Learning Distributed Representations for Recommender Systems with a Network Embedding Approach

Wayne Xin Zhao[1,2(✉)], Jin Huang[1,2], and Ji-Rong Wen[1,2]

[1] School of Information, Renmin University of China, Beijing, China
{batmanfly,jin.huang,jrwen}@ruc.edu.cn
[2] Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China

Abstract. In this paper, we present a novel perspective to address recommendation tasks by utilizing the network representation learning techniques. Our idea is based on the observation that the input of typical recommendation tasks can be formulated as graphs. Thus, we propose to use the k-partite adoption graph to characterize various kinds of information in recommendation tasks. Once the historical adoption records have been transformed into a graph, we can apply the network embedding approach to learn vertex embeddings on the k-partite adoption network. Embeddings for different kinds of information are projected into the same latent space, where we can easily measure the relatedness between multiple vertices on the graph using some similarity measurements. In this way, the recommendation task has been casted into a similarity evaluation process using embedding vectors. The proposed approach is both general and scalable. To evaluate the effectiveness of the proposed approach, we construct extensive experiments on two different recommendation tasks using real-world datasets. The experimental results have shown the superiority of our approach. To the best of our knowledge, it is the first time that a network representation learning approach has been applied to recommendation tasks.

Keywords: Recommender systems · Network embedding · Item recommendation · Tag recommendation

1 Introduction

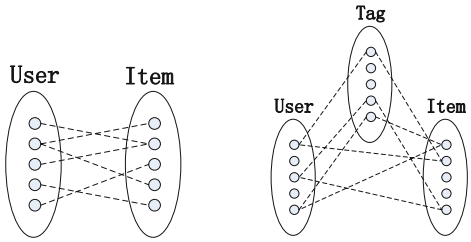
In recent years, recommender systems have played an important role in helping match users with information resources [4]. Various recommendation algorithms have been developed in the past years [1], including collaborative filtering methods, content-based methods, and hybrid methods. Collaborative filtering methods build a model from a user's past behaviors as well as decisions made by other similar users. Content-based methods extract a set of important features of an item in order to recommend new items with similar features. These two

S. Ma et al. (Eds.): AIRS 2016, LNCS 9994, pp. 224–236, 2016.
DOI: 10.1007/978-3-319-48051-0_17

types of methods are often combined in practical systems to form the hybrid methods. Although previous methods have been shown to be effective to some extent, there exist several problems with these approaches. First, these methods are usually task-oriented and cannot serve as a general solution to multiple recommendation settings. It may not be easy for existing methods to adapt to a different recommendation setting. Second, existing recommendation algorithms may not be scalable to large datasets. For example, the efficiency of classic item-based KNN recommendation algorithms is largely limited by the construction of the KNN graph [4]; matrix factorization involves eigen-decomposition of the data matrix which is expensive and usually with approximation calculation [13]. Thus, how to balance generality and scalability has become an important problem in practice. The main research focus of our paper is to develop a general and scalable recommendation framework.

To address this issue, our intuition is based on the observation that the input of typical recommendation tasks can be formulated as graphs. For example, we have presented two illustrative examples for top-N item recommendation and tag recommendation respectively in Fig. 1. For item recommendation, we have two sets of vertices, namely users and items; While for tag recommendation, we have three sets of vertices, namely users, items and tags. Once we have built the graph representation, the recommendation task can be considered as a relatedness or relevance evaluation problem: given a or more query vertices, we would like to identify the most related vertices. For example, for item recommendation, the query vertex can be set to a specific user, while for tag recommendation, the query vertices can be set to a combination of a user and an item. With such a formulation, the difficulty lies in how to develop an effective way to evaluate the relatedness on the graph.

Our approach is inspired by the recent progress in network representation learning and deep learning [8,10,18]. Network representation learning characterizes a vertex in a graph with a low-dimensional dense vector, *a.k.a.*, embedding vector. Embedding representations provide a promising way to represent and extract structural patterns in the networks, and several pioneering works



(a) User-item bipartite network. (b) User-item-tag tripartite network.

