obtain the contexts used in the NSEmbed module.

Figure 1: An overview of the proposed CSE framework

framework for collaborative recommender systems with the aim of modeling the direct and in-direct edges of user-item interactions in a simple and effective way. CSE involves a *direct similarity* embedding module for modeling user-item associations as well as a *neighborhood similarity* embedding module for modeling user-user and item-item similarities. The former module provides the flexibility to implement various types of modeling techniques for user-item associations, whereas the later module models user-user and item-item relations via k-order neighborhood proximity. To simultaneously manage the two modules, we introduce *triplet embedding* into the proposed framework to ideally model user-user, item-item clustering and user-item relations in a single and joint-learning model, while most prior arts use only one or two embedding mappings in their methods. Moreover, the two sub-modules are fused by a carefully designed sampling technique for scalability and flexibility For scalability, the space complexity and time of convergence are both only linear with respect to the number of observed user-item associations. In addition, with the proposed sampling techniques, CSE provides the flexibility to shape different relation distributions in its optimization.

Extensive experiments were conducted on eight recommendation datasets that cover different user-item interaction types, levels of data sparsity, and data sizes. We compare the performance of CSE with classic methods such as matrix factorization (MF) and Bayesian personalized ranking (BPR) [25], recent methods that incorporate user-user and/or item-item relations [12, 18, 35], as well as several general graph embedding methods [6, 22]. The evaluation shows that CSE outperforms the competing methods for seven out of the eight datasets.

In summary, we propose a simple yet effective representation learning framework aiming at making the best use of information embedded in observed user-item associations for recommendation. Our framework advances the state-of-the-art recommendation algorithms along the following five dimensions.

(1) The CSE serves as a generalized framework that models comprehensive pairwise relations among users and items with a unified objective function in a simple and effective manner.
(2) The proposed sampling technique enables the suitability of CSE for large-scale user-item recommendations.
(3) We provide model analyses for flexibility, scalability and time and space complexity of the proposed CSE.
(4) We report extensive experiments over eight recommendation datasets covering different user-item interaction types, levels of data sparsity, and data sizes, demonstrating the robustness, efficiency, and effectiveness of our framework.
(5) For reproducibility, we share the source code of CSE online at a GitHub repo,[1] by which the learning process can be done within an hour for each dataset performed in this work.

The rest of this paper is organized as follows. In Section 2, we present the proposed CSE framework in detail, including the problem definition, the two similarity embedding modules, and the sampling techniques. We then provide model analyses for flexibility, scalability and time and space complexity of the proposed CSE in Section 3. Experimental results are provided and discussed in Section 4. Section 5 concludes the paper.

2 PROPOSED CSE FRAMEWORK

Problem Formulation. A recommender system provides a list of ranked items to users based on their historical interactions with

[1] https://github.com/cnclabs/proNet-core

items. Let U and I denote the sets of users and items, respectively. User-item associations can be presented as a bipartite graph $G = (V, E)$, where $V = \{v_1, \ldots, v_{|V|}\} = U \cup I$, and E represents the set of observed user-item associations. Note that for explicit rating data, the weights of the user-item preference edges can be positive real numbers, whereas for implicit interactions, the bipartite graph becomes a binary graph. The goal of the CSE framework is to obtain an embedding matrix $\Phi \in \mathbb{R}^{|V| \times d}$ that maps each user and item into a d-dimensional embedding vector for item recommendation; that is, with the learned embedding matrix Φ, for a user $v_i \in U$, the proposed framework generates the top-N recommended items via computing the similarity between the embedding vector of the user, i.e., Φ_{v_i}, and those of all items, i.e., Φ_{v_j} for all $v_j \in I$, where Φ_{v_x} denotes the row vector for vertex $v_x \in V$ from matrix Φ.

Framework Overview. Figure 1 provides an overview of CSE. In the figure, CSE consists of two similarity embedding modules: a direct similarity embedding (DSEmbed) module to model user-item associations, and a neighborhood similarity embedding (NSEmbed) module to model user-user and item-item similarities. The DSEmbed model provides the flexibility to implement two mainstream types of modeling techniques: rating-based and ranking-based models to preserve direct proximity of user-item associations; NSEmbed, in turn, models user-user and item-item relations using the contexts within a k-step random walk, as shown in Fig. 1(b), to preserve k-order neighborhood proximity between users and items. To minimize the sum of the losses from DSEmbed and NSEmbed modules, which are denoted as $\mathcal{L}_{DS}$ and $\mathcal{L}_{NS}$ respectively, the objective function of the proposed framework is designed as

$$\mathcal{L} = \mathcal{L}_{DS} + \lambda \mathcal{L}_{NS},$$

where λ controls the balance between the two losses. The rationale behind this design is that $\mathcal{L}_{DS}$ controls the optimization of the embedding vectors towards preserving direct user-item associations, and $\mathcal{L}_{NS}$ encourages users/items sharing similar neighbors to be close to one another in the learned embedding space.

2.1 Direct Similarity Embedding (DSEmbed) Module

Definition 2.1. **(Direct Proximity)** Given a bipartite graph $G = (V, E)$, the *direct proximity* between a user $v_i \in U$ and an item $v_j \in I$ is represented by the presence of an edge $(v_i, v_j) \in E$ between these two vertices. If there is no edge between user v_i and item v_j, then their direct proximity is defined as 0.