Fig. 1. Illustrative examples of bipartite and tripartite networks for recommendation tasks.

have shown the effectiveness of network embedding models [10,18]. Especially, vertex similarity or relatedness can be well measured with the embedding vectors. Following this point, our general solution to recommendation tasks has been developed as a three-step procedure. In the first step, we build a k-partite adoption network which is constructed with all the historical adoption records (*e.g.,* product purchase records). Here, the term of "adoption" has been used because the recommendation task can be considered as modeling the adoption process of a user. Second, we apply the network embedding techniques to learn the embeddings for the vertices on the k-partite adoption network. Embeddings from different vertices are projected into the same latent space, where similarity measurements can be used to evaluate the relatedness between vertices (*e.g.,* cosine similarity). Finally, the recommendation task will be casted into a similarity evaluation process. For example, given a user, we can directly rank the candidate items by the cosine similarity values between the user and item embeddings.

The proposed approach is both general and scalable. On one hand, our formulation (*i.e.,* k-partite adoption network) can be used to characterize multiple recommendation settings with rich contextual information. When new contextual information is needed to consider, we can simply discretize the contextual information into discrete variables and represent them as new vertices. On the other hand, our approach utilizes the neural network models to derive the vertex embedding representations. In this case, the model is designed to optimize within local neighborhoods instead of performing global computations. This allows us to develop scalable algorithms such as stochastic gradient descent with weight sampling [18]. To evaluate the effectiveness of the proposed approach, we construct extensive experiments on two different recommendation tasks using real-world datasets. The experimental results have shown the superiority of our approach. To the best of our knowledge, it is the first time that recommendation task has been addressed by a network representation learning approach.

2 Prelimenaries

Comparing to traditional recommendation algorithm, our goal is to provide a unified approach to multiple recommendation tasks. In a general sense, the input of the recommendation task corresponds to the adoption behaviors of users. The main idea is to represent the adoption records using a k-partite network, and then embedding representations are used to represent vertices on the graph. With such representations, we can fulfill the recommendation task using simple similarity measurements. Next we introduce the preliminaries for this paper.

Definition 1. ***k-Tuple Adoption Record.*** *An adoption record can be modeled as a k-tuple: $\langle e_1, \ldots, e_j, \ldots, e_k \rangle$, where each entry e_j ($1 \leq j \leq k$) refers to the value for the j-th feature (*a.k.a. *attribute) in an adoption record. Here we require that the value of each entry must be a positive discrete value.*

Such a formulation is general to model different recommendation tasks, even with rich context information. For example, in top-N item recommendation, a pair $\langle u, i \rangle$ is used to represent the record that user u has adopted item j. While in top-N tag recommendation, a triplet $\langle u, i, t \rangle$ is used to represent the record that tag t has been given by user u on item i. It is similar to the feature coding in context-aware recommendation models such as SVDFeature [3] and libFM [11].

Definition 2. ***k-Partite Adoption Network.*** *A k-partite adoption network is a graph whose vertices are or can be partitioned into k different independent sets: edges only exist in vertices from the same set. The edge weight is a real number which indicates the importance of the corresponding link.*

Given a set of k-tuple adoption records, it is easy to construct a k-partite network. The values for the j-th attribute are characterized by the j-th vertex set in the k-partite adoption network. With loss of generality, we next present the construction of adoption graph with the settings of $k = 2$ and $k = 3$. These two cases correspond to two classic and widely studied tasks, top-N item and tag recommendation. Other cases with large values for k can be solved in a similar way, and we leave it as future work. We assume that the k-partite network is an undirected graph.

Definition 3. ***Bipartite User-Item (UI) Network.*** *Let $\mathcal{U}$ denote the set of all the users, and $\mathcal{I}$ denote the set of all the items. A bipartite user-item network can be denoted by $\mathcal{G}^{(bi)} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$, the edge set $\mathcal{E} \subset \mathcal{U} \times \mathcal{I}$, the weight matrix $\mathbf{W}$ stores the edge weights, and $W_{u,i}$ denote the link weight between a user u and an item i.*

Definition 4. ***Tripartite User-Item-Tag (UIT) Network.*** *Let $\mathcal{U}$ denote the set of all the users, $\mathcal{I}$ denote the set of all the items, and $\mathcal{T}$ denote the set of all the tags. A tripartite user-item-tag network can be denoted by $\mathcal{G}^{(tri)} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{T}$, the edge set $\mathcal{E} \subset \big((\mathcal{U} \times \mathcal{I}) \cup (\mathcal{U} \times \mathcal{T}) \cup (\mathcal{I} \times \mathcal{T})\big)$, and the weight matrix $\mathbf{W}$ stores the edge weights.*

In a bipartite UI network, we set the weight to the number that a user has adopted an item. Different from a UI network, in a UIT network, there can be three different types of edges consisting of vertices from two out of the three vertex sets, which correspond to the edge weights $W_{u,i}$, $W_{u,t}$ and $W_{i,t}$. To set these weights, we use a simple counting method, W_{e_1,e_2} is equal to the number that e_1 and e_2 occur in all the adoption records.

With the above definitions, we can perform the recommendation task by evaluating the relatedness between query vertices and candidate vertices. For example, in top-N item recommendation, a given user will be treated as the query vertex, and we search over the candidate item vertices to find out the most related ones. Next, we introduce a network embedding approach.

3 A Network Embedding Approach to Recommendation Tasks

Recently, networking representation learning is widely studied [10,18], and it provides an effective way to explore the networking structure patterns using low-dimensional embedding vectors. Not limited to discover structure patterns, network representations have been shown to be effective to serve as important features in many network-independent tasks, such as text classification [17]. In our current task, we aim to learn low-dimensional representations for the vertices on the k-partite adoption network. The embedding representation should encode important topological information and the similarity can be evaluated by these embedding vectors.

3.1 The General Network Embedding Model

Formally, we use a d-dimensional embedding vector $\boldsymbol{v}_e \in \mathbb{R}^d$ to denote the embedding representation for a vertex e on the k-partite adoption network. We first describe a general network embedding model.

Let us start with studying how to model the generative probability for an undirected edge between two vertices e_s and e_t, formally denoted as $P(e_s, e_t)$. The main intuition is if two vertices v_i and v_j form a link on the network, their networking representations should be similar. In other words, the inner product $\boldsymbol{v}_{e_s}^\top \cdot \boldsymbol{v}_{e_t}$ between the corresponding two networking representations will yield a large similarity value for two linked vertices. We define the probability of a link (e_s, e_t) by using a sigmoid function as follows

$$P(e_s, e_t) = \sigma(\boldsymbol{v}_{e_s}^\top \cdot \boldsymbol{v}_{e_t}) = \frac{1}{1 + \exp(-\boldsymbol{v}_{e_s}^\top \cdot \boldsymbol{v}_{e_t})}. \tag{1}$$

The probability $P(e_s, e_t)$ indicates that the link strength between two vertices e_s and e_t. Recall that we also have the real weights for edges, i.e., the weight matrix $\mathbf{W}$. We can also derive an empirical estimation $\hat{P}(e_s, e_t)$ as follows

$$\hat{P}(e_s, e_t) = \frac{W_{e_s,e_t}}{\sum_{(e_{s'},e_{t'})\in\mathcal{E}} W_{e_{s'},e_{t'}}}. \tag{2}$$

Following previous study on networking representation learning [18], we minimize the KL-divergence of two probability distributions

$$L(\mathcal{G}) = D_{KL}(\hat{P}(\cdot,\cdot)||P(\cdot,\cdot)) \propto \sum_{(e_s,e_t)\in\mathcal{E}} W_{e_s,e_t} \log P(e_s, e_t). \tag{3}$$

This essentially models first-order proximity modeled in the LINE model [18]. Although we can also model the second-order proximity as in LINE, the empirical results showed that the second-order did not perform well on tag recommendation. A possible reason will be that the second-order proximity mainly captures the shared contexts between vertices, and such an indirect modeling

method does not work well on recommendation task. Thus, we only model the first-order proximity of the k-partite adoption network in this work. We follow the learning method in [18] to optimize Eq. 3, which applies stochastic gradient descent with negative sampling and weight sampling. The running complexity is about $\mathcal{O}(n \cdot d \cdot \#neg \cdot M)$, where n is the iteration number, d is the number of latent factors, $\#neg$ is the number of negative samples and M is the number of training instances.

3.2 Utilizing Embedding Representations for Recommendations

In the above, we have presented how to learn networking embeddings on the k-partite adoption graph. After parameter learning, we can obtain the embeddings for each vertex on the graph. Next, we will study how to make recommendations with these embeddings. We consider two tasks, namely the top-N item recommendation and the top-N tag recommendation.