The DSEmbed module is designed to model the direct proximity of the user-item associations defined in Definition 2.1. For a rating-based approach, the objective is to find the embedding matrix Φ that maximizes the log-likelihood function of observed user-item pairs:

$$\begin{aligned} &\arg\max_{\Phi} \sum_{(v_i, v_j) \in E} \log p(v_i, v_j | \Phi) \\ &= \arg\min_{\Phi} \sum_{(v_i, v_j) \in E} -\log p(v_i, v_j | \Phi). \end{aligned} \tag{1}$$

In contrast, a ranking-based approach cares more about whether we can predict stronger association between a 'positive' user-item pair $(v_i, v_j) \in E$ than a 'negative' user-item pair $(v_i, v_k) \in \bar{E}$ [25], where $\bar{E}$ denotes the set of edges for all the unobserved user-item associations. This can be approached by maximizing the log-likelihood function of observed user-item pairs over unobserved user-item pairs for each user:

$$\begin{aligned} &\arg\max_{\Phi} \sum_{(v_i, v_j, v_k)} \log p(v_j >_i v_k | \Phi) \\ &= \arg\min_{\Phi} \sum_{(v_i, v_j, v_k)} -\log p(v_j >_i v_k | \Phi), \end{aligned} \tag{2}$$

where $v_i \in U$ and $v_j, v_k \in I$, and $>_i$ indicates that user v_i prefers item v_j over item v_k. In the above two equations, $p(v_i, v_j | \Phi)$ and $p(v_j >_i v_k | \Phi)$ is calculated by

$$p(v_i, v_j | \Phi) = \sigma\left(\Phi_{v_i} \cdot \Phi_{v_j}\right),$$

and

$$p(v_j >_i v_k | \Phi) = \sigma\left(\Phi_{v_i} \cdot \Phi_{v_j} - \Phi_{v_i} \cdot \Phi_{v_k}\right),$$

respectively, and $\sigma(\cdot)$ denotes the sigmoid function.

2.2 Neighborhood Similarity Embedding (NSEmbed) Module

Definition 2.2. **(k-Order Neighborhood Proximity)** Given a bipartite graph $G = (V, E)$ representing the observed user-item associations of the set of users and items in $V = U \cup I$, the *k-order neighborhood proximity* of a pair of users (or items) is defined as the similarity between their neighborhood network structures retrieved by k-step random walks. Mathematically speaking, given the k-order neighborhood structures of a pair of users (or items), $v_i, v_j \in U$ (or $v_i, v_j \in I$, respectively), which are denoted as two sets of neighbor nodes N_{v_i} and N_{v_j}, with $|N_{v_i}| = |N_{v_j}| = k$, the k-order neighborhood proximity between v_i and v_j is decided by the similarity between these two sets N_{v_i} and N_{v_j}. If there are no shared neighbors between v_i and v_j, the neighborhood proximity between them is 0.

The NSEmbed module is designed to model k-order neighborhood proximity for capturing user-user and item-item similarities. Given a set of neighborhood relations for users (or items) $S_U = \{(v_i, v_j) | \forall v_i \in U, v_j \in N_{v_i}\}$ (or $S_I = \{(v_i, v_j) | \forall v_i \in I, v_j \in N_{v_i}\}$, respectively), the NSEmbed module seeks a set of embedding matrices $\Phi, \Phi^{UC}, \Phi^{IC} \in \mathbb{R}^{|V| \times d}$ that maximizes the likelihood of all pairs in S_U (or S_I, respectively), where Φ is a *vertex mapping matrix* akin to that used in the DSEmbed module, and Φ^{UC} and Φ^{IC} are two *context mapping matrices*. Note that each vertex (representing a user or an item) plays two roles for modeling the neighborhood proximity: 1) the vertex itself and 2) the context of other vertices [3, 7, 22, 28, 37]. With this design, the embedding vectors of vertices that share similar contexts are thus closely located in the learned vector space. Therefore, the maximization of the likelihood function can be defined as

$$\begin{aligned} \arg\max_{\Phi, \Phi^{UC}, \Phi^{IC}} \quad & \prod_{(v_i, v_j) \in S_U} p(v_j | v_i; \Phi; \Phi^{UC}) \\ & + \prod_{(v_i, v_j) \in S_I} p(v_j | v_i; \Phi; \Phi^{IC}). \end{aligned}$$

Similar to Eqs. (1) and (2), the above objective function becomes

$$\arg\min_{\Phi,\Phi^{UC},\Phi^{IC}} \quad \sum_{(v_i,v_j)\in S_U} -\log p(v_j|v_i;\Phi;\Phi^{UC}) + \sum_{(v_i,v_j)\in S_I} -\log p(v_j|v_i;\Phi;\Phi^{IC}), \tag{3}$$

where

$$p(v_j|v_i;\Theta) = \begin{cases} \sigma\left(\Phi_{v_i}\cdot\Phi^{UC}_{v_j}\right) & \text{if } v_i \in U, \\ \sigma\left(\Phi_{v_i}\cdot\Phi^{IC}_{v_j}\right) & \text{if } v_i \in I. \end{cases} \tag{4}$$

It is worth mentioning that most prior arts use only one or two embedding mappings; while the former approach fails to consider high-order neighbors (e.g., [4, 9, 16, 28, 31]), the later one cannot model user-user, item-item, and user-item relations simultaneously (e.g., [1, 3, 7, 22, 23, 37]). Our newly designed triplet embedding solution (i.e., $\Theta = \{\Phi, \Phi^{UC}, \Phi^{UC}\}$) can ideally model user-user, item-item clustering and user-item relations in a single and joint-learning model.