Top-N Item Recommendation with Bipartite Network Embedding. Given a user u, the first task aims to produce a candidate list of N items based on her adoption history. In this task, we have two kinds of entities in the recommendation setting, namely users and items. We follow Definition 3 to construct the bipartite user-item network $\mathcal{G}^{(bi)}$ (See Fig. 1(a)). Then we run the network embedding model shown in Sect. 3.1, and derive the embedding representations for both users and items, denoted by $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ respectively. Given a query vertex, *i.e.,* a user, we would like to identify the most related item vertices. Formally, the task can be fulfilled using the following ranking funciton

$$\text{score}(u, i) = \boldsymbol{v}_u^\top \cdot \boldsymbol{v}_i. \tag{4}$$

Given a user u, we can rank the items using Eq. 4 to generate the recommendations. Here, we do not consider repetitive adoption behaviors of users, it will be easy to adapt to the case where repetitive adoptions are considered.

Top-N Tag Recommendation with Tripartite Network Embedding. Given a user u and an adopted item i, the second task aims to produce a candidate list of N tags based on the tagging history. In this task, we have three kinds of entities in the recommendation setting, namely users, items and tags. We first follow Definition 4 to construct the tripartite user-item-tag network $\mathcal{G}^{(tri)}$ (See Fig. 1(b)). Then run the network embedding model (in Sect. 3.1) on the tripartite network, and derive the embedding representations for both users, items and tags, denoted by $\boldsymbol{v}_u$, $\boldsymbol{v}_i$ and $\boldsymbol{v}_t$ respectively. Formally, the task can be fulfilled using the following ranking function

$$\text{score}(u, i, t) = \boldsymbol{v}_u^\top \cdot \boldsymbol{v}_t + \boldsymbol{v}_i^\top \cdot \boldsymbol{v}_t. \tag{5}$$

Our ranking function follows the idea in [13], which models the three-way data by using pairwise interaction factorization.

High-Order Recommendation with Network Embedding. In more complex tasks, we can have multiple kinds of entities (*i.e.,* attributes or features) to consider. Our approach is quite general to incorporate arbitrary types of discrete features into the recommendation setting. The procedure can be described as follows. We first construct the k-partite adoption graph, and then learn the embedding representations for each vertex on the network. After obtaining the embedding representations, we can define the score functions, such as Eqs. 4 and 5, to rank candidate vertices for recommendation.

In what follows, we will call *top-N item recommendation* as *item recommendation*, and call *top-N tag recommendation* as *tag recommendation* for short.[1] We refer to our method as *Network Embedding based Recommendation Model* ***(NERM)***.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of the proposed approach in two tasks, namely item and tag recommendation.

Table 1. Summary statistics of the two datasets for item recommendation.

Datasets	#Users	#Items	#Records	Sparsity
JD	94,440	46,573	2,767,366	0.063 %
MovieLens	198,155	17,505	22,290,822	0.6426 %

4.1 Evaluation on Top-*N* Item Recommendation

Dataset. We use two shared datasets for the evaluation of item recommendation: the *JD* dataset in [24] and the *MovieLens* dataset[2]. JD dataset is a large product purchase collection, in which each adoption record consists of a user ID, a product ID and an adoption timestamp. MovieLens dataset is a large movie rating collection, in which each adoption record consists of a user ID, a movie ID and an adoption timestamp.[3] Table 1 summarizes the basic statistics of the two datasets. We select these two datasets because they are large and represent different data applications.

Evaluation Metrics. For item recommendation, we adopt four widely used evaluation metrics, including Precision@K, Recall@K, Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). Usually, only top ranked recommendations are important to consider, thus we set K to 10 in the experiments.

[1] A tag itself can be treated as an item, too. Here we follow the conventions in tag recommendation which distinguishes between an item and a tag.

[2] http://grouplens.org/datasets/movielens.

[3] The dataset was originally used for rating prediction, and we use it for item recommendation.

Table 2. Performance comparisons of the proposed method and baselines on item recommendation.

Methods	JD				MovieLens			
	P@10	R@10	MAP	MRR	P@10	R@10	MAP	MRR
BPR	0.171	0.360	0.337	0.564	0.097	0.169	0.148	0.195
DeepWalk	0.259	0.443	0.502	0.806	0.203	0.243	0.249	0.358
NERM	**0.275**	**0.477**	**0.528**	**0.819**	**0.206**	**0.256**	**0.258**	**0.368**

Experimental Setting. Since each adoption record is attached with a timestamp, we consider a time-sensitive evaluation. We split the entire dataset by timestamps: the first 80 % data is used as training data while the rest 20 % data is used as test data.[4]

Baselines. We consider using the following methods as the comparison baselines

- **BPR** [12]. BPR is a Bayesian personalized ranking method for learning with implicit feedback. It adopts a pairwise loss function which assumes that an adopted item should be more weighted compared with an unadopted item.
- **DeepWalk** [10]. DeepWalk is a recently proposed network embedding method. It first generates multiple random paths based on a social network, and further employs the word2vec [8] to deal with vertex sequences.

The BPR method was originally proposed to solve the item recommendation task, representing state-of-the-art. DeepWalk is also a network embedding method and we incorporate it as a comparison. There can be several parameters to tune in baselines and our method. We hold out 10 % of training data as the development set for parameter optimization. For BPR, the number of latent factors is set to 256, and the number of negative samples is set to 300. For DeepWalk, the number of embedding dimensions is set to 1024 and we use the hierarchal softmax algorithm to learn the parameters. For our method NERM, the number of embedding dimensions is set to 1024, and the number of negative samples is set to 8.

Results and Analysis. Table 2 presents the experimental results of the compared methods on the task of item recommendation. Overall, we have made the following observations. First, both network embedding methods are much better than the competitive baseline BPR. Second, NERM is slightly better than DeepWalk. These results indicate the effectiveness of the network embedding approach. BPR employs a pairwise ranking function to learn the preference order using implicit feedback, and each training case is a pair consisting a positive item and a negative item. DeepWalk generates truncated random vertex sequences,

[4] The number of items in both datasets is large, and it will be quite time-consuming to consider all the unadopted items as candidate recommendations. We follow [24] to pair each adopted item with 50 negative unadopted items to form the candidate recommendation list.

and derive the embeddings by using a hierarchical softmax. Compared to these two methods, NERM directly optimizes each edge (*i.e.,* each user-item adoption record) in the network and adopt the negative sampling as the optimization method.

4.2 Evaluation on Top-*N* Tag Recommendation

Dataset. We use two shared datasets in [13] for the evaluation of tag recommendation. These two datasets have been widely used for tag recommendation. Different from [13], we do not perform p-core filtering. Table 3 summarizes the basic statistics of the two datasets.

Table 3. Summary statistics of the two datasets for tag recommendation.

Datasets	#Users	#Items	#Tags	#Records
Last.fm	1,893	12,524	9,750	186,479
Bookmarks	1,868	69,224	40,898	437,593

Experimental Setting. For tag recommendation, we following the same experimental setting in [13] for evaluation. We adopt Precision@K, Recall@K and F@K as the evaluation metrics. For each user, we take the last annotated item together with the attached tags as the test data, while the rest tagging records are used as training data.

Baselines. We consider using the following methods as the comparison baselines

- **PITF** [13]. PITF is a factorization model for tag recommendation, that explicitly models the pairwise interactions (PITF) between users, items and tags. The model is learned with an adaption of the Bayesian personalized ranking (BPR) criterion which originally has been introduced for item recommendation.
- **DeepWalk** [10]. It is similar to that is described in previous experiments on item recommendation.

PITF represents a competitive baseline for tag recommendation[5]. As shown in [13], PITF is better than FolkRank [5] on the two datasets, thus we do not

Table 4. Performance comparisons of the proposed methods and baselines on tag recommendation.

Methods	Last.fm						Bookmarks					
	P@1	R@1	F@1	P@5	R@5	F@5	P@1	R@1	F@1	P@5	R@5	F@5
PITF	0.305	0.125	0.178	0.189	0.351	0.245	0.381	0.132	0.197	0.204	0.304	0.244
DeepWalk	0.088	0.044	0.059	0.040	0.099	0.057	0.064	0.024	0.035	0.038	0.074	0.050
NERM	**0.327**	**0.165**	**0.220**	0.182	**0.370**	0.244	**0.396**	**0.135**	**0.201**	**0.228**	**0.323**	**0.267**

[5] We do not compare with other methods with item contents or temporal information.