2.3 Sampling-based Expectation Loss

In order to minimize the above objective functions, we need to go through all the pairs in E for Eq. (1), E and $\bar{E}$ for Eq. (2), and S_U and S_I for Eq. (3), to compute all the pairwise losses. This is not feasible in real-world recommendation scenarios as the complexity is $O(|V| \times |V|)$. To address this, we propose a sampling technique to work in tandem with the above two modules to enhance CSE's scalability and flexibility in learning user and item representations from large-scale datasets.

In CSE, the DSEmbed and NSEmbed modules are fused with the shareable data sampling technique described below. For each parameter update, we first sample an observed user-item pair $(v_i, v_j) \in E$, as shown as `U1` and `I1` in Fig. 1(b), where $v_i \in U$ and $v_j \in I$. Then, we search for the k-order neighborhood structures of user v_i and item v_j via the k-step random walks. To improve computational efficiency, we use negative sampling [28]. Consequently, for a rating-based approach (see Eq. (1)), the expected sampled loss of the DSEmbed module can be re-written as

$$\mathcal{L}_{DS} = \mathbb{E}_{(v_i,v_j)\sim E}\left[-\log p(v_i,v_j|\Phi)\right] + \sum_{M}\mathbb{E}_{(v_k,v_h)\sim\bar{E}}\left[\log p(v_k,v_h|\Phi)\right], \tag{5}$$

where M denotes the number of negative pairs adopted. For a ranking-based approach (see Eq. (2)), the DSEmbed module can be re-written as

$$\mathcal{L}_{DS} = \mathbb{E}_{(v_i,v_k)\sim\bar{E}}\left[\mathbb{E}_{(v_i,v_j)\sim E}\left[-\log p(v_j >_i v_k|\Phi)\right] \mid v_i\right]. \tag{6}$$

Note that for the ranking-based approach, there is no need to explicitly include M negative sample pairs as this kind of method naturally involves negative pairs from $\bar{E}$. Similarly, given a user or an item vertex v_i, its k-order neighborhood structure N_{v_i} is composed of nodes in the k-step random walks surfing on G, $\mathcal{W}_{v_i} = (\mathcal{W}^0_{v_i}, \mathcal{W}^1_{v_i}, \mathcal{W}^2_{v_i}, \ldots, \mathcal{W}^k_{v_i})$, where the vertex for $\mathcal{W}^j_{v_i}$ is randomly chosen from the neighbors of the vertex v given $\mathcal{W}^{j-1}_{v_i} = v$ and $\mathcal{W}^0_{v_i} = v_i$. The expected sampled loss of the NSEmbed module can be re-written as

$$\mathcal{L}_{NS} = \mathbb{E}_{(v_i,v_j)\sim S_U}\left[-\log p(v_j|v_i;\Phi;\Phi^{UC})\right] + \sum_{M}\mathbb{E}_{(v_i,v_j)\sim\bar{E}}\left[\log p(v_j|v_i;\Phi;\Phi^{UC})\right] + \mathbb{E}_{(v_i,v_j)\sim S_I}\left[-\log p(v_j|v_i;\Phi;\Phi^{IC})\right] + \sum_{M}\mathbb{E}_{(v_i,v_j)\sim\bar{E}}\left[\log p(v_j|v_i;\Phi;\Phi^{IC})\right]. \tag{7}$$

Since $\mathcal{L}_{DS}$ and $\mathcal{L}_{NS}$ are described in a sampling-based expectation form, CSE provides the flexibility for accommodating arbitrary distributions of positive and negative data. In the following experiments, we produce the positive data according to primitive edges distribution of given user-item graph. As to negative sampling, we propose to directly sample the negative data from whole data collection instead of unobserved data collection.

2.4 Optimization

In the optimization stage, we use asynchronous stochastic gradient descent (ASGD) [24] to efficiently update the parameters in parallel. The model parameters are composed of the three embedding matrices $\Theta = \{\Phi, \Phi^{UC}, \Phi^{IC}\}$, each having the size $O(|V|d)$. They are updated with learning rate α according to

$$\Theta \leftarrow \Theta - \alpha\left(\frac{\partial\mathcal{L}_{DS}}{\partial\Theta} + \lambda\left(\frac{\partial\mathcal{L}_{NS}}{\partial\Theta}\right) - \lambda_V\|\Phi\|\right), \tag{8}$$

where λ_V is a hyper-parameter for reducing the risk of overfitting.

3 MODEL ANALYSIS

The CSE framework not only modularizes the modeling of pairwise user-item, user-user, and item-item relations, but also integrates them into a single objective function through shared embedding vectors. Together with the DSEmbed and NSEmbed sub-modules for modeling these relations, CSE involves a novel sampling technique to improve scalability and flexibility. To give a clear view of CSE, we further provide the comparison to general graph embedding and deep learning models.