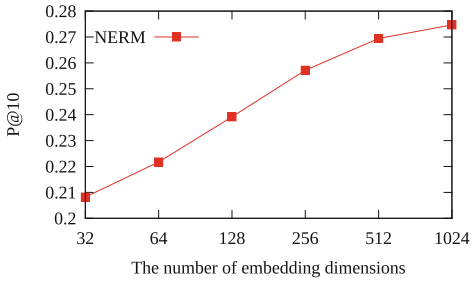


Fig. 2. Varying the number of embedding dimensions for item recommendation on JD dataset.

compare with FoldRank here. We hold out 10 % of items in the training set as the development set for parameter optimization. For PITF, the number of latent factors is set to 256, and the number of negative samples is set to 200. For DeepWalk, the number of embedding dimensions is set to 128 and we use the hierarchal softmax algorithm to learn the parameters. For our method NERM, the number of embedding dimensions is set to 128, and the number of negative samples is set to 200.

Results and Analysis. Table 4 presents the experimental results of the compared methods on the task of tag recommendation. Overall, we have made the following observations. First, the proposed method NERM nearly performs best for all the entries, slightly worse than the state-of-the-art method PITF on Last.fm in terms P@5 and F@5. Second, the network embedding method DeepWalk performs poorly on the tag recommendation task. By combining the results on item recommendation, we can conclude that NERM is effective to deal with recommendation tasks as a general method. DeepWalk does not perform well on tag recommendation, a possible reason is that it may require more principled random walk methods on k-partite graphs. Currently, we follow [10] to use a uniform sampling method, however, such a method may not be suitable to k-partite graphs. For example, it is likely that vertices in some independent set cannot be well covered even with many random paths. We will investigate into it as a future work.

4.3 Parameter Tuning

In our model NERM, an important parameter to tune is the number of embedding dimensions. We vary it in the set $\{32, 64, 128, 256, 512, 1024\}$ and see how it affects the performance. The tuning results are shown in Figs. 2 and 3. As we can see that, for item recommendation, we need to set a large number; While for tag recommendation, the optimal number is set to 128. The major reason is that the two datasets used for item recommendation are much larger than those used in tag recommendation.

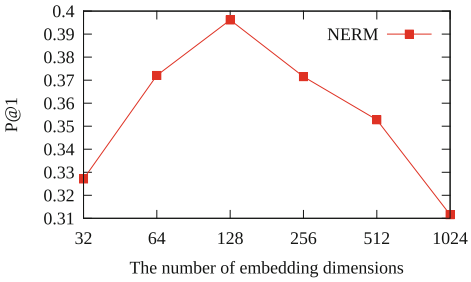


Fig. 3. Varying the number of embedding dimensions for tag recommendation on Bookmarks dataset.

5 Related Work

Recommender Systems. In the literature of recommender systems, two widely studied tasks are *rating prediction* and *top-N recommendation*. Rating prediction aims to predict the ratings from users to items, while top-N recommendation aims to generate a short list of recommendations for users [4]. Our focus in this paper is top-N recommendation. There are three typical approaches for top-N recommendation. First, rating prediction methods were directly applied where the predicted rating value was used for ranking [9,23]. Second, implicit feedback information was utilized to improve the recommendation performance, such as the weighting-based method [6]. Thirdly, specific loss function in the optimization objective was developed, including AUC-based loss function [7,12] and MAP-based loss function [14]. Tag recommendation can be considered as a special task for top-N recommendation, where tag are recommended to a user on a specific item. Various methods have been proposed for tag recommendation, including random walk methods [5], time-sensitive methods [15], higher order singular value decomposition [16], and pairwise interaction tensor factorization [13]. Recently, several context-aware models have been also proposed in order to utilize complex contextual information for rating prediction [3,11]. Different from previous studies, we aim to build a general and scalable recommendation framework for top-N recommendation, and present a new perspective by applying the network embedding techniques.