3.1 Scalability

Typical factorization methods usually work on a sparse user-item matrix and do not explicitly model high-order connections from the corresponding user-item bipartite graph $G = (V, E)$. Several methods, including our CSE, propose to explicitly incorporate high-order connections for modeling user-user and item-item relations into the recommendation models to improve performance. As discussed in [17–19], such modeling can then be seen as conducting matrix factorization on a $|V| \times |V|$ point-wise mutual information (PMI) matrix:

$$\text{PMI}(v_i, v_j) = \log\left(\frac{p(v_i,v_j)}{p(v_i)p(v_j)}\right) - \log M.$$

Recall that $V = U \cup I$, and M denotes the number of negative pairs adopted in Eqs. (5) and (7). However, given the high-order connections for modeling user-user and item-item relations, most $\text{PMI}(v_i, v_j)$ for $v_i, v_j \in U$ (or $v_i, v_j \in I$) are nonzero and thus the PMI matrix is considered non-sparse. Conducting matrix factorization on such a matrix is computationally expensive in both time

Dataset	#Users	#Items	#Edges	$\frac{\#\text{Edges}}{\#\text{Users}}$	Density	Edge type	Network type
Frappe[a]	957	4,028	96,202	100.52	2.50%	click count	app-clicks
CiteULike[b]	5,551	16,980	210,504	37.92	0.22%	like/dislike	references
Netflix[c]	65,533	17,759	25,120,129	383.32	2.15%	5-star	movie-ratings
MovieLens-Latest[d]	259,137	40,110	24,404,096	94.17	0.23%	5-star	movie-ratings
Last.fm-360K[e]	359,347	294,015	17,559,530	48.86	0.02%	play count	artist-plays
Amazon-Book[f]	603,668	367,982	8,898,041	14.73	0.004%	5-star	book-ratings
Epinions-Extend[g]	755,760	120,492	13,668,319	18.08	0.02%	5-star	product-reviews
Echonest[h]	1,019,318	384,546	48,373,586	47.45	0.01%	play count	song-plays

Table 1: Statistics of the datasets considered in our experiments.

[a] http://baltrunas.info/research-menu/frappe
[b] http://www.wanghao.in/CDL.htm
[c] http://academictorrents.com/
[d] https://grouplens.org/
[e] http://www.dtic.upf.edu//~ocelma/MusicRecommendationDataset/
[f] http://jmcauley.ucsd.edu/data/amazon/
[g] http://www.trustlet.org/epinions.html
[h] https://labrosa.ee.columbia.edu/millionsong/tasteprofile

and space, and is therefore infeasible in many large-scale recommendation scenarios.

To explicitly consider all the collaborative relations into our model while keeping it practical for large-scale datasets, the CSE uses the sampling technique together with k-step random walks surfing on G to preserve direct proximity of user-item associations as well as harvest the high-order neighborhood structures of users and items. By doing so, we approximate factorization of the corresponding PMI matrix and thus reduce the complexity in space and the training time.

3.2 Flexibility

In the CSE, the DSEmbed and NSEmbed modules are united with the sampling technique. In addition to improved scalability, such a sampling perspective facilitates the shaping of different relation distributions for optimization via different weighting schemes or sampling strategies. Here we resort to use the perspective of KL divergence to explain this model characteristic. Specifically, minimizing the losses in our framework (see Eqs. (5), (6), and (7)) can be related to minimizing the KL divergence of two probability distributions [28]. Suppose there are two distributions over the space $V \times V$: $\hat{p}(\cdot,\cdot)$ and $p(\cdot,\cdot)$, denoting the empirical distribution and the target distribution, respectively, we have

$$\begin{aligned} & \arg\min_{p} \mathrm{KL}(\hat{p}(\cdot,\cdot), p(\cdot,\cdot)) \\ = \quad & \arg\min_{p} -\sum_{(v_i,v_j)\in E} \hat{p}(v_i,v_j) \log\left(\frac{p(v_i,v_j)}{\hat{p}(v_i,v_j)}\right) \\ \propto \quad & \arg\min_{p} -\sum_{(v_i,v_j)\in E} \hat{p}(v_i,v_j) \log p(v_i,v_j). \end{aligned} \tag{9}$$

In Eq. (9), the empirical distribution $\hat{p}(\cdot,\cdot)$ can be treated as the probability density (mass) function of the distribution in our loss functions, from which each pair of vertices (v_i, v_j) is sampled. This indicates that applying different weighting schemes or different sampling strategies (i.e., different $\hat{p}(\cdot,\cdot)$) in CSE shapes different relation distributions for learning representations.

3.3 Complexity

The time and space complexity of the proposed method depends on the implementation. The training procedure of CSE framework involves a sampling step and an optimization step. Since all the required training pairs, including observed associations and unobserved associations, can be derived from the user-item bipartite graph G, we adopt the compressed sparse alias rows (CSAR) data structure to perform weighted edge sampling for direct similarity and weighted random walk for neighborhood similarity [2]. With CSAR data structure, sampling an edge requires only $O(1)$ and the overall demand space complexity is linearly increased with the number of positive edges $O(|E|)$.[2] As to the optimization, SGD-based update has a closed form so that updating the embedding of a vertex in a batch depends only on the dimension size $O(d)$. As for the time of convergence, many studies on graph embedding as well as our method empirically show that the required total training time for the convergence of embedding learning is also linear in $|E|$ [8].

3.4 Comparison to General Graph Embedding Models

General graph embedding algorithms, such as DeepWalk [22] and node2vec [7], can be used for the task of recommendation. Yet, we do not focus on comparing the proposed CSE with general graph embedding models and only provide the results of DeepWalk because many prior works on recommendation [21, 36] have shown that many of the baseline methods considered in our paper outperform these general graph embedding algorithms. The main reason for this phenomenon is that most graph embedding methods cluster vertices that have similar neighbors together, and thus make the users apart from the items because user-item interactions typically form a bipartite graph.

[2] Note that the space for the learned embedding matrices $\Theta = \{\Phi, \Phi^{UC}, \Phi^{IC}\}$ is $O(|V|)$ and $|V| \ll |E|$.

3.5 Comparison to Deep Learning Models

Our method, and the existing methods we discussed and compared in the experiments, focus on improving the modeling quality of user and item embeddings that can be directly and later used for user-item recommendation with similarity computation. Many approximation techniques, such as approximate nearest neighbor[3](ANN), can be applied to speed up the similarity computation between user and item embeddings, which facilitates real-time online predictions and makes the recommendation scalable to large-scale real-world datasets. In contrast, many deep learning methods, including NCF [11], DeepFM [10], etc, do not learn the directly comparable embeddings of users or items. There are a few deep learning methods (e.g., Collaborative Deep Embedding [32] and DropoutNet [30]) that can produce user and item embeddings, but to our knowledge, efficiency is still a major concern of these methods. Therefore, improving the embedding quality is still a critical research issue for building up a recommender system especially when the computation power is limited in real-world application scenarios. For readability and to maintain the focus of this work, we opt for not comparing the deep learning methods in our paper. It is also worth mentioning that our solution can obtain user and item embeddings within only an hour for every large dataset listed in the paper; the efficiency and scalability is thus one of the highlights of the proposed method.

4 EXPERIMENT

4.1 Settings

4.1.1 Datasets and Preprocessing. To examine the capability and scalability of the proposed CSE framework, we conducted experiments on eight publicly available real-world datasets that vary in terms of domain, size, and density, as shown in Table 1. For each of the datasets, we discarded the users who have less than ten associated interactions with items. In addition, we converted each data into implicit feedback:[4] 1) for 5-star rating datasets, we transformed ratings higher than or equal to 3.5 to 1 and the rest to 0; 2) for count-based datasets, we transformed counts higher than or equal to 3 to 1 and the rest to 0; 3) for the CiteULike dataset, no transformation was conducted as it is already a binary preference dataset.

4.1.2 Baseline Algorithms. We compare the performance of our model with the following eight baseline methods: 1) POP, a naive popularity model that ranks the items by their degrees, 2) DeepWalk [22], a classic algorithm of network embedding, 3) WALS [13], a weighted rating-based factorization model, 4) ranking-based factorization models: BPR [25], WARP [33], and K-OS [34], 5) BiNE [6], a network embedding model specialized for bipartite networks, and 6) recent advanced models considering user-user/item-item relations: coFactor [18], CML [12] and WalkRanker [35], Note that except for POP, the embedding vectors for users and items learned by these competitors as well as by our method can be directly used for item recommendations. Additionally, while CML adopts Euclidean distance as the scoring function, all other methods including ours utilize the dot product to calculate the score of a pair of user-item embedding vectors. The experiments for WALS and BPR were conducted using the matrix factorization library QMF,[5] and those for WARP and K-OS were conducted using LightFM;[6] for coFactor, CML, and WalkRanker, we used the code provided by the respective authors.

4.1.3 Experimental Setup. For all the experiments, the dimension of embedding vectors was fixed to 100; the values of the hyper-parameters for the compared method were decided via implementing a grid search over different settings, and the combination that leads to the best performance was picked. The ranges of hyper-parameters we searched for the compared methods are listed as follows.

- learning rate: [0.0025, 0.01, 0.025, 0.1]
- regularization: [0.00025, 0.001, 0.0025, 0.01, 0.025, 0.1]
- training epoch: [10, 20, 40, 80, 160]
- sampling time: $[20\times|E|, 40\times|E|, 60\times|E|, 80\times|E|, 100\times|E|]$
- walk time: [10, 40, 80]
- walk length: [40, 60, 80]
- window size: [2, 3, 4, 5, 6, 8, 10]
- stopping probability for random walk: [0.15, 0.25, 0.5, 0.75]
- k-order: [1, 2, 3]
- rank margin: 1 (commonly used default value)
- number of negative samples: 5 (commonly used default value)

For our model, the learning rate α was set to 0.1, λ_V was set to 0.025; the hyper-parameter λ was set to 0.05 and 0.1 for rating-based CSE and ranking-based CSE, respectively, and k was set to 2 as the default value. The sensitivity of CSE parameters are additionally reported. For each dataset, the sample time for convergence depends on the number of non-zero user-item interaction edges and is set to $80\times|E|$. Sensitivity analysis for k and λ and convergence analysis are later provided in the section for convergence analyses.