Distributed Representation Learning. Recent years have witnessed the great success of distributed representation learning and neural networks. It provides an effective way to represent and extract useful knowledge in many tasks, including text classification [17], knowledge graph mining [22] and recommender systems [21]. Especially, a promising direction is network embedding with distributed representation learning [10,18]. For example, DeepWalk [10] adapted Skip-Gram [8], a widely used language model in natural language processing area, for network representation learning on truncated random walks. LINE [18] is a scalable network embedding algorithm which modeled the first-order and second-order proximities between vertices. More recently, heterogenous network

embedding [2] or focus on deep network embedding [19] have been studied. We are also aware that several works have applied distributed representation learning [20] or neural network models [21] to recommendation tasks. Our work is highly built on these studies. The novelty lies in the idea which casts the recommendation task into a network embedding task. To our knowledge, it is the first time that network embedding methods have been applied to recommendation tasks.

6 Conclusion and Future Work

In this paper, we made the first attempt that utilized the network representation learning techniques for recommendation tasks. We first transformed the adoption records into a k-partite adoption network, then learned distributed representations for the vertices, and finally calculated the embedding similarity for recommendation. To evaluate the effectiveness of the proposed approach, we constructed extensive experiments on two different recommendation tasks using real-world datasets. The experimental results have shown the superiority of our approach. To the best of our knowledge, it is the first time that a network representation learning approach has been applied to recommendation tasks. Currently, we adopt a simple network architecture for efficient parameter learning. In the future, we consider employing more complex deep neural networks [2,19] for recommender systems. We will also test how the current framework performs on context-aware recommendation which involves multiple kinds of contextual information, such as users' demographics and items' reviews.

Acknowledgements. The authors thank the anonymous reviewers for their valuable and constructive comments. The work was partially supported by National Natural Science Foundation of China under the grant number 61502502 and Beijing Natural Science Foundation under the grant number 4162032.

References

1. Bobadilla, J., Ortega, F., Hernando, A., GutiéRrez, A.: Recommender systems survey. Know. Based Syst. **46**, 109–132 (2013)
2. Chang, S., Han, W., Tang, J., Qi, G., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: SIGKDD, pp. 119–128 (2015)
3. Chen, T., Zhang, W., Lu, Q., Chen, K., Zheng, Z., Yu, Y.: Svdfeature: a toolkit for feature-based collaborative filtering. J. Mach. Learn. Res. **13**, 3619–3622 (2012)
4. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inform. Syst. (TOIS) **22**(1), 143–177 (2004)
5. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Folkrank: a ranking algorithm for folksonomies. In: LWA 2006, pp. 111–114 (2006)
6. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: ICDM, pp. 263–272 (2008)
7. Kabbur, S., Ning, X., Karypis, G.: Fism: factored item similarity models for top-n recommender systems. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 659–667. ACM (2013)

8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
9. Ning, X., Karypis, G.: Slim: sparse linear methods for top-n recommender systems. In: IEEE 11th International Conference on Data Mining, pp. 497–506. IEEE (2011)
10. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of SIGKDD (2014)
11. Rendle, S.: Factorization machines with libfm. ACM TIST **3**(3), 57 (2012)
12. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461 (2009)
13. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM, pp. 81–90 (2010)
14. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., Oliver, N.: Tfmap: optimizing map for top-n context-aware recommendation. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 155–164. ACM (2012)
15. Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W., Giles, C.L.: Real-time automatic tag recommendation. In: SIGIR, pp. 515–522 (2008)
16. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: RecSys (2008)
17. Tang, J., Qu, M., Mei, Q.: Pte: Predictive text embedding through large-scale heterogeneous text networks. In: SIGKDD (2015)
18. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW (2015)
19. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: SIGKDD (2016)
20. Wang, H., Wang, N., Yeung, D.: Collaborative deep learning for recommender systems. In: SIGKDD, pp. 1235–1244 (2015)
21. Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., Cheng, X.: Learning hierarchical representation model for nextbasket recommendation. In: SIGIR (2015)
22. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: IJCAI (2015)
23. Yang, X., Steck, H., Guo, Y., Liu, Y.: On top-k recommendation using social networks. In: Proceedings of the Sixth ACM Conference on Recommender Systems, pp. 67–74. ACM (2012)
24. Zhao, W.X., Wang, J., He, Y., Wen, J., Chang, E.Y., Li, X.: Mining product adopter information from online reviews for improving product recommendation. TKDD **10**(3), 29 (2016)