4.1.4 Evaluations. The performance is evaluated between the recommended list R_u containing top-N recommended items and the corresponding ground truth list T_u for each user u. We consider following two commonly-used metrics over these N recommended results:

- Recall: Denoted as Recall@N, which describes the fraction of the ground truth (i.e., the user preferred items) that are successfully recommended by the recommendation algorithm:

$$\text{Recall@}N = \sum_{u\in U}\frac{1}{|U|}\frac{\sum_{v\in R_u}\mathbb{1}(v\in T_u)}{\min(N,|T_u|)}.$$

- Mean Average Precision: Denoted as mAP@N, computing the mean of the average precision at k (AP@k) for each user

[3] https://github.com/erikbern/ann-benchmarks

[4] Note that in real-world scenarios, most feedback is not explicit but implicit [25]; we here converted the datasets into implicit feedback as most of the recent developed methods focus on dealing with such type of data. However, our method is not limited to binary preference since the presented sampling technique has the flexibility to manage arbitrary weighted edge distributions and rating estimation is also allowed in the proposed RATE-CSE.

[5] https://github.com/quora/qmf

[6] https://github.com/lyst/lightfm

	Frappe		CiteULike		Netflix		MovieLens-Latest	
	Recall@10	mAP@10	Recall@10	mAP@10	Recall@10	mAP@10	Recall@10	mAP@10
Pop	0.1750	0.0708	0.0270	0.0114	0.0861	0.0359	0.0882	0.0289
DeepWalk [22]	0.0430	0.0256	0.0875	0.0458	0.0235	0.0112	0.0207	0.0061
WALS [13]	0.1632	0.1117	0.1851	0.0915	0.1214	0.0471	0.2350	†0.1682
BPR [25]	0.2785	0.1550	0.0861	0.0426	0.1496	0.0757	0.2163	0.1130
WARP [33]	0.3012	0.1796	0.1468	0.0813	†0.1887	†0.1004	†0.2712	0.1651
K-OS [34]	0.3018	0.1914	0.1356	0.0756	0.1783	0.0868	0.2522	0.1641
BiNE [6]	0.2159	0.1201	0.0422	0.0201	-	-	-	-
coFactor [18]	0.2110	0.1309	0.1323	0.0721	-	-	-	-
CML [12]	†0.3311	0.1958	0.1740	0.1008	0.1035	0.0444	0.1109	0.0957
WalkRanker [35]	0.3286	†**0.2099**	†0.2059	†0.1192	0.1090	0.0483	0.1307	0.0351
RATE-CSE	**0.3347**	0.2047	*__0.2362__	*__0.1452__	*0.2014	*0.1039	*__0.3225__	*__0.1990__
Improv. (%)	+1.0%	−2.4%	+14.7%	+21.9%	+6.7%	+3.5%	+18.9%	+18.3%
RANK-CSE	0.3155	0.2005	0.1993	*0.1228	*__0.2156__	*__0.1202__	*0.3094	*0.1902
Improv. (%)	−4.7%	−4.4%	−3.2%	+3.0%	+14.2%	+19.7%	+14.1%	+13.1%

	Last.fm-360K		Amazon-Book		Epinions-Extend		Echonest	
	Recall@10	mAP@10	Recall@10	mAP@10	Recall@10	mAP@10	Recall@10	mAP@10
Pop	0.0309	0.0133	0.0053	0.0015	0.0450	0.0246	0.0257	0.0104
WALS [13]	0.1621	0.0857	†0.0540	†0.0227	0.1479	0.0634	0.1287	†0.0638
BPR [25]	0.1120	0.0545	0.0248	0.0119	0.1126	0.0579	0.0499	0.0210
WARP [33]	0.1556	0.0832	0.0457	0.0199	†0.1509	†0.0775	0.1001	0.0447
K-OS [34]	†0.1641	†0.0888	0.0511	0.0215	0.1493	0.0766	†0.1249	0.0597
CML [12]	0.0496	0.0199	0.0129	0.0052	0.1171	0.0629	0.0357	0.0195
WalkRanker [35]	0.0233	0.0088	0.0080	0.0036	0.0560	0.0289	0.0309	0.0133
RATE-CSE	*0.1687	*0.0909	0.0540	0.0240	*0.1659	0.0788	0.1260	0.0605
Improv. (%)	+2.8%	+2.3%	+0.0%	+5.7%	+9.9%	+1.7%	+0.8%	−0.5%
RANK-CSE	*__0.1762__	*__0.0970__	*__0.0625__	*__0.0274__	*__0.1767__	*__0.0921__	*__0.1358__	*__0.0679__
Improv. (%)	+8.2%	+14.4%	+15.7%	+20.7%	+17.0%	+20.2%	+8.7%	+6.4%

Table 2: Recommendation performance. The † symbol indicates the best performing method among all the baseline methods; '*' and '%Improv.' denote statistical significance at $p < 0.01$ with a paired t-test and the percentage improvement of the proposed method, respectively, with respect to the best performing baseline.

u as

$$\text{mAP@}N = \frac{1}{|U|}\sum_{u\in U}\text{AP}_u@k = \frac{1}{|U|}\sum_{u\in U}\frac{\sum_{k=1}^{N}\text{P}_u(k)\times\mathbb{1}(r_k\in T_u)}{\min(N,|T_u|)}, \quad (10)$$

where r_k is the k-th recommended item and $\text{P}_u(k)$ denotes the precision at k for user u. This is a rank-aware evaluation metric because it considers the positions of each recommended item. For each dataset, the reported performance was averaged over 10 times; in each time, we randomly split the data into 80% training set and 20% testing set.

4.2 Results

4.2.1 Recommendation Performance Comparison. The results for the ten baseline methods along with the proposed method are listed in Table 2, where RATE-CSE and RANK-CSE denote two versions of our method that employ respectively rating-based and ranking-based loss functions for user-item associations. Note that the best results are always indicated by the bold font, and for coFactor and BiNE we report only the experimental results on Frappe and CiteULike because of resource limitations.[7] As discussed in Section 3.4, DeepWalk is not suitable for user-item recommendation as it make the users apart from items in the embedding space. In addition, observe that BiNE does not perform well in our experiments; such a result is due to the fact that BiNE is a general network embedding model and thus does not incorporate the regularizer in their objective function, which is however an important factor for the robustness of recommendation performance. Comparing the performance of the other baseline methods, we observe that

[7]While the memory usage of coFactor implementation is $O(|V|^2)$, BiNE's requires extensive computational time, e.g., more than 24 hours to learn the embedding for the large dataset, Movielens-Latest.

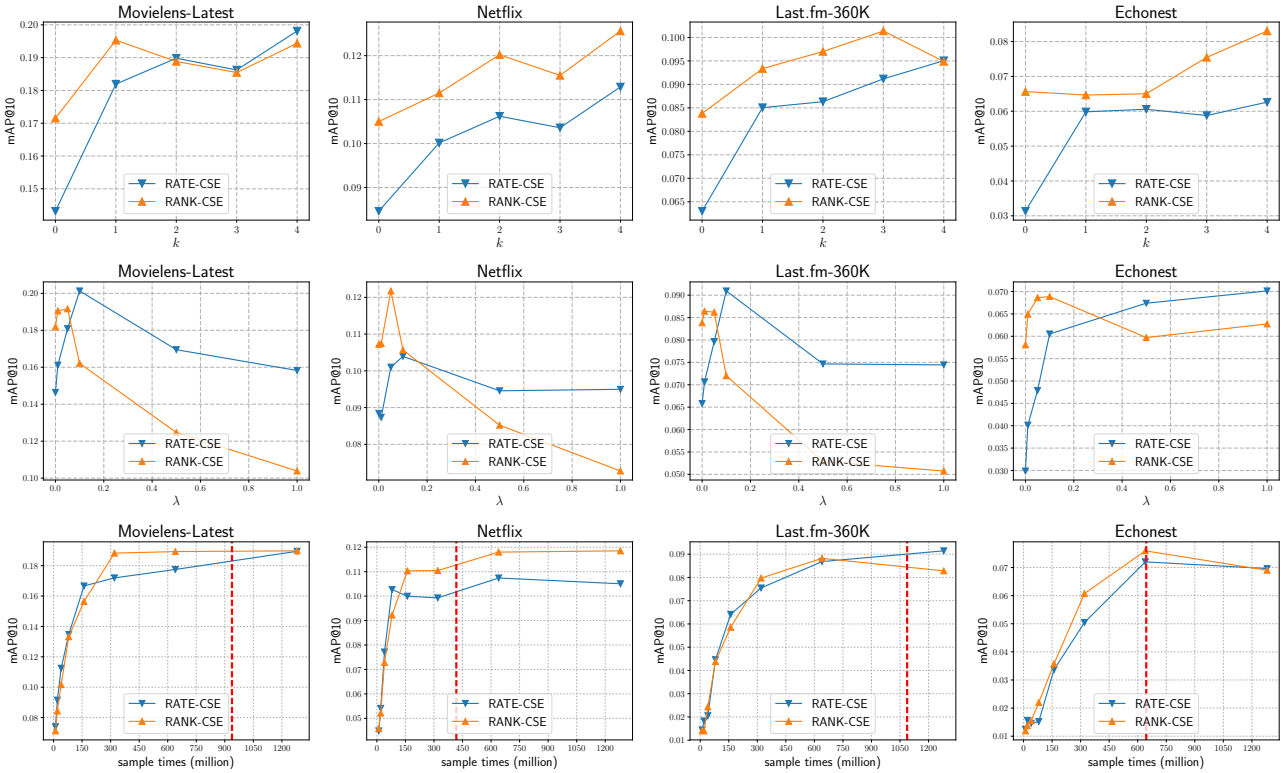


Figure 2: Sensitivity and convergence analyses

the performance of WALS, WARP and K-OS is very competitive. That is, these methods achieve the top performance among all the baselines on several datasets. The performance of WalkRanker and CML, on the other hand, seems satisfactory only on two rather small datasets – Frappe and CiteULike – and performs more poorly on most of the other datasets.

We observe that our method achieves the best results in terms of both Recall@10 and mAP@10 for most datasets. Moreover, RANK-CSE generally outperforms RATE-CSE in the experiments, re-confirming that using a ranking-based loss is indeed better for datasets with binary implicit feedbacks [25, 33, 34]. Specifically, except for Frappe, RATE-CSE or RANK-CSE achieves significantly much better performance than the best performing baseline methods with a maximum improvement of +20.7%.

4.2.2 Parameter Sensitivity and Convergence Analyses. Figure 2 shows the results of the sensitivity analysis on two hyper-parameters k and λ in the first and second rows, respectively, and those of the convergence analysis based on sample times in the third row.[8] We first observe that increasing the order k of modeling neighborhood proximity between users or items improves the performance in general. We first observe from Figure 2 is that the optimal value of k is data dependent and has to be empirically tuned considering the trade-off between accuracy and time/space complexity. In general, a larger k leads to better result, and from our experience, the result would reach a plateau when k is sufficiently large (e.g., when $k > 3$).

[8]Note that due to space limits, we report the results for the four largest datasets only.

The second row of Figure 2 shows how the balancing parameter λ affects performance: RANK-CSE obtains better performance with a value around 0.05, while RATE-CSE performs well with a value around 0.1. Finally, we empirically show that the required total sample times for convergence is linear with respect to $|E|$ as illustrated in the third row, where the vertical dash line indicates the boundary of $|E| \times 80$ as we applied to the previous recommendation experiment. As the training time depends linearly on the sample times, it can be said that both RATE-CSE and RANK-CSE converge with less than a constant multiple of $|E|$ sample times. This demonstrates the nice scalability of CSE.

5 CONCLUSION

We present CSE, a unified representation learning framework that exploits comprehensive collaborative relations available in a user-item bipartite graph for recommender systems. Two types of proximity relations are modeled by the proposed DSEmbed and NSEmbed modules. Moreover, we propose a sampling technique to enhance the scalability and flexibility of the model. Experimental results show that CSE yields superior recommendation performance over a wide range of datasets with different sizes, densities, and types than many state-of-the-art recommendation methods.

REFERENCES

[1] Oren Barkan and Noam Koenigstein. 2016. Item2Vec: Neural item embedding for collaborative filtering. In *Workshop IEEE MLSP*.

[2] Chih-Ming Chen, Yi-Hsuan Yang, Yian Chen, and Ming-Feng Tsai. 2017. Vertex-Context Sampling for Weighted Network Embedding. *CoRR* (2017).

[3] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. 2016. Query-based Music Recommendations via Preference Embedding. In *Proc. ACM RecSys*.

[4] Wei-Sheng Chin, Bo-Wen Yuan, Meng-Yuan Yang, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. 2016. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. *Journal of Machine Learning Research* (2016).

[5] Evangelia Christakopoulou and George Karypis. 2016. Local Item-Item Models For Top-N Recommendation. In *Proc. ACM RecSys*.

[6] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. BiNE: Bipartite Network Embedding. In *Proc. ACM SIGIR*.

[7] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *Proc. ACM SIGKDD*.

[8] Yupeng Gu, Yizhou Sun, Yanen Li, and Yang Yang. 2018. RaRE: Social Rank Regulated Large-scale Network Embedding. In *Proc. WWW*.

[9] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings. In *Proc. AAAI*.

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *CoRR* (2017).

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proc. WWW*.

[12] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proc. WWW*.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proc. IEEE ICDM*.

[14] Dietmar Jannach, Paul Resnick, Alexander Tuzhilin, and Markus Zanker. 2016. Recommender systemsâĂŤbeyond matrix completion. *Commun. ACM* (2016).

[15] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proc. ACM KDD*.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. (2009).

[17] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding As Implicit Matrix Factorization. In *Proc. NIPS*.

[18] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence. In *Proc. ACM RecSys*.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proc. NIPS*.

[20] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *Proc. IEEE ICDM*.

[21] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. entity2rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation. In *Proc. ACM RecSys*.

[22] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proc. ACM SIGKDD*.

[23] Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. 2016. Walklets: Multiscale Graph Embeddings for Interpretable Network Classification. *CoRR* (2016).

[24] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Proc. NIPS*.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback]. In *Proc. UAI*.

[26] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Proc. NIPS*.

[27] Xiaoyuan Su and Taghi M Khoshgoftaar. [n. d.]. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 ([n. d.]).

[28] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proc. WWW*.

[29] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. In *Proc. WWW*.

[30] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Proc. NIPS*.

[31] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proc. ACM KDD*.

[32] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proc. ACM SIGKDD*.

[33] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: Scaling Up to Large Vocabulary Image Annotation. In *Proc. IJCAI*.

[34] Jason Weston, Hector Yee, and Ron J. Weiss. 2013. Learning to Rank Recommendations with the K-order Statistic Loss. In *Proc. ACM RecSys*.

[35] Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang. 2018. WalkRanker: A Unified Pairwise Ranking Model with Multiple Relations for Item. In *Proc. AAAI*.

[36] Wayne Xin Zhao, Jin Huang, and Ji-Rong Wen. 2016. Learning Distributed Representations for Recommender Systems with a Network Embedding Approach.. In *AIRS*.

[37] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable Graph Embedding for Asymmetric Proximity. In *Proc. AAAI*